Universidad de los Andes
Sede Caribe

https://mic2019.uniandes.edu.co/

SPONSORS

Universidad de los Andes Colombia

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

ASOCIO
Asociación Colombiana de Investigación Operativa

EU/ME
METAHEURISTICS

Fundación Universidad de Antioquia

XIII Metaheuristics International Conference
MIC 2019

BOOK OF ABSTRACTS

Cartagena (Colombia) – July 28-31, 2019

MIC STEERING COMMITTEE

Fred Glover University of Colorado and OptTek Systems (USA)
Celso Ribeiro Universidade Federal Fluminense (Brazil)
Éric Taillard University of Applied Sciences of Western Switzerland (Switzerland)
Stefan Voss University of Hamburg (Germany)

ORGANIZING COMMITTEE

Andrés L. Medaglia Universidad de los Andes (Colombia) – Chair
Alfaima L. Solano blanco Universidad de los Andes (Colombia)
Vanina Jacob Dávila Universidad de los Andes (Colombia)
John William Rodríguez Universidad de los Andes (Colombia)
Julián Organista Universidad de los Andes (Colombia)
Alejandro Palacio Álvarez Universidad de los Andes (Colombia)
Marcelo Botero Gómez Universidad de los Andes (Colombia)
Samuel Rodríguez Universidad de los Andes (Colombia)
Sebastián Castellanos Universidad de Antioquia (Colombia)
Freddy Orozco Universidad de los Andes (Colombia)
Web Site Design
Wendy Kryx Gomez Aldana Universidad de los Andes (Colombia)

PROGRAM COMMITTEE

Andrés L. Medaglia Universidad de los Andes (Colombia) Co-Chair
Jorge E. Mendoza HEC Montréal (Canada) Co-Chair
Rubén Ruiz Universitat Politècnica de València (Spain) Co-Chair
Juan G. Villegas Universidad de Antioquia (Colombia) Co-Chair

David Álvarez Martínez Universidad de los Andes (Colombia)
Ramón Álvarez-Valdés Universitat de València (Spain)
Lionel Amodeo Université de Technologie de Troyes (France)
Christian Blum Universitat Autònoma de Barcelona (Spain)
Luciana Buriol Universidade Federal do Rio Grande do Sul (Brazil)
Marco Caserta IE University (Spain)
Jean-François Cordeau HEC Montréal (Canada)
Mauro Dell'Amico Università degli Studi di Modena e Reggio Emilia (Italy)
Karl Dörner Universität Wien (Austria)
Abraham Duarte Universidad Rey Juan Carlos (Spain)
Rachid Ellaia Mohammed V University (Morocco)
Javier Faulin Universidad Pública de Navarra (Spain)
Paola Festa Università degli Studi di Napoli (Italy)
Andreas Fink Helmut-Schmidt-Universität Hamburg (Germany)
Frederic Gardi LocalSolver (France)
Martin Josef Geiger Helmut-Schmidt-Universität Hamburg (Germany)
Michel Gendreau École Polytechnique de Montréal (Canada)
Bruce Golden University of Maryland (United States)

José Luis González-Velarde Instituto Tecnológico de Monterrey (Mexico)
Peter Greistorfer Karl-Franzens-Universität Graz(Austria)
Saïd Hanafi Université Polytechnique des Hauts-de-France (France)
Jin-Kao Hao Université d'Angers (France)
Geir Hasle SINTEF (Norway)
Johann Hurink University of Twente(Netherlands)
Laetitia Jourdan University of Lille 1, INRIA Lille (France)
Ángel A. Juan Universitat Oberta de Catalunya (Spain)
Graham Kendall University of Nottingham (United Kingdom)
Rhyd Lewis Cardiff University (United Kingdom)
Vittorio Maniezzo Università di Bologna (Italy)
Rafael Martí Universitat de València (Spain)
Simone Martins Universidade Federal Fluminense (Brazil)
Antonio Mauttone Universidad de la República (Uruguay)
Kaisa Miettinen University of Jyväskylä (Finland)
Nenad Mladenovic Serbian Academy of Sciences and Arts (Serbia)
Jairo Montoya-Torres Universidad de la Sabana (Colombia)
Nysret Musliu Technische Universität Wien (Austria)
Frank Neumann University of Adelaide (Australia)
José Fernando Oliveira Universidade do Porto / INESC TEC (Portugal)
Ibrahim Osman American University of Beirut (Lebanon)
Mario Pavone Università degli Studi di Catania (Italy)
Erwin Pesch Universität Siegen (Germany)
Jean-Yves Potvin Université de Montréal (Canada)
Christian Prins Université de Technologie de Troyes (France)
Jakob Puchinger SystemX / CentraleSupélec (France)
Guenther Raidl Technische Universität Wien (Austria)
Helena Ramalhinho Lourenço Universitat Pompeu Fabra (Spain)
Andreas Reinholz German Aerospace Center (Germany)
Mauricio Resende Amazon (USA)
Celso Ribeiro Universidade Federal Fluminense (Brazil)
Roger Z. Ríos Mercado Universidad Autónoma de Nuevo León (Mexico)
Luiz Satoru Ochi Universidade Federal Fluminense (Brazil)
Andrea Schaerf Università degli Studi di Udine (Italy)
Marc Sevaux Université de Bretagne-Sud (France)
Patrick Siarry Université Paris-Est Créteil (France)
Thomas Stützle Université Libre de Bruxelles (Belgium)
Anand Subramanian Universidade Federal da Paraíba (Brazil)
El-Ghazali Talbi University of Lille (France)
Michael Trick Carnegie Mellon University (United States)
Tommaso Urli Satalia (UK)
Pascal Van Hentenryck Georgia Institute of Technology (United States)
Ana Viana INESC TEC Porto (Portugal)
Mutsunori Yagiura Nagoya University (Japan)
Farouk Yalaoui Université de Technologie de Troyes (France)
Nicolas Zufferey Université de Genève (Switzerland)

Table of contents

A GRASP metaheuristic for a districting problem arising in urban distribution

Andrés Tejada, Olga Usuga, Juan G. Villegas[1],

[1] Universidad de Antioquia
Calle 70 No. 52 – 21. 050010, Medellin, Colombia
omar.tejada@udea.edu.co, olga.usuga@udea.edu.co , juan.villegas@udea.edu.co

Abstract

This paper presents a simple greedy randomized adaptive search procedure (GRASP) embedded in a decision support system for the design of delivery districts in urban distribution. The proposed GRASP aims at balancing the workload of delivery routes by considering both the travel time within customers and the service time at the customers. To compute travel times we use a continuous approximation of the length of a travelling salesman tour for each district. Computational experiments, with the data of a major Colombian company serving hundreds of stores a day, reveal a potential reduction of the workload imbalance of their routes by more than 50%.

1 Introduction

Urban distribution/City logistics is about finding efficient and effective ways to transport goods in urban areas while taking into account the negative effects on congestion, safety, and environment.[1]. Urban distribution in emerging markets faces additional challenges due to the important market share of the traditional channel. This channel comprises family owned business that range from mini-stores (15-40 sq. ft. of store surface) to nano-stores (less than 15 sq. ft. of store surface) and street carts [2]. While planning the distribution process to these stores theirs suppliers (dairy, food, and soft-drink producers among others) have to partition the stores into territories or districts in order to assign their marketing (sellers) and delivery (trucks) resources efficiently [3]. In this work we describe the metaheuristic embedded in a decision support system employed by a major Colombian food company for the districting of their customers.

1.1 The districting problem

The company under study serves hundreds of customers (modern retail stores, mini- and nano-stores) a day in all major Colombian cities. In their tactical planning process, they solve a districting problem (DP) that aims to have an even workload for the crews assigned to their delivery territories. The DP they solve can be defined as follows. Given a set of customers $\mathcal{C}$, each one with a service time $t_i, (i \in \mathcal{C})$, and a set of districts $\mathcal{K}$,. DP assigns each customer to one and only one delivery district, by partitioning the set of customers into $|\mathcal{K}|$ disjoint subsets. Binary variables $x_{ik}, i \in \mathcal{C}, k \in \mathcal{K}$ define such partitioning ($x_{ik} = 1$,.if customer i is assigned to district k and $x_{ik} = 0$, otherwise). Using these variables, the workload for a given district k is defined by equation (1). The first term of this equation takes into account the total service time whereas the second term ($tt(k)$), corresponds to the approximation of the total travel time of the route serving customers on district k.

$$w_k = \sum_{i \in \mathcal{C}} t_i x_{ik} + tt(k), \forall k \in \mathcal{K} \quad (1)$$

Following, the approach proposed in [4], we use the classical Beardwood–Halton–Hammersley formula to calculate the expected travel time of the tour serving the customers assigned to a district [5]. Then, given the number of customer assigned to the district (n_k) and the area of the district (A_k), equation (2) defines the value of $tt(k)$. In this expression, β is a constant that we calculate empirically using the data from the company.

$$tt(k) \approx \beta\sqrt{A_k n_k}, \quad (2)$$

Finally, the proposed DP seeks to minimize a composite objective function with two terms: (i) the

average workload $\left(\overline{w} = \frac{\sum_{k\in\mathcal{K}} w_k}{|\mathcal{K}|}\right)$, and the average absolute deviation with respect to $\overline{w}$ $\left(\overline{\Delta} = \frac{\sum_{k\in\mathcal{K}}|w_k - \overline{w})|}{|\mathcal{K}|}\right)$.

2 A GRASP for districting stores in urban distribution

Districting problems have been studied in different applications ranging from political districting, to sales territory design [6]. To solve these problems the preferred solution methods are metaheuristics, including adaptive large neighborhood search [4], tabu search [7], greedy randomized adaptive search procedures (GRASP) [8], among others. In view of that, to solve the DP we implemented a simple GRASP [9] comprising a greedy randomized construction phase (`BuildDistricts`) and a local search phase (`ImproveDistricts`) that repeat over `T` iterations. These two phases aim at generating compact districts with a balanced workload between delivery districts.

2.1 Greedy randomized construction

The construction phase seeks to create compact districts taking into account only the service time of the customers assigned to them. To do so, procedure `BuildDistricts` creates districts sequentially using a simple nearest neighbor heuristic. To build a district, it first selects a random seed customer and then adds the unassigned customer that is closer to the centroid of the district. The insertion of customers in the district iterates until its cumulative service time exceeds the ideal value $\overline{t} = \frac{\sum_{i\in\mathcal{C}} t_i}{|\mathcal{K}|}$. Note that each time a customer is assigned to a district, the procedure `BuildDistricts` updates the set of unassigned customers, the centroid of the district and its cumulative service time. Once $\overline{t}$ is surpassed, this procedure closes the current district and selects randomly an unassigned seed customer to build a new district. The construction phase stops once $|\mathcal{K}|$ districts has been created. In the last district, the procedure ignores the limit $\overline{t}$ and assigns all remaining unassigned customers.

2.2 Local search

In the local search phase, we use equation (2) to update the workload of each district by considering the second term of equation (1) (i.e., its travel time). Then, in a greedy fashion procedure `ImproveDistricts` selects the district k' with the maximum deviation to $\overline{w}$ and tries to reduce the difference. If $w_{k'} < \overline{w}$, the procedure select the customer c' not assigned to district k' that is closer to it centroid and reassigns this customer to it. On the other hand, if $w_{k'} > \overline{w}$, the procedure takes the customer c' assigned to $w_{k'}$ that is farther to its centroid and reassigns it to another district. When reassigning c' the procedure selects the closer district (i.e., the district different from k' with the closest centroid). In the local search phase we implement a nonmonotone GRASP [10], therefore `ImproveDistricts` stops after `u` non-improving movements.

3 Case study results

The proposed GRASP has been implemented using Visual Basic for Applications (VBA) and is embedded in a decision support system with Excel as main user interface. To calculate the relevant geographic information of the customers and to display the results we use their Microsoft Power Maps extension. A run of the proposed GRASP takes about five minutes to obtain a solution in a standard desktop computer. To evaluate the effectiveness of our GRASP, we compared the current districting of the company with respect to the one obtained with the proposed approach in a small Colombian city with 800 customers (stores). The current districting of the company is mainly obtained using the Network Analyst toolbox for ArcGIS to obtain compact districts but ignoring the balancing of the workloads. Table 1 summarizes the results of this comparison. The table presents the values in minutes for $\overline{w}$ and $\overline{\Delta}$. To illustrate the improvement in the work balance it also presents the maximum deviations above (Δ^+) and below (Δ^-) $\overline{w}$, and the range ($R_w = \Delta^+ + \Delta^-$) of w_k over all districts. Using a GRASP for the districting of the company reduced the average absolute deviation of the workload by 55.5%,

from 63 minutes to 28 minutes, a similar improvement is achieved (54.8%) if the imbalance is measured using R_w (i.e., the range of the workloads). Moreover, Figure 1 illustrates the districts obtained with the proposed approach showing that this solution has a better workload balance without losing the compactness of the districts. Additional results in a test instance from a larger city with 3000 customers will be presented at the conference.

Performance measure (minutes)	$\overline{w}$	$\overline{\Delta}$	Δ^+	Δ^-	R_w
Company	496	63	67	57	124
GRASP	491	28	23	33	56
Improvement	1.0%	55.5%	65.7%	42.1%	54.8%

Table 1: Comparison of the GRASP results with the districting of the company in a small city

Figure 1. Delivery districts for a small city with 800 customers

References

[1] Martin Savelsbergh and Tim Van Woensel, T. 50th anniversary invited article—city logistics: Challenges and opportunities. *Transportation Science*, 50(2): 579–590, 2016

[2] Edgar Blanco and Jan Fransoo. Reaching 50 million nanostores: Retail distribution in emerging megacities. Technical Report 404 Technische Universiteit Eindhoven: Eindhoven. 2013

[3] J. Fabián López-Pérez, and Roger Z. Ríos-Mercado. Embotelladoras ARCA uses operations research to improve territory design plans. *Interfaces* 43(3): 209–220, 2013.

[4] Hongtao Lei, Gilbert Laporte, Yajie Liu and Tao Zhang. Dynamic design of sales territories. *Computers & Operations Research*, 56: 84–92, 2015

[5] Anna Franceschetti, Ola Jabali, and Gilbert Laporte. Continuous approximation models in freight distribution management. *Top* 25(3): 413–433, 2017.

[6] Jörg Kalcsics. Districting problems. In: Gilbert Laporte, Stefan Nickel and Frascisco Saldanha da Gama Editors. *Location science*. Springer, pages 595-622, 2015.

[7] Alain Hertz and Nadia Lahrichi. A patient assignment algorithm for home care services. Journal of the Operational Research Society, 60(4): 481–495, 2009.

[8] Roger Z Rios-Mercado, and Hugo Jair Escalante. GRASP with path relinking for commercial districting.*Expert Systems with Applications* 44:102–113, 2016.

[9] Mauricio GC Resende, and Celso C. Ribeiro. *Optimization by GRASP*. Springer, 2016.

[10] Marianna De Santis, Paola Festa, Gianluigi Liuzzi, Stefano Lucidi and Francesco Rinaldi. A nonmonotone GRASP. *Mathematical Programming Computation*

[11] Daniel Merchán and Mathias Winkenbach. An empirical validation and data-driven extension of continuum approximation approaches for urban route distances. *Networks*. DOI: 10.1002/net.21874, 2019.

A practical application of a long haul freight transportation problem with transhipments

J. Isaac Pemberthy R.[1a], Juan E. Muriel[2], Juan G. Villada[1b],

[1] Instituto Tecnológico Metropolitano
Calle 73 No. 76A - 354, Vía al Volador, Medellín, Colombia
[a] jorgepemberthy@itm.edu.co, [b] juanvillada@itm.edu.co

[2] Centre for Supply and Logistics, Deakin University
221 Burwood Hwy, Burwood VIC 3125, Australia
jmuriel@dekin.edu.au

Extended abstract

1 Introduction

The increase of free trade agreements between nations and the formation of transnational trading blocs have increased the pressure over transportation systems. As stated by the Commission of the European Communities, costs such as transport and storage represent 10-15% of the final cost of a finished product [2]. Transportation systems are involved in virtually all products and services produced in the economy [4], doubling their energy use in the last 30 years [3] and being responsible for approximately 62% of the world oil consumption [6]. According to the International Energy Agency, if no more policies are adopted to support efficiency, the global transport sector could increase its energy consumption up to 70% by 2050, showing the great importance of transportation related activities [3].

Despite all the economic benefits generated by international free trade agreements, they put an enormous pressure over logistic operators, particularly long-haul freight transportation carriers. For these companies the challenges imposed by long-distance trips, strict time windows, location and relocation of vehicle fleets, the differences among transportation regulations between countries, the use of transshipments operations across borders and a fierce competition, entails the search, development and application of methods and techniques for strategic planning and operational efficiency. Therefore, the main objective of this paper is the development of a solution approach for a real transnational, cross-border, long-haul problem for a transportation carrier in Colombia, Latin America.

2 Problem statement

This work faces a long-haul, transnational, cross-border transportation problem with transshipments applied to a transportation carrier in Colombia, Latin America. The case study company has a combination of challenges that are still rare in the literature, such as time windows, a vast fleet and different transport regulations between different countries. The problem involves a combination of different variants of the Vehicle Routing Problem (VRP), including the Pickup and delivery problem (PDVRP), the Vehicle Routing Problem with Time Windows (VRPTW), the Heterogeneous Fleet Vehicle Routing Problem (HFVPR), the Multi Depot Vehicle Routing Problem (MDVRP), the Vehicle Routing Problem with Backhauls (VRPB) and the Roll-on Roll-off Vehicle Routing Problem (RRVRP).

The problem consists on different sets of tractors and trailers sparsed in predefined in road network. This is the main difference with classical VRP's, where the whole fleet is located at a single depot. The objective is to pick-up an order request from one location and deliver it to another location. An order consists in the transportation of a specified freight which must be fitted in a predetermined tractor size. The different tractor types and trailers are indicated in Table 1 and Table 2. A tractor may need a trailer

or can be a rigid tractor (no trailer needed). The trailers can be wooden, cabined or flat beds. When a customer submits a request, it specifies the pickup and delivery locations, and a pickup time window. A tractor type is assigned to the request and a transshipment (if necessary) and pickup and delivery paths are known in advance. From this point several options are available.

Table 1. Types of tractors.

Tractor Type		Capacity (ton)	Trailer enabled
Rigid	Box tractor (2 axle)	5	No
	Single tractor (2 axle)	8	No
	Double trailer[1] (5 axle)	17	No
Tractor	Tractor (2 axle)	18	Yes
	Tractor (3 axle)	35	Yes

Table 2. Types of trailers

Trailer type	Compatibility
Wooden trailer	Tractor (2 axle)
	Tractor (3 axle)
Cabined	Tractor (2 axle)
	Tractor (3 axle)
Flat bed	Tractor (2 axle)
	Tractor (3 axle)

The objective is to reduce the number of empty vehicle journeys for both third party and company owned resources, achieving a greater loyalty of vehicle owners to the transport company since is one of the most significant resources that impacts transportation carriers in Colombia. Different constraints are represented by the diversity of regulations between countries that directly affect the transportation service like: difference in weight limit, Enabling transit of vehicles and trailers, Modalities of transport - Border crossing (Transfer, Transshipment, Direct).

3 Solution method

The problem is solved using a combination of strategies: simulation and a metaheuristic. The metaheuristic was tested using two heuristics to generate initial feasible solutions. This method generated better solutions than the method applied by the carrier, showing that significant gains can be obtained. The solution is divided in two stages: an initial stage generates feasible solutions using a combination of random generation with a nearest neighbor algorithm, assigning requests to resources (tractors and trailers). The second stage uses the result of the first stage as the initial solution and applies Simulated Annealing (SA) [5] with a combinatorial function. Its objective is to minimize the total empty distance in the planning horizon by selecting the best feasible solution. Figure 1 shows the procedure graphically.

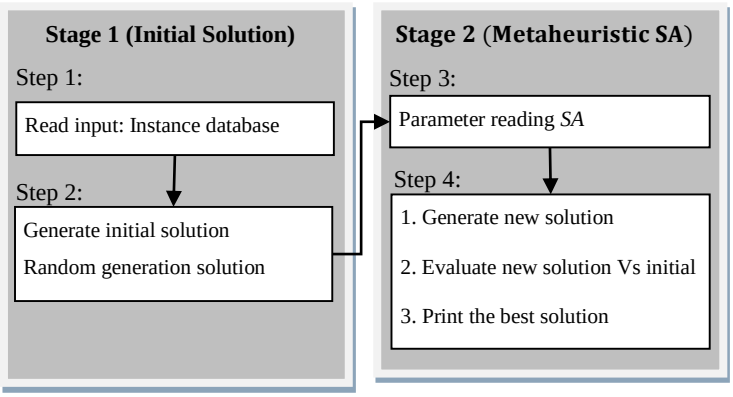


Figure 1. Illustration of the solution procedure.

[1] Despite this is a tractor with a trailer it is treated as rigid since the trailer and the tractor cannot decouple.

4 Results

We applied the proposed algorithm to a real-world problem from the carrier company with 364 service requests with pickup and deliveries in two countries. The original fleet capacity consists of tractors and trailers sparse across the transportation network. The algorithm was implemented on Visual Basic for Applications with Ms Excel in an intel core i5 with 8GB of Ram. Using the simulation, we found the baseline value of the objective function to be 125,727 total empty km travelled. We then used our solution approach with the initial random with nearest neighbor algorithm obtaining a total empty travelled distance of 90,857 km (a 27.4% reduction). The algorithm found 3,599 solutions before reaching the best value of the objective function. The total computation time was 5.2 hours (18,694 seconds).

Keywords: metaheuristics, simulation, logistic systems, vehicle routing problem, transshipments points

Reference

[1] Burnson, P. (2012), “Slow and Steady” Logistics Management, July 2012, pp. 26–39.
[2] Commission of the European communities (2006), “Freight Transport Logistics in Europe” Technical report. Brussels.
[3] Dulac, J. (2012), “Global transport outlook to 2050”, International Energy Agency Technical report.
[4] Greene, D., Jones, D.W. and Delucchi, M.A. (Eds) (1997), “The Full Costs and Benefits of Transportation”. Berlin: Springer.
[5] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). “Optimization by Simulated Annealing”. Science 220 (4598), 671-680.
[6] Li, W., Dai, Y., Ma, L., Hao, H., Lu, H., Albinson, R. and Li, Z. (2015), “Oil-saving pathways until 2030 for road freight transportation in China based on a cost-optimization model” Energy, vol. 86, no. 0, pp. 369–384.
[7] Weise, T., Podlich, A., Reinhard, K., Gorldt, C. and Geihs, K. (2009), “Evolutionary Freight Transportation Planning” in Applications of Evolutionary Computing; Tubingen, Germany; April 2009; Code 76608

A simulated annealing approach to the combined maintenance routing optimization problem for vehicles subject to failures

Eduyn López-Santana[1,*], Carlos Franco [2], German Mendez-Giraldo [1,**]

[1] Universidad Distrital Francisco José de Caldas
Carrera 7 No. 40b - 53, Bogotá D.C., Colombia
[*]erlopezs@udistrital.edu.co, [**]gmendez@udistrital.edu.co

[2] Universidad del Rosario, Escuela de Administración
Calle 200 entre carrera 7 y autopista norte, Bogotá D.C., Colombia
carlosa.franco@urosario.edu.co

Abstract

This work focuses on the problem of planning and scheduling preventive maintenance operations for a set of vehicles subject to non-deterministic failures where the set of vehicles serve a set of geographically distributed customers. This problem has real world application where the vehicles travel over long distances between cities in a difficult environment favoring a non-negligible probability of failure of critical components. To solve this problem, we propose a method that combine a simulated annealing metaheuristic and a maintenance model for a single vehicle. The maintenance model consists in a replacement model that determine the number of preventive maintenance operations for the vehicles. We present some preliminary results, comparing the proposed method with a mixed integer programming model for a set of instances.

1 Introduction

The planning and scheduling of preventive maintenance operations with routing is a problem studied recently in the literature and leads to significant improvements in the reliability of an industrial installation or a distribution network. Maintenance planning determines the set of operations, time intervals, and resources (staff, supplies, and spare parts) necessary to conduct maintenance operations [1]. López-Santana et al. [2] state the combined maintenance and routing optimization problem for the planning and scheduling preventive maintenance operations for a set of geographically distributed machines, subject to non-deterministic failures with a set of technicians that perform preventive maintenance and repair operations on the machines at the customer sites within a specific time window and planning horizon. They work integrates maintenance scheduling and routing model considering that the customer's machines fail.

We focused on the maintenance operations for vehicles that must be travelling to serve customers. This problem has real world applications where the vehicles travel over long distances between cities in a difficult environment favoring a non-negligible probability of failure of critical components [3]. In the literature, there are different applications of maintenance operations tightly coupled with vehicle routing problems, for instances Tang et al. [4] present a real-world planned maintenance scheduling problem in United Technologies Corporation; Ortiz et al. [5] present the problem of allocating maintenance jobs to a set of workers over a planning horizon in a distribution network of pumping stations; Pillac et al. [6] solve the technician routing and scheduling problem; and Goel and Meisel [7] present a combined routing and scheduling problem for the maintenance task of electricity networks.

In this paper we consider a set of customers that are geographically distributed over a region and a set of vehicles to serve the customers. These vehicles are subject to unforeseen failures that result in downtime and hence loss of productivity. To reduce the occurrence of these unforeseen downtimes, a Preventive Maintenance (PM) operation is scheduled with a certain frequency. When a vehicle fails suddenly, the maintenance crew travel to the vehicle locations and must perform a repair operation, called Corrective Maintenance (CM). Given a planning horizon, this problem consists in determining a routing-maintenance policy for all vehicles and customers. The problem aims to minimize the total routing costs (e.g., distances, times, etc.) plus maintenance cost for the routing view.

To solve this problem, we use a solution approach which integrates the model proposed by [2], called the Combined Maintenance and Routing Model (CMR) into the simulated annealing approach.

Firstly, with the assumption of zero waiting times, we determine a maintenance schedule with CMR model. Secondly, using the output from the CMR, a simulated annealing is executed to find the routes to serve all customers and perform the scheduled PM operations for the vehicles.

2 Solution approach

We consider a system with a set of customers $V_C = \{1,2, \dots, N\}$ geographically distributed over a region. Each customer must be visited once in the planning horizon and has a hard time window to be served for the vehicle. We consider a set of identical vehicles $K = \{1,2,..,m\}$, who need to visit the set of customers. The vehicles are mutually independent regarding their failure behavior and hence it is possible to determine an optimal maintenance policy for each vehicle separately. We consider a central depot indexed by 0 as the site of departure and destination for all vehicles. Given a planning horizon, this problem consists of determining a joint routing-maintenance policy for all vehicles such that it minimizes the total expected maintenance cost (PM and CM costs) and routing costs. The assumptions and conditions of the problem are that: the vehicles are mutually independent regarding their failure behavior, start in *as good as new* condition, and after the PM or CM operation they return to as good as new condition again; all vehicles start in the depot at the beginning of the planning horizon and must return to the depot by the end of the planning horizon; the mean time required to perform a CM operation is longer than the mean time required to perform a PM operation on each vehicle and the CM cost is larger than the PM cost; and the travel times are deterministic and fulfill the triangle inequality.

Our method consists in the algorithm stated in Algorithm 1. Our representation for a routing solution r_i consists in a set of arrays of integers that represents the routes for each vehicle and indicates the order to serve the customers, i.e., for [2,3,4] indicates that the vehicle serve first the customer 2, then goes to customer 3 and finally for customer 4. Our representation for a maintenance solution m_i consists in a set of arrays for each route and indicates when a PM is scheduled., i.e., if the number of PM is two then the array could be [0,2] and indicates that the PM are scheduled, for the route [2,3,4], before customer 2 and before customer 4. Both solutions, r_i and m_i, are combined in a solution s_i. We set a initial temperature and method to decrease the temperature like $T_n = aT_{n-1}$, where $a \in [0,1)$, according with [8], [9]. The operator → *neigborhoood(s)* consist in get new solutions using 2-opt and swap methods.

Algorithm 1. SA approach

r_0 →intial solution()
$m_0 \rightarrow CMR(s_0)$
$s \rightarrow combine\_solutions(r_0, m_0)$
T →Initial temperature, $i \rightarrow 0$
$s_{best} \rightarrow s$
while $T > T_{min}$ or $i < MaxIterations$, **do**
  $r_1 \rightarrow$ *neigborhoood(s)*
  $m_1 \rightarrow CMR(r_1)$
  $s_1 \rightarrow combine\_solutions(s_1, m_1)$
  if $f(s_1) \leq f(s)$ **then**
    $s \rightarrow s_1$
  else
    $q = \min\{1, e^{-\frac{f(s_2)-f(s)}{T}}\}$
    if $random(0,1) < q$ **then**
      $s \rightarrow s_1$
    end-if
  end-if
  if $f(s_1) \leq f(s_{best})$ **then**
    $s_{best} \rightarrow s_1$
  else
  $i \rightarrow i + 1$
  T →Decrease(T)
end-while

The objective function $f(s_i)$ considers the routing cost and the maintenances cost. The routing costs is computed using the routing solution r_i and for the maintenances cost use r_i and m_i into the CMR model. The CMR model proposed by [2] determine the optimal PM period (δ^*) that minimizes the maintenance cost. The total expected cost per unit time on a vehicle is given by

$$\mathcal{C}(\delta) = \frac{E[C(\delta)]}{E[T(\delta)]} = \frac{\text{Total expected cost of a cycle}}{\text{Expected cycle length}} = \frac{C_{PM}\big(1 - F(\delta)\big) + C_{CM}F(\delta)}{(\delta + T_{PM})\big(1 - F(\delta)\big) + (M(\delta) + T_{CM})F(\delta)}. \quad (1)$$

The optimal value of δ i.e., δ^* is found from equation (1) using a simple one-dimensional procedure for unconstrained nonlinear optimization. The date δ^* (minimum value of $\mathcal{C}(\delta)$) represents a balance between preventive maintenance and corrective maintenance costs. If the planning horizon has the length $\mathcal{T}$, then η is defined as the frequency of PM operations given by $\eta = \left\lfloor \frac{\mathcal{T}}{E[T(\delta^*)]} \right\rfloor$. For the j-th PM operation over the planning horizon the expected execution date is $\delta^* + (j-1)E[T(\delta^*)]$. We assume that the optimal cost of the maintenance policy remains unchanged and this model is solved for each vehicle $k \in K$.

3 Results

This section shows a numerical example of our proposed model. We use the VRP instance r101 proposed in [12]. Table 1 presents the results of SA and MIP models over four instances. In the table, we present the objective function, the routing and maintenance costs as well as the computational time in seconds. In the results we can observe that the computational times for both methods are similar and the GAP between the SA and the MIP methods are no longer than 18% and in instance 20-4-230 the SA wins to MIP because the maintenance cost is lower for SA. Also, it can be determining that over the six instances the SA method can obtain better solutions for maintenance costs in four cases, however our method couldn't obtain or improve the results of routing of the MIP method. We could be insight that the SA method is better than MIP in the maintenance model but in the routing cannot to reach the same performance, but in the running times the SA approach is faster than MIP method when the problem's size is increasing.

Instance N-K-PH	η	SA				MIP				GAP
		$f(.)$	Routing	Maintenance	Time (s)	$f(.)$	Routing	Maintenance	Time (s)	
6-2-230	5	214.63	154	60.63	5.62	199.32	154	45	3.05	7.6%
6-2-160	4	192.57	174	18.57	4.63	166.27	154	22.27	2.55	15.2%
10-4-230	10	416.46	356	60.46	5.62	396.65	306	90	5.05	5.05%
10-4-160	8	368.94	324	44.94	4.63	310.55	286	24.55	3.65	18.7%
20-4-230	16	653.4	388	265.40	6.12	773.18	328	445.18	16.65	-15.52%
20-4-160	11	557.44	388	169.44	5.80	527.86	328	199.86	12.65	5.69%

Table 1: Results of SA and MIP

References

[1] S. O. S. Duffuaa, "Mathematical models in maintenance planning and scheduling," *Maintenance, Modeling and Optimization*, pp. 39–53, Jan. 2000.

[2] E. López-Santana, R. Akhavan-Tabatabaei, L. Dieulle, N. Labadie, and A. L. Medaglia, "On the combined maintenance and routing optimization problem," *Reliability Engineering & System Safety*, vol. 145, pp. 199–214, Jan. 2016.

[3] S. Jbili, A. Chelbi, M. Radhoui, and M. Kessentini, "Integrated strategy of Vehicle Routing and Maintenance," *Reliability Engineering & System Safety*, vol. 170, pp. 202–214, Feb. 2018.

[4] H. Tang, E. Miller-Hooks, and R. Tomastik, "Scheduling technicians for planned maintenance of geographically distributed equipment," *Transportation Research Part E: Logistics and Transportation Review*, vol. 43, no. 5, pp. 591–609, Sep. 2007.

[5] J. F. Ortiz, A. Medaglia, and J. Sefair, "Metodología para la ubicación estratégica del personal técnico y programación del mantenimiento en estaciones distribuidas geográficamente," Tesis Magíster en Ingeniería industrial, Universidad de los Andes, Bogotá, 2010.

[6] V. Pillac, C. Guéret, and A. L. Medaglia, "A parallel matheuristic for the technician routing and scheduling problem," *Optimization Letters*, pp. 1–11, 2012.

[7] A. Goel and F. Meisel, "Workforce routing and scheduling for electricity network maintenance with downtime minimization," *European Journal of Operational Research*, vol. 231, no. 1, pp. 210–228, Nov. 2013.

[8] F. W. Glover and G. A. Kochenberger, *Handbook of metaheuristics*, vol. 57. Springer US, 2003.

[9] S. A. D. Dang and A. Moukrim, "Heuristic solutions for the vehicle routing problem with time windows and synchronized visits," *Optimization Letters*, pp. 511–525, 2016.

An iterated local search algorithm for solving a practical rich location routing problem

Raphael Kramer[1], Nilson F. M. Mendes[2], Giorgio Zucchi[3], Manuel Iori[2]

[1] Universidade Federal de Pernambuco
Departamento de Engenharia de Produção, Recife, Brazil
raphael.kramer@ufpe.br

[2] Università degli Studi di Modena e Reggio Emilia
Dipartimento di Scienze e Metodi dell'Ingegneria, Reggio Emilia, Italy
{nilson.mendes, manuel.iori}@unimore.it

[3] R & D division, Coopservice S.Coop.p.A.
Via Rochdale 5, Reggio Emilia, Italy
giorgio.zucchi@coopservice.it

Abstract

We present and solve a rich vehicle routing problem (RVRP) derived from two realistic cases involving the delivery of pharmaceuticals to healthcare establishments, in Italy. The studied problem is characterized by having many features and constraints, such as multiple depots, multiple periods, heterogeneous fleet, time-windows, and site-dependency (i.e., customer-vehicle incompatibilities). In addition, decisions related to the supply and the location of the depots, and the installation of warehouses in the hospitals are considered. To solve the problem we propose an iterated local search (ILS) algorithm enhanced with classical neighborhoods and auxiliary data structures from the literature. The proposed algorithm is evaluated through computational experiments by solving realistic and artificial instances We also present a sensitive analysis to evaluate the routing costs and the algorithm performance by changing the maximum number of depots allowed to be opened.

1 Introduction

Vehicle Routing Problem (VRP) is a classical combinatorial optimization problem, well-studied in the field of Operations Research. One of the highest motivation for its study is due to the large potential of applications in the real-life. For instance, applications in the solid waste, beverage, food, dairy and newspaper industries can be found in [5] and [2].

Facility location problem (FLP) is another well-studied topic in combinatorial optimization that has several applications in logistics and supply chain management. The location of warehouses, the location of intermediate consolidation points in urban distribution of goods, the location of firehouses, hospitals or mailboxes are all examples of FLPs arising in logistics management (see, e.g., [7]). In the context of vehicle routing, the location and quantity of depots are of strategical importance, as this can lead to savings (or losses) in the routing costs over a long period. A well known problem that combines VRP and FLP is the location routing problem (LRP, see, e.g., [9], [4], [10]).

Since its introduction in the literature [3], many VRP variants have been proposed. Such variants usually integrates new features or characteristics to the original version with the aim to better represent the reality. For a review on classical VRP variants we refer the interested reader to the book by [12]. According to [6], the combination of multiple features in a single VRP including strategic and tactical aspects, as well as operational restrictions, characterizes it as a *Rich Vehicle Routing Problem* (RVRP). In addition, [1] state that a RVRP must represent the most relevant attributes of a real-life vehicle routing distribution system, such as heterogeneity, multi-periodicity, and diversity of users and policies.

In this paper we study a RVRP+FLP derived from two realistic cases involving the delivery of pharmaceuticals to healthcare establishments, in Italy (Section 2). To solve the problem we propose an iterated local search (ILS) algorithm enhanced with classical neighborhoods and auxiliary data structures from the literature (Section 3). The proposed algorithm is evaluated through computational experiments by solving realistic and artificial instances involving up to 300 customers and 6 periods (Section 4). We also present a sensitive analysis to evaluate the routing costs and the algorithm performance by changing the maximum number of depots allowed to be opened (Section 4).

2 Case study description

The studied RVRP is derived from practical VRP applications faced by *Coopservice* (in a recently won public tender). *Coopservice* is an Italian company, created in 1991, whose aim is to provide industrial, commercial and hospital services such as cleaning, logistics, security and waste management. The pharmaceutical logistics is a sector of the company that ensures the correct storage and distribution of medicines and other hospital products to parapharmacy stores and hospitals.

In the recent years, motivated by the guidelines of the Italian national health system, logistic complexes (the so-called *Vast Areas*) are being created to concentrate the local healthcare units (the so-called *AUSLs* – Azienda Unità Sanitaria Locale) into a single centralized warehouse. The benefits for institutions are the reduction of business costs and the reduction of the staff dedicated to the purchasing function. Coopservice is responsible for warehouse management and distribution of some Italian regions.

Products must be delivered to a set of hospitals from one or multiple depots. Each hospital must visited by a single vehicle and the company has a heterogeneous fleet. Phisicians should be at the delivery place when the vehicle arrives to take care of the medicines, so we have customer time windows. In addition, the characteristics of some hospitals (such as location and size) may prohibit the access of certain types of vehicles. Customers have demands to be attended from Monday to Saturday, and the routes must not exceed 9 hours.

The RVRP solved is defined according to the features and constraints faced while designing the routes to deliver pharmaceutical products in two different regions of Italy. Both cases involve multiple depots, multi-period demands, heterogeneous fleet, customer time-windows, customer-vehicle incompatibilities, and maximum duration of routes. From the first case (Region 1), the depots can be classified as *main* or *auxiliary*. Auxiliary depots must be supplied by the main depots one day before the deliveries. Moreover, some hospital may be equipped with warehouses that allow the company to deliver the products one day in advance (within a larger time-window). From the second case (Region 2), the location of T depots must be chosen from a set of candidate locations. The objective consists in minimizing the total traveling distance in the whole period (6 days). The company is interested on the estimation and evaluation of different solutions based on different number of depots to be opened. Then, it can choose a solution that depends on the trade off between the cost to open a depot and the total duration of the routes.

3 Solution algorithm

The presented RVRP is solved by means of a Multi-Start Iterated Local Search (MS-ILS) algorithm that makes use of classical neighborhoods and auxiliary data structures. The algorithm restarts η_{ITER} times. At each iteration, an initial solution is built according to a greedy strategy, then the local search (LS) procedure is executed according to the *randomized variable neighborhood descent* [8, 11]. Once the LS is done, the solution is perturbed either by performing random customer swaps inter-routes or by performing random depot exchanges. The loop of LS and perturbation is executed for η_{ILS} consecutive iterations without improving the incumbent solution.

The initial solution is built as follows. Initially, T depots are chosen at random. Then, for each day d, it is initialized H_d routes containing a single hospital. Afterward, the remaining customers are inserted in the best position of the current routes, following a random order. Violations of the time window constraints are accepted, but penalized in the objective function. The neighborhoods considered in the LS are the classical swap inter-routes, swap intra-route, relocate inter-routes, relocate intra-route, 2-opt, and 2-opt*. In addition, it also considers depot-swaps and hospital inter-days relocation. To speed-up the moves evaluation, the proposed algorithm makes use of the auxiliary data structures proposed by [13] and accepts infeasible solutions with respect to time-windows and route durations (but it penalizes them in the objective function).

4 Preliminary computational results

The proposed MS-ILS was coded in C++ and executed on a single thread of an Intel Core i5-5200U 2.2GHz with 16GB of RAM, running under Linux Mint 17.2 64-bit. The following parameter values

were adopted: $\eta_{\text{ITER}} = 20$, and $\eta_{\text{ILS}} = 20$. The algorithm was evaluated by solving three sets of instances: Set 1 is composed by 6 instances derived from the first case (Region 1); Set 2 is composed by 10 instances derived from the second case (Region 2); and Set 3 is composed by 30 artificial instances. The summary of the results for the Set 3 is presented in Table 1.

Table 1: Results for the artificial instances (Set 3)

					best greedy solution		best MS-ILS solution			average MS-ILS solution				Gap (%)	
	#nodes	#dep	#per	#maxdep	dist.	$\|R\|$	dist.	#antic.	$\|R\|$	dist.	#antic.	$\|R\|$	T(s)	to greedy	to best MS-ILS
KMZI100-4	100	4	6	2	10212.90	78	5927.70	2.00	58	5994.46	4.10	60.00	12.37	-41.31	1.13
KMZI100-6	100	6	6	3	8411.04	83	5084.20	10.00	53	5293.97	15.60	58.10	11.92	-37.06	4.13
KMZI100-8	100	8	6	4	7247.55	90	5185.30	6.00	78	5403.75	7.30	81.00	13.33	-25.44	4.21
KMZI150-4	150	4	6	2	20464.90	106	12395.50	11.00	77	12628.73	16.80	82.20	40.42	-38.29	1.88
KMZI150-6	150	6	6	3	16956.80	115	10768.10	5.00	87	11057.90	12.50	91.90	43.81	-34.79	2.69
KMZI150-8	150	8	6	4	14295.90	115	9771.50	11.00	93	10082.75	14.30	95.90	41.39	-29.47	3.19
KMZI200-4	200	4	6	2	33532.90	139	19143.40	3.00	116	19437.16	8.40	118.10	107.79	-42.04	1.53
KMZI200-6	200	6	6	3	29546.00	144	15638.80	8.00	98	16246.02	17.20	104.80	94.20	-45.01	3.88
KMZI200-8	200	8	6	4	26525.30	159	16637.20	11.00	116	17462.84	17.10	120.40	99.90	-34.17	4.96
KMZI250-4	250	4	6	2	51156.70	179	29903.10	10.00	131	30389.86	20.90	136.80	165.63	-40.59	1.63
KMZI205-6	250	6	6	3	44736.60	173	25780.10	6.00	125	27553.38	18.80	128.60	163.69	-38.41	6.88
KMZI250-8	250	8	6	4	35728.80	183	24127.30	14.00	149	25067.61	20.80	152.90	214.48	-29.84	3.90
KMZI300-4	300	4	6	2	78853.00	207	45331.40	22.00	150	45701.07	29.80	153.00	332.25	-42.04	0.82
KMZI300-6	300	6	6	3	54680.20	211	34682.40	23.00	165	36526.97	28.80	169.10	274.87	-33.20	5.32
KMZI300-8	300	8	6	4	55742.00	207	36684.50	14.00	177	37521.76	19.50	180.60	294.99	-32.69	2.28
KMZI100-4	100	4	6	3	9653.47	78	5492.20	7.00	60	5562.46	11.30	62.80	11.94	-42.38	1.28
KMZI100-6	100	6	6	4	8136.07	87	4731.00	8.00	56	4950.64	13.90	60.10	12.55	-39.15	4.64
KMZI100-8	100	8	6	5	7131.98	92	5152.50	0.00	80	5303.73	2.50	82.60	13.09	-25.63	2.94
KMZI150-4	150	4	6	3	19590.80	108	11987.90	12.00	84	12196.98	17.10	86.50	39.66	-37.74	1.74
KMZI150-6	150	6	6	4	16590.40	116	10310.50	7.00	89	10623.43	13.60	94.00	44.02	-35.97	3.04
KMZI150-8	150	8	6	5	13640.60	116	9241.40	10.00	90	9809.09	13.50	95.30	40.55	-28.09	6.14
KMZI200-4	200	4	6	3	29689.10	144	17997.90	9.00	119	18754.55	14.00	124.50	111.09	-36.83	4.20
KMZI200-6	200	6	6	4	27083.80	152	14114.50	10.00	105	15282.15	17.60	108.50	88.24	-43.57	8.27
KMZI200-8	200	8	6	5	25844.00	163	15258.00	9.00	119	16549.96	14.10	122.20	108.17	-35.96	8.47
KMZI250-4	250	4	6	3	44986.50	188	26908.20	15.00	144	27265.16	25.00	147.10	148.28	-39.39	1.33
KMZI205-6	250	6	6	4	44502.30	176	26246.60	10.00	127	27291.73	18.40	133.30	160.29	-38.67	3.98
KMZI250-8	250	8	6	5	34515.20	189	23366.10	17.00	151	23787.98	22.40	158.90	209.95	-31.08	1.81
KMZI300-4	300	4	6	3	70426.80	211	39493.60	20.00	154	41272.88	31.40	158.90	333.37	-41.40	4.51
KMZI300-6	300	6	6	4	53825.80	216	34232.20	16.00	171	34880.14	25.40	173.60	267.28	-35.20	1.89
KMZI300-8	300	8	6	5	50964.50	212	33679.70	12.00	184	34831.50	22.10	187.90	301.01	-31.66	3.42

In order to evaluate the impact of the number of depots on the routing costs we solved the instance of the Region 2 considering different numbers of depots (*#maxdep*). The results are presented in Table 2.

Table 2: Results for Region 2 instances considering different number of depots (Set 1)

					best MS-ILS solution			average MS-ILS solution			Gap(%)		
	#nodes	#dep	#per	#maxdep	dist.	$\|R\|$	T(s)	dist	$\|R\|$	T(s)	dist	$\|R\|$	T(s)
R2-74-10-1	74	10	6	1	18490.80	68.00	55.89	33057.24	106.30	61.50	78.78	56.32	10.04
R2-74-10-2	74	10	6	2	10164.30	54.00	55.98	10989.63	55.70	66.65	8.12	3.15	19.06
R2-74-10-3	74	10	6	3	8947.98	54.00	59.70	9413.99	56.30	66.40	5.21	4.26	11.22
R2-74-10-4	74	10	6	4	7837.67	53.00	55.68	8435.06	54.60	65.44	7.62	3.02	17.52
R2-74-10-5	74	10	6	5	7477.41	53.00	60.64	8022.39	56.60	68.16	7.29	6.79	12.40
R2-74-10-6	74	10	6	6	6715.91	53.00	58.16	7229.68	56.00	63.57	7.65	5.66	9.30
R2-74-10-7	74	10	6	7	6618.10	55.00	54.04	6897.55	59.20	63.44	4.22	7.64	17.39
R2-74-10-8	74	10	6	8	6626.81	58.00	52.48	6865.81	63.20	60.53	3.61	8.97	15.32
R2-74-10-9	74	10	6	9	6544.30	63.00	48.70	6810.28	65.60	59.79	4.06	4.13	22.78
R2-74-10-10	74	10	6	10	6689.25	62.00	45.61	6911.16	66.80	54.42	3.32	7.74	19.31

From Table 2, we verify that the total traveling distance (i.e., routing costs) decreases as the number of depots increases. Despite the CPU time does not change significantly with the variation on the number of depots allowed to be opened, we verify that the robustness of the algorithm is better on instances with large number of depots. This is expected because when the number of depots becomes closer to the number of candidate the locations, the decisions related to depot location becomes easier and the problem concentrates on the routing decisions.

5 Conclusions and future research

We presented and solved a particular rich location routing problem derived from practical applications of pharmaceutical distribution faced by an Italian company. The preliminary results shows that the current version of the algorithm is able to provide significant cost savings when compared with those obtained by greedy methods that are commonly adopted in practice by decision makers. However, some refinement on the algorithm are still being performed with the aim to improve the robustness and the solution quality. As the problems are derived from a public tender rencently won by Coopservice, the solutions have not been implemented yet. The company plans to implement and evaluate them in the next six months.

References

[1] J. Caceres-Cruz, P. Arias, D. Guimarans, D. Riera, and A.A. Juan. Rich vehicle routing problem: Survey. *ACM Computing Surveys*, 47(2):32:1–32:28, 2014.

[2] L. C. Coelho, J. Renaud, and G. Laporte. Road-based goods transportation: a survey of real-world logistics applications from 2000 to 2015. *INFOR: Information Systems and Operational Research*, 54(2):79–96, 2016.

[3] George B. Dantzig. Discrete-variable extremum problems. *Operations Research*, 5(2):266–277, 1957.

[4] Michael Drexl and Michael Schneider. A survey of variants and extensions of the location-routing problem. *European Journal of Operational Research*, 241(2):283–308, 2015.

[5] B. L. Golden, A. A. Assad, and E. A. Wasil. Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy and newspaper industries. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, Monographs on Discrete Mathematics and Applications, pages 245–286. SIAM, Philadelphia, 2002.

[6] R. Lahyani, M. Khemakhem, and F. Semet. Rich vehicle routing problems: from a taxonomy to a definition. *European Journal of Operational Research*, 241(1):1–14, 2015.

[7] G. Laporte, S. Nickel, and F. Saldanha da Gama, editors. *Location Science*. Springer, 2015.

[8] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[9] Gabor Nagy and Said Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177(2):649–672, 2007.

[10] Maximilian Schiffer, Michael Schneider, Grit Walther, and Gilbert Laporte. Vehicle routing and location routing with intermediate stops: A review. *Transportation Science*, 2019. "Forthcoming".

[11] A. Subramanian. *Heuristic, Exact and Hybrid Approaches for Vehicle Routing Problems*. PhD thesis, Universidade Federal Fluminense, Computing Graduate Program, Niterói, Brazil, 2012.

[12] P. Toth and D. Vigo, editors. *The Vehicle Routing Problem*. Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, 2nd edition, 2014.

[13] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475–489, 2013.

The Magnifying Glass Heuristic for the Generalized Quadratic Assignment Problem

Peter Greistorfer, Rostislav Staněk, Vittorio Maniezzo

[1] Karl-Franzens-Universität
Universitätsstraße 15/E3, 8010 Graz, Austria
peter.greistorfer@uni-graz.at

[2] Karl-Franzens-Universität
Universitätsstraße 15/E3, 8010 Graz, Austria
rostislav.stanek@uni-graz.at

[3] Università di Bologna
Mura Anteo Zamboni, 7, 40127 Bologna, Italy
vittorio.maniezzo@unibo.it

Abstract

We investigate the generalized quadratic assignment problem and introduce a number of mat- and metaheuristic algorithms. Especially highlighted is an improvement procedure, a so-called magnifying glass heuristic, which has already proved to be successful for the solving of traveling salesman problems. All approaches are validated on test instances from the literature and on a generated set of random instances. Results demonstrate a very appealing computational performance, offering a promising foundation for further developments of the base concept in different contexts.

1 Introduction

In our study, we explore the so-called *Generalized Quadratic Assignment Problem* (GQAP). The GQAP has its origin in the *Linear Assignment Problem* (LAP), where a number of agents (machines, supplies) have to be assigned to a number of jobs (tasks, demands) in order to minimize the total cost service, while assignment constraints, which state that each job hast to be serviced by exactly one agent and vice versa, must be obeyed. The LAP in turn is a special case of the the *Generalized Assignment Problem* (GAP), in which assignment constraints on the supply-side are replaced with upper bounds of supplies (which may exceed the assignment unit case). Another generalisation of the LAP is the *Quadratic Assignment Problem* (QAP), which models multiplicative cost factors between agents and jobs. Finally, the GQAP can be thought of as a combination of GAP and QAP and therefore receives a quadratic component in its objective function, the same way as it is done in the QAP.

While there exist efficient polynomial algorithms for the LAP, e.g. the Hungarian method, the GAP, the QAP, and the GQAP are $\mathcal{NP}$-hard. The research topic of this work is related to both mathematical formulations of the GQAP as well as to the development of LP- and ILP- based matheuristics, which have become popular with the rise of recent powerful hardware and solvers like CPLEX or Gurobi (see [4]). The focus lies on the presentation of an improvement heuristic, which, under special conditions, is even capable of serving as a construction procedure. The general idea was coined as *Magnifying Glass Heuristic* and was already successfully applied to the *Quadratic Travelling Salesman Problem* in [6].

2 Model

The *Generalized Quadratic Assignment Problem* (GQAP) can be described by means of the following quadratic integer program. We are given $m \in \mathbb{N}$, the number of agents, $n \in \mathbb{N}$, the number of jobs, with linear costs and weights $p_{ij} \in \mathbb{R}$ and $w_{ij} \in \mathbb{R}_0^+$, where $1 \leq i \leq m$ and $1 \leq j \leq n$, respectively. The weights present resources to be spent by agent i for job j, without exceeding an available capacity $a_i \in \mathbb{R}_0^+$. Quadratic costs are defined by the product of $d_{ir} \in \mathbb{R}$, the cost factor between the agents i and r, and $f_{js} \in \mathbb{R}$, the cost factor between the resources j and s. Binary decision variables $x_{ij} \in \{0, 1\}$

determine whether a job j is served by agent i, or not. Then the GQAP is defined by the following quadratic integer program (QIP):

$$\min \sum_{i=1}^{m}\sum_{j=j}^{n} p_{ij}x_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{r=1}^{m}\sum_{s=1}^{n} d_{ir}f_{js}x_{ij}x_{rs} \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_{ij}x_{ij} \leq a_i \qquad \forall\, 1 \leq i \leq m, \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall\, 1 \leq j \leq n, \tag{3}$$

$$x_{ij} \in \{0,1\} \quad \forall\, 1 \leq i \leq m,\ \forall\, 1 \leq j \leq n. \tag{4}$$

3 Solution Approaches

The improvement process builds upon the results of a constructive solution procedure, which was in its original form presented as *Guided Adaptive Relaxation Rounding Procedure* (`GARRP`) in [3]. `GARRP` starts from the scratch, and iteratively creates partial solutions by fixing exactly one assignment $x_{ij} := 1$ in each iteration, based upon an optimized relaxation of GQAP, leaving a subset of decision variables free. These partial solutions, i.e. incomplete or partial assignments of jobs j to specific agents i, are organized within a tree, in which each tree-node represents one assignment made. So for any (partial) solution all fixed assignments left can be found by pegging links back to the root, where the latter stores the result of the first relaxation solved ([2]), underlying all variables to be free. Relaxed, i.e. continuous values x_{ij}, can be interpreted as choice-probabilities among the k-best successors of a given node. The tree is explored in a depth-first manner, which includes back-tracing if a node is fathomed.

The *Magnifying Glass Heuristic* for the GQAP, `MG-GQAP`, can be understood as a *large neighborhood search*, where a (starting) solution is partially destroyed and repaired by corresponding operators; a process, which is iteratively repeated until a local optimum is reached (see [5]). The underlying idea of our approach works as follows: we consider K randomly chosen columns (jobs) and create a new auxiliary instance containing only these K columns (and all rows). After modifying the linear costs p_{ij} and re-adjusting the total amounts of resources a_i in order to reflect the overall problem, we solve the auxiliary problem optimally and get a new, possibly improved solution with changes restricted to the K chosen columns. This process is then repeated for I iterations.

Let $z_X\big((x_{ij})\big)$ be the objective function value of the solution $(x_{ij}) \in \{0,1\}^{m\times n}$ for the instance X. Then for a given number of iterations $I \in \mathbb{N}$ and a given size of magnifying glass $K \in \mathbb{N}$, the magnifying glass matheuristic can be formally described as follows:

Require: A GQAP instance X and a solution given as $(x_{ij}) \in \{0,1\}^{m\times n}$
Ensure: New solution $(x_{ij}^{new}) \in \{0,1\}^{m\times n}$ with $z_I\big((x_{ij}^{new})\big) \leq z_I\big((x_{ij})\big)$

1: $(x_{ij}^{old}) := (x_{ij})$;
2: **for** I iterations **do**
3: Choose K columns $c_1 \neq c_2 \neq \ldots \neq c_K \in \{1,2,\ldots,n\}$ and let $C = \{c_1, c_2, \ldots, c_K\}$;
4: Create a new (auxiliary) GQAP instance X' containing all rows and the columns $c_1, c_2, \ldots, c_K$ of the instance X^{old} and having new costs and total amounts of resources instead of p_{ij} and a_i

$$p'_{ij} := p_{ij} + \sum_{r=1}^{m}\sum_{\substack{s=1\\ s\notin C}}^{n} (d_{ir}f_{js} + d_{ri}f_{sj})x_{rs}^{old} \quad \forall\, 1 \leq i \leq m,\ \forall\, j \in C \tag{5}$$

$$a'_i := a_i - \sum_{\substack{j=1\\ j\notin C}}^{n} w_{ij}x_{ij}^{old} \qquad \forall\, 1 \leq i \leq m; \tag{6}$$

5: Solve the instance X' using the QIP (1)–(4) and update the solution $(x_{ij}^{new}) \coloneqq (x_{ij}^{old}) \cup (x'_{ij})$;
6: $(x_{ij}^{old}) \coloneqq (x_{ij}^{new})$;
7: **end for**
8: **return** (x_{ij}^{new});

Algorithm 1: `MG-GQAP`

Algorithm 1 is based on the quadratic integer program (1)–(4) (see Section 2). In general, it can be used with any linearisation as well, but the model has to be slightly adapted.

4 Implementation and Preliminary Results

The basic MG-GQAP-implementation is tested on several choice criteria to systematically define different sets K, where for all of them it turned out that, to a specific amount, a random contribution is absolutely meaningful. The same holds for the inclusion of fast local search procedures based on swap- and insert-moves. Currently, the test bed of [1] with 21 instances is solved in all cases with an average deterioration of only 1.57%, which constitutes an average improvement of 6.3% over `GARRP.MG-QQAP` running times for $I = 1000$ iterations on an Intel(R) Core i7-3770K CPU @3.6GHz, Windows 10 Pro system with 16 GB Ram, lie between roughly 1 to 4 minutes. With respect to the own "soft-constrained" test bed of 20 symmetrically sized instances with sizes $10 \leq m = n \leq 100$ (in steps of 5), it should be highlighted that, without any starting solution, the `MG-GQAP` is able to serve as construction procedure and provides reasonable feasible solutions within not more than 50 minutes.

5 Summary and Outlook

In this work we offer solution approaches for the Generalized Quadratic Assignment Problem. Especially highlighted is an improvement procedure, originally coined as Magnifying Glass Heuristic, a matheuristic built upon the paradigm of joining established (meta)heuristic ideas with the strengths of mathematical programming. Attractive opportunities for more advanced tuning options include memory aspects for search guidance, the dynamic inclusion of cutting planes as well as optimizing the implementation, seen from both a code- and solver-oriented perspective. All these aspects, along with an enlargement of the test bed, are already ongoing research and will be covered in in the final presentation.

References

[1] Jean-François Cordeau, Manlio Gaudioso, Gilbert Laporte, and Luigi Moccia. A memetic heuristic for the generalized quadratic assignment problem. *INFORMS Journal on Computing*, 18:433–443, 11 2006.

[2] Leonard Kaufman and Fernand CM. Broeckx. An algorithm for the quadratic assignment problem using Bender's decomposition. *European Journal of Operational Research*, 2(3):207–211, 1978.

[3] Vittorio Maniezzo, Peter Greistorfer, and Rostislav Staněk. Exponential neighborhood search for the generalized quadratic assignment problem, 2018. EURO/ALIO International Conference on Applied Combinatorial Optimization, 25.–27.6.2018, Bologna, Italy.

[4] Vittorio Maniezzo, Thomas Stützle, and Stefan Voß. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[5] David Pisinger and Stefan Ropke. Large neighborhood search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, pages 399–419. Springer, 2010.

[6] Rostislav Staněk, Peter Greistorfer, Klaus Ladner, and Ulrich Pferschy. Geometric and LP-based heuristics for the quadratic travelling salesman problem, 2018. Accepted for publication in Comput. Oper. Res., available as: `arXiv:1803.03681`.

A heuristic for the minimum cost chromatic partition problem

Celso C. Ribeiro[1], Philippe L. F. dos Santos[2]

[1] Institute of Computing, Universidade Federal Fluminense, Niterói, RJ 24210-346, Brazil.
celso@ic.uff.br

[2] Instituto Federal de Educação, Ciência e Tecnologia Fluminense,
Campos dos Goytacazes, RJ 28030-130, Brazil.
philippeleal@gmail.com

Abstract

The minimum cost chromatic partition problem is a variant of the graph coloring problem in which there are costs associated with the colors and we seek a vertex coloring minimizing the sum of the costs of the colors used in each vertex. It finds applications in VLSI design and some scheduling problems modeled on interval graphs. We propose a trajectory search heuristic using local search, path-relinking, perturbations, and restarts for solving the problem and discuss numerical results.

1 Introduction

The minimum cost chromatic partition problem (MCCPP) is a variant of the graph coloring problem in which there are costs associated with the colors and one seeks a vertex coloring minimizing the sum of the costs of the colors used in each vertex. The problem was introduced by Mead and Conway [8] and has applications in the design of VLSI circuits [11, 12]. Kroon et al. [5] showed that MCCPP for interval graphs is equivalent to the fixed interval scheduling problem with machine-dependent processing costs.

No exact algorithms are reported for MCCPP in the literature. We propose a trajectory search with path-relinking heuristic for MCCPP. In Section 2, the minimum cost chromatic partition problem is formulated as an integer programming problem and its complexity is discussed. Section 3 presents the building blocks of the trajectory search heuristic with path-relinking. Section 4 reports on the computational experiments and numerical results.

2 Formulation and complexity

Let $G = (V, E)$ be a graph with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set $E \subseteq V \times V$. We suppose that a set $\mathcal{C} = \{C_1, \ldots, C_n\}$ of colors is available and there is a cost w_c associated with each color C_c, $c = 1, \ldots, n$. Let x_{ic} be a binary variable such that $x_{ic} = 1$ if color C_c is assigned to vertex v_i, $x_{ic} = 0$ otherwise. Seen et al. [11] formulated the minimum cost chromatic partition problem as the following 0-1 integer programming problem:

$$\min \sum_{i=1}^{n} \sum_{c=1}^{n} w_c \cdot x_{ic} \tag{1}$$

subject to:

$$\sum_{c=1}^{n} x_{ic} = 1, \ \forall i = 1, \ldots, n \tag{2}$$

$$x_{ic} + x_{jc} \leqslant 1, \ \forall i, j = 1, \ldots, n : (v_i, v_j) \in E, \ \forall c = 1, \ldots, n \tag{3}$$

$$x_{ic} \in \{0, 1\}, \ \forall i, c = 1, \ldots, n. \tag{4}$$

Constraints (2) impose that each vertex is colored with exactly one color. Constraints (3) guarantee that if some color is used, then only one of any two adjacent nodes can receive this color. The objective function (1) minimizes the sum of the costs of the color assignments.

The minimum cost chromatic partition problem was shown to be NP-hard for general graphs in [11] from a restriction to the chromatic sum problem, which was proved to be NP-hard in [6]. The next section presents a trajectory search with path-relinking heuristic for MCCPP.

3 Trajectory search heuristic with path-relinking

The trajectory search heuristic with path-relinking (TSH+PR) developed in this section makes use of a randomized greedy heuristic combined with a simple neighborhood for local search to build diversified colorings, combined with perturbation and path-relinking strategies. A short-term, one-pass tabu list is used to avoid that the current solution is revisited in the next iteration.

Any solution to MCCPP in $G = (V, E)$ may be represented by a partition S: $\langle V_1, \ldots, V_n \rangle$ of the nodes in V, such that all nodes in color class $V_j \subseteq V$ are colored with color C_j, for $j = 1, \ldots, n$. A feasible solution to MCCPP using exactly k different colors is called a proper k-coloring.

TSH+PR considers an evaluation function associated with any (proper or improper) coloring S: $\langle V_1, \ldots, V_n \rangle$ that is given by $f(S) = \sum_{c=1}^{n}(w_c \cdot |V_c| + M \cdot |E(V_c)|)$, where $E(V_c) \subseteq E$ is the subset of conflicting edges with both extremities in color class V_c and M is a sufficiently large penalty. Since $\sum_{c=1}^{n} |E(V_c)| = 0$ in any proper coloring, the penalty M should be large enough to discard the presence of conflicting edges in any solution minimizing $f(S)$.

A variant of the greedy Recursive Largest First (RLF) algorithm [7] is used to build an initial solution for MCCPP. In order to introduce diversity in the construction of the initial solution, the algorithm is interrupted when there are less than $\lceil RF \times |V| \rceil$ uncolored vertices, where the randomization factor $RF \in (0, 1)$. At this point, the uncolored vertices are randomly assigned to the color classes already created. Typically, the initial solution obtained will be an improper k-coloring.

Let S be a k-coloring obtained as the initial solution. We denote by $S(v) = i$ the color class V_i of vertex $v \in V$ and by $N_j(v) \subseteq V_j$ the subset of vertices adjacent to v in color class V_j, for any $j = 1, \ldots, k$: $j \neq S(v)$.

A proper coloring can be obtained from S by the application of a sequence of moves in the Critical One-Move Neighborhood, widely used in coloring problems [1, 3]. A neighbor solution is obtained by moving a vertex v involved in a color conflict from its color class V_i to another class V_j, with $1 \leqslant j \leqslant k + 1$, $i \neq j$, and $|N_j(v)| = 0$. The vertex selected to be moved is always the one that promotes the largest reduction in the evaluation function. Furthermore, the selected pair $\langle v, j \rangle$ cannot be classified as a tabu move (as the result of the last sequence of perturbation moves) unless it satisfies an aspiration criterion leading to a neighbor solution that improves upon the best solution found.

After a proper coloring has been obtained, a solution improvement phase attempts to reassign vertices to smaller cost color classes, while preserving feasibility. The least cost color is assigned to the largest cardinality color class [2]. The procedure attempts to reassign vertices of the next largest cardinality color class to a smaller cost color class of the solution under construction that does not create a conflict (or, otherwise, to a new color class with the yet unused least cost color).

Backward path-relinking is applied by TSH+PR between the proper coloring previously obtained and a solution randomly selected from the elite set.

The last phase of each iteration amounts to a diversification strategy that creates perturbations after a feasible solution has been obtained and improved by path-relinking. Vertices of a proper k-coloring are randomly selected and reassigned to another color class of the current solution or to a new color class. Moves leading to the reversal of the perturbations should be forbidden and are placed in a one-pass tabu list for the next application of the feasibility search procedure that obtains a proper coloring. The number $maxPerturbations$ of perturbations is handled as a parameter of the algorithm.

4 Computational experiments

Heuristic TSH+PR was implemented in C and compiled with gcc version 5.4.0. An Intel Core i7-4790K processor with a 4 GHz clock and 16 GB of RAM memory running Linux Ubuntu 16.04 LTS 64 bits was

used in the computational experiments. We considered 62 benchmark graphs from http://mat.gsia.cmu.edu/COLOR03 and http://mat.gsia.cmu.edu/COLOR/instances.html [4]. We generated integer random color cost values in the interval $[1, 1000]$ associated with $|V|$ colors for each test instance.

We selected ten instances out of the 62 benchmark graphs with color cost values for tuning the best parameters values for heuristic TSH+PR. Heuristic TSH+PR with $RF = 5\%$, $maxElite = 30$, and $maxPerturbations = \sqrt{V}$ was applied to the remaining 52 benchmark instances. Ten independent runs were performed for each instance, with the stopping criterion being 300 iterations without improvement in the best solution found. Results of this computational experiment show that heuristic TSH+PR matched or improved the best solution found by CPLEX (limited to 3600 seconds) for 43 out of 52 instances and found the optimal solution for six of them, in significantly less time than CPLEX.

Resende and Ribeiro [9, 10] showed that restart strategies are able to reduce the running time to reach a target solution value for many problems. We applied the same type of restart(κ) strategy, in which the elite set is emptied out and the heuristic restarted from scratch after κ consecutive iterations have been performed without improvement in the best solution found. We evaluated the performance of the restart strategy for the minimum cost chromatic partition problem for $\kappa = 50, 100, 200$.

Run time distributions (or time-to-target plots) display on the ordinate axis the probability that an algorithm will find a solution at least as good as a given target value within a given running time, shown on the abscissa axis. Figure 1 displays typical time-to-target plots for instance 4-FullIns_5 (with the target value set to 6334) for TSH+PR without restarts and TSH+PR with the restart(κ) strategy for $\kappa = 50, 100, 200$. This plot makes the case for the use of the restarts strategy, showing that at least one of the improved strategies always finds the target with the same probability systematically faster.

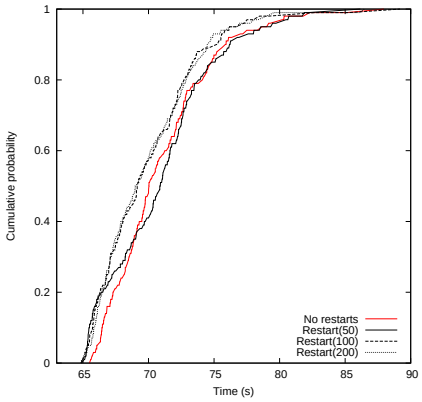


Figure 1: Time-to-target plots for instance 4-FullIns_5 with target value set to 6334.

We also performed long runs for 47 instances for which the value of the best solution found by short runs of TSH+PR without restarts did not match the best value obtained by CPLEX. Two variants of TSH+PR were compared. The strategy without restarts was made to stop after 1000 iterations without improvement in the best solution found. Strategy restart(50) was made to stop after the same running time taken by the strategy without restarts. Given the same running time, strategy restart(50) found the best solution for 34 out of the 47 instances and performed much better than the strategy without restarts, which obtained only 21 best solutions. Table 1 summarizes the results obtained for a subset of 15 among the largest tested instances. The best solution value for each instance is marked in boldface. Full computational results will be presented in a forthcoming paper.

References

[1] M. Chiarandini, I. Dumitrescu, and T. Stützle. Stochastic local search algorithms for the graph colouring problem. In T. F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, Computer & Information Science Series, pages 63.1–63.17. Chapman & Hall, Boca Raton, 2007.

Table 1: Long runs: no restarts and restart(50), one run for each instance. Stopping criterion: 1000 iterations without improvement in the best value for no restarts, same running time as no restarts for restart(50).

			no restarts 1000 iterations without improvement		restart(50) same time as no restarts	
Instance	$\|V\|$	$\|E\|$	Cost	Time (s)	Cost	Time (s)
DSJC1000.5	1000	249826	49412	236.94	**48296**	236.97
flat1000_50_0	1000	245000	43682	384.67	**42562**	384.69
flat1000_60_0	1000	245830	42373	173.72	**41614**	173.73
flat1000_76_0	1000	246708	43325	116.32	**42746**	116.34
DSJC1000.9	1000	449449	111284	476.95	**108604**	477.02
C1000.9	1000	450079	112708	705.85	**110222**	705.87
qg.order40	1600	62400	**15284**	93.80	15291	93.82
wap07	1809	103368	**13687**	177.20	13797	177.21
wap08	1870	104176	14621	595.14	**14487**	595.16
wap02	2464	111742	17625	440.12	**17611**	440.17
qg.order60	3600	212400	**35951**	777.13	35960	777.15
4-FullIns_5	4146	77305	6285	174.78	**6283**	174.81
wap03	4730	286722	20879	2907.46	**20865**	2907.56
wap04	5231	294902	24046	4478.23	**23754**	4478.35
qg.order100	10000	990000	**55601**	4725.44	**55601**	4727.49

[2] A. Helmar and M. Chiarandini. A local search heuristic for chromatic sum. In L. Di Gaspero, A. Schaerf, and T. Stützle, editors, *Proceedings of the 9th Metaheuristics International Conference*, pages 161–170, Udine, 2011.

[3] A. Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, (4):345–351, 1987.

[4] Y. Jin, J.-P. Hamiez, and J.-K. Hao. Algorithms for the minimum sum coloring problem: A review. *Artificial Intelligence Review*, 47:367–394, 2017.

[5] L. G. Kroon, A. Sen, H. Deng, and A. Roy. The optimal cost chromatic partition problem for trees and interval graphs. In F. d'Amore, P. G. Franciosa, and A. Marchetti-Spaccamela, editors, *Graph-Theoretic Concepts in Computer Science*, volume 1197 of *Lecture Notes in Computer Science*, pages 279–292. Springer, Berlin, 1997.

[6] E. Kubicka and A. J. Schwenk. An introduction to chromatic sums. In *Proceedings of the ACM Seventeenth Annual Computer Science Conference*, pages 39–45. ACM Press, 1989.

[7] F. T. Leighton. A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards*, 84:489–506, 1979.

[8] C. Mead and L. Conway. *Introduction to VLSI Systems*. Addison-Wesley, Boston, 1980.

[9] M. G. C. Resende and C. C. Ribeiro. Restart strategies for GRASP with path-relinking heuristics. *Optimization Letters*, 5:467–478, 2011.

[10] M. G. C. Resende and C. C. Ribeiro. *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures*. Springer, Boston, 2016.

[11] A. Sen, H. Deng, and S. Guha. On a graph partition problem with application to VLSI layout. *Information Processing Letters*, 43:87–94, 1992.

[12] K. J. Supowit. Finding a maximum planar subset of a set of nets in a channel. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6:93–94, 1987.

An optimization and pattern mining based approach for solving the RCPSP

Benjamin Dalmas[1], Damien Lamy[2], Arnaud Laurent[3], Vincent Clerc[2]

[1] Mines Saint-Etienne, Centre CIS, F-42023 Saint-Etienne France
benjamin.dalmas@emse.fr

[2] Mines Saint-Etienne, Institut Henri Fayol, F-42023 Saint-Etienne France
damien.lamy@emse.fr; vincent.clerc@emse.fr

[3] Inria Lille, France
arnaud.laurent@inria.fr

Abstract

In this paper, we introduce a pattern-based module to improve the efficiency of local search-based optimizations. The objective of this module is to extract frequent dependencies between tasks from good solutions, then to use them to guide the search phase. We aim at showing that knowledge can be extracted from solutions built to generate either better solutions or to generate similar solutions but faster. The first results obtained tend to validate our hypothesis.

1 Introduction

The Resource-Constrained Project Scheduling Problem (RCPSP) is a problem in which the aim is to schedule a set of tasks using a set of finite but renewable resources of different types; each task having a known processing time and maybe precedence constraints. The RCPSP has been widely addressed in the literature and different methods have been developed to solve it. More specifically, we are interested in knowledge discovery-based metaheuristics. Metaheuristics are very efficient approximation methods to solve the RCPSP. Traditional techniques improve a solution by searching its neighborhood, but without much idea of where exactly in this neighborhood. Therefore, several papers addressed the problem of using information contained in current solutions to guide the neighbors generation [1–3]. However, the knowledge discovery procedure implemented in these techniques are specifically designed, and are not meant to be reusable in other techniques.

The objective of our work is to propose an algorithm-independent knowledge discovery module. As a first step, we focus on local search-based metaheuristics, a class of optimization methods widely used in the literature. The idea is to improve their efficiency regardless of the algorithm used. The work in progress presented in this paper intends to validate the hypothesis according which we can effectively use the information contained in the structure of solutions to perform a smart search. As opposed to existing techniques that use this information to guide the search toward promising areas of the space, our method focuses instead on detecting areas to forbid, which should ensure a better exploration of the search space.

2 The proposed methodology

In the approach we propose, the coding of a solution for the RCPSP is taken from the literature : a solution X is represented by a vector noted σ. It corresponds to a sequence of tasks such that $\sigma=\langle\sigma_1, \sigma_2, ..., \sigma_N\rangle$ is a sequence of N tasks meeting the precedence requirements. Each task is executed in the order defined in σ; i.e., if a task i is before a task j in σ then no resource can execute j before i. To determine a scheduling based on σ, a Schedule Generation Scheme is applied. It schedules each task as soon as possible as given in σ, with respect to the precedence constraints and the resource availability. The goal is to minimize the *makespan*; i.e., the total duration of the scheduled tasks, the makespan of a vector σ being noted mkp(σ). In classical local search, a neighborhood is built by transforming this vector using tasks switches or task insertions, among others.

In this method, we use sequential pattern mining; i.e., an unsupervised learning technique aiming at extracting frequent ordered behaviors, to detect dependencies between tasks is σ. and use them to guide the neighborhood generation. Let *N* be the set of tasks to execute, a sequential pattern $I{=}\langle t_1, t_2, ..., t_N\rangle$, $\forall t_i{\in}s\sigma, m{\leq}N$ is a set of ordered tasks in which the task t_i occurs after the task t_j with $1{\leq}i{<}j{\leq}m$. A vector σ contains a pattern I, noted $I{\in}\sigma$, if the tasks contained in I occur in the same order in σ; for $I{=}\langle t_1, t_2, ..., t_N\rangle, \forall t_i, t_j{\in}I$ with $1{\leq}i{<}j{\leq}m : \sigma_k \prec \sigma_p$ with $\sigma_k{=}t_i$ and $\sigma_p{=}t_j$.

From a set of vectors S, we can measure the support of a pattern, $sup_S(I)$, that is equal to the number of vectors containing the pattern, divided by the total number of vectors. A pattern is frequent if its support is greater or equal to a predefined threshold: $sup_S(I){\geq}minsup$. In this preliminary work, we bound the patterns to two items, $|I|{=}2$, for simplicity reasons.

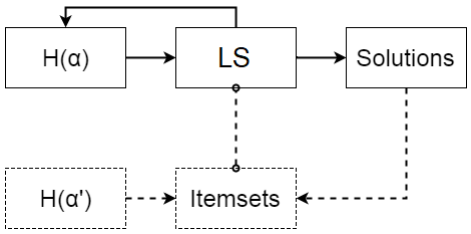


Figure 1: Optimization and pattern mining coupling approach.

To test whether the patterns extracted enables the improvement of the local searches, we propose the experiments shown in Figure 1. The baseline method, presented in solid lines, is a Multistart Local Search - a process of repeated local searches - in which the initial solutions are randomly generated. In this baseline approach, the best solutions generated at each iteration are kept, and the best of these is chosen as the final solution. The main limit here is the fact that what is "learned" during the local searches is forgotten between two iterations. To overcome this limit, we append to this approach our *pattern mining layer*, represented by dotted lines in Figure 1. The principle is simple: at the end of each iteration, the most frequent patterns are extracted from the best solutions. As opposed to existing work that would detect these patterns in order to make them appear in the new solutions generated, our idea is instead to forbid "breaking" these dependencies, because the specific order of the tasks involved in these dependencies might be an indicator of good solutions.

However, as it is, the proposed method is too constraining and prevents some solutions to be reached. To avoid this pitfall, we propose 3 optimizations as follows:

1. ***"Polluting" itemsets removal:*** A frequent pattern should be of interest only if it occurs in good solutions. The goal of this first optimization is to remove patterns occurring more than a predefined threshold both in good and in worst solutions.

2. ***Opposite pattern frequency:*** Even though some patterns are frequent only in the best solutions, it can happen this is the result of "luck"; i.e., the local searches did not change to much this specific behavior or did it but in a bad way. The second optimization intends to, for each frequent pattern and among the best solutions, compare the quality of the vectors containing the pattern and those that do not. If the average makespan of the solutions not containing the pattern is lower than those that do, then the frequent pattern in discarded. Additional constraints state that the difference in the average makespan and the difference between the number of vectors containing a pattern and the number of those that do not have to be higher than predefined thresholds.

3. ***Memory:*** The patterns finally kept are used to guide the next iteration of local searches. Then rises the question of keeping or removing them between two iterations. If removed, then we should end up oscillating between the good patterns of each iteration. The idea of the memory optimization is to keep the patterns found in early iterations in order to keep improving along the process. The

two previous optimizations prevent the memory from remembering too many patterns, so the local searches can reach better new solutions.

The plots presented in Figure 2 shows an example of the promising results of our approach. On small/medium instances, we notice that, compared to phase 0 (initial iteration without any guiding pattern), the following phases reached solutions with a lower average makespan (30 tasks instances in (a)). The local searches reach (i) better solutions and (ii) faster. This validates our hypothesis according which local patterns in the vectors can be indicators of good solutions. However, we face some limits with bigger instances. On 90 tasks instances, the plot (b) shows that after an improvement of the results in the first pattern-based phase, the average makespan increases again, becoming even higher than the initial pattern-free phase. Efforts to understand and overcome this increase has to be made in the following steps of this preliminary work.

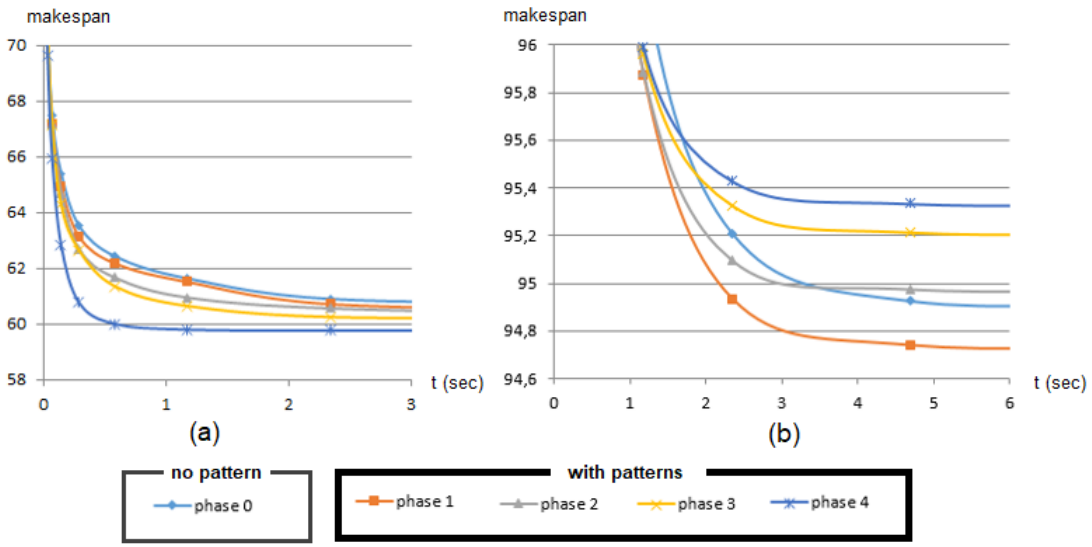


Figure 2: Experiments led on 30 tasks (a) and 90 tasks (b) instances.

3 Conclusion and Perspectives

This paper introduces an approach based on a optimization and pattern mining coupling for solving the RCPSP. From a initial Multistart Local Search, our contribution lies in the addition of an optimized itemset mining layer to guide and improve the generation of good solutions. The first results showed the applicability of our approach. In the following steps of our work, we intend to overcome the decrease in the solutions quality through the different iterations. Mainly, we will focus on a new version of the memory optimization, by introducing a ”conditional memory” that would have the ability to *forget* a good pattern under certain conditions. A second step will be to include this pattern module in a more global approach; a metaheuristic for example.

References

[1] Lin Lin and Mitsuo Gen. Hybrid evolutionary optimisation with learning for production scheduling: state-of-the-art survey on algorithms and applications. *Int. Jour. of Prod. Res.*, pages 193–223, 2018.

[2] Balasundaram R., Baskar N., and Siva Sankar R. Discovering dispathcing rules for job shop schdeuling using data mining. In *Adv. in Comp. and Inf. Tech.*, 2013.

[3] Wang Y., Zhang Y., Yu Y., and Zhang C. Data mining based approach for jobshop scheduling. In *Int. Asia Conf. on Indus. Eng. and Man. Inn.*, 2014.

Adaptative Bacterial Foraging Optimization for Solving the Multi-Mode Resource Constrained Project Scheduling Problem

Luis F. Machado[1], Carlos D. Paternina[1], Agustín Barrios[1]

Universidad del Norte
Km 5, vía Puerto Colombia, Barranquilla, Colombia
lfmachado@uninorte.edu.co, cpaterni@uninorte.edu.co, abarrios@uninorte.edu.co

Abstract

In this paper, a metaheuristic solution procedure for the multi-mode resource-constrained project scheduling problem is proposed. The problem consists of determining a schedule such that the project is completed, maximizing the net present value of the project and complying with the delivery deadline. The adaptative bacterial foraging optimization (ABFO) is a variation of the original bacterial foraging optimization (BFO) which is a nature-inspired metaheuristic optimization algorithm. The ABFO employing the adaptive foraging strategies to improve the performance of the original BFO. This metaheuristic has been tested in two benchmark datasets available in the literature, the PSPLIB and MMLIB with promising results.

1 Introduction

The bacterial foraging optimization algorithm (BFO) created by [5] is inspired by the group behavior of foraging bacteria such as *E.Coli* and *M.xanthus*. That is, it imitates how bacteria feed in a landscape of nutrients to perform a non-aggressive parallel optimization, perceiving chemical gradients in the environment (such as nutrients) and moving towards or away from specific signals. Chen et al. [1] proposes an adaptive bacterial foraging algorithm based on artificial bacteria that are capable of self-adapting their exploration and exploitation behaviors in the foraging process. The proposal to use a crossing technique of the genetic algorithm [4] is to improve the evaluation of the objective function. The mutation will be applied to change the structure and position of the bacteria with reference to the nutrient-rich areas and to reintroduce the genetic material lost in the colony and create a variation in the bacteria. The multi-mode resource-constrained project scheduling problem (MRCPSP), which emerges is an important practical problem in project management and combinatorial optimization. The MRCPSP is NP-hard and, if there is more than one nonrenewable resource, the problem of finding a feasible solution for the MRCPSP is NP-complete [2]. The classical MRCPSP is the problem of selecting an execution mode and the assignment of an initialization or completion time for each activity under resource and precedence restrictions with the objective of minimizing the makespan. The variants of the MRCPSP with respect to the measure performance (see [7]) are (1) to maximize the net present value (NPV) of the project, (2) to minimize the resource availability cost, or (3) to minimize the project cost. Here we propose an ABFO to resolve the MRCPSP maximizing the NPV of the project, defined as the sum between discounted cash inflows and outflows, with performance measure the maximize . There are several metaheuristic procedures to solve the MRCPSP with different objective functions [7]. However, the use of metaheuristic strategies based on the behavior of bacteria for the MRCPSP is null.

2 Adaptive bacterial foraging optimization

Algorithm 1 provides a overview of the adaptive bacterial foraging optimization for maximizing a NPV function, with payments due at activitie's completion times (PAC), [7]. In the execution of the ABFO, an initial colonial bacterium of size $nCol$ is initially constructed, each bacterium is represented by a pair (λ, μ). Here, λ the list of activities, a permutation of activities numbers with considerations for precedence relations and μ is the mode vector. The value of the objective function is computed

as $Npv(\lambda, \mu) = \sum_{i=1}^{n}(c_{i,in} + c_{i,out}) \cdot e^{-\alpha f_i}$, with $c_{i,in}(> 0), c_{i,out}(< 0)$ cash in- and outflow, respectively. The discount rate is α and f_i is completion time of the activity i. The chemotactic step $ChemotaxisAndSwim$ is based on the forward/backward improvement method and mode change. Each activity i is selected from the list λ. Then, the interval $[ES_i, LS_i]$ is calculated for each mode. Next, if $ES_i \leq LS_i$, the time where the i activity could be sequenced is calculated. If $c_{i,in} \geq c_{i,out}$, the activity is sequenced to the left, and to the right otherwise. In the case of swarming, we select the bacteria with better fitness which releases attractants and repellents to send signals to other bacteria so that they are grouped, provided that they obtain an environment rich in nutrients or avoid harmful environments. A copy of the best solution, $(Bact_{best})$, is subsequently created. The bacteria are then ordered in descending order ($SortByCallHealth$) according to whether NPV is maximized. Bacteria are treated as parental bacteria in order to select two groups of progenitor bacteria and thus generate two offsprings, using a uniform crossing operation $(Crossover)$. Bacteria with the highest fitness are selected for the next generation. Given a probability $(Mutation)$, the bacteria can make a change in their structure and position with respect to the environment; this behavior can either improve or worsen bacteria's quality. Bacteria affected by the spontaneous mutation may return to the structure or previous position, in order to leave environments with few nutrients. Due to the gradual or sudden change in the local environment, the life of the bacteria may be affected. Then, to incorporate this phenomenon, we use the $(CreateBactAtRandomLocation)$ step to eliminate each bacterium in the population with certain probability and randomly initialize a new replacement to in the search space. The result is a bacterium with objective function optimized.

Algorithm 1: PSEUDOCODE FOR THE BFOA.

Input: $n, S, N_c, N_s, N_{re}, N_{ed}, P_{ed}, P_{cross}, P_{mutE}, P_{mutR}$
Output: $Bact_{best}$
$Col \leftarrow InicializarColonia(S, n)$
for $l = 1$ **to** N_{ed} **do**
  for $k = 1$ **to** N_{re} **do**
    for $j = 1$ **to** N_c **do**
      $Col \leftarrow ChemotaxisAndSwim(Col, S, N_s)$
      foreach $Bact \in Col$ **do**
        if $NPV_{Bact} \geq NPV_{Bact_{best}}$ **then**
          $Bact_{best} \leftarrow Bact$
    $Col \leftarrow SortByCellHealth(Col)$
    $Crossover(Col, P_{cross})$
    $Mutation(Col, P_{mutE}, P_{mutR})$
    $Col \leftarrow SelectByCellHealth(Col, S/2)$
  foreach $Bact \in Col$ **do**
    if $Rand() \leq P_{ed}$ **then**
      $Bact \leftarrow CreateBactAtRandomLocation()$
return $Bact_{best}$

3 Computational experiment

In this section we evaluate the performance of our ABFO algorithm under several instances available in the literature. The ABFO has been coded and compiled in C++ and used in the computational test on a Toshiba laptop computer with an Intel Core 2.3GHz processor and 4GB of RAM. We calculated the average deviation considering 5000 schedules with fixed parameters of the J10, J12, J14, J16, J18, J20 and J30 sets of the PSPLIB library [3], and a set of benchmark instances MMLIB composed for MMLIB50, MMILB100 and MMLIB+, [6]. Overall, the execution time of the algorithm is acceptable and the quality of the sequences is quite good. Table 1 summarizes the results of the experiments. The contents of the table include of each set of instance, the percentage deviation from the best found solution

reported, and the average NPV for the project $(AvNpv)$ of all feasible solutions with an increase of 5%, 10%, 15% and 20% for the deadline.

Set Instance	5000 schedules	$AvNpv$ with $D - Incre$			
	Av.Dev $(C_{\max})(\%)$	5%	10%	15%	20%
J10	0.0	420.58	460.87	473.12	433.14
J12	0.01	243.41	256.18	270.54	301.89
J14	0.03	320.74	328.62	332.34	351.47
J16	0.07	270.6	379.75	478.13	494.12
J18	0.1	159.62	462.37	475.35	483.12
J20	0.4	309.22	468.45	486.32	508.06
J30	13.5	345	680.65	701.33	745.19
MMLIB50	30.42	850.12	855.75	876.85	906.25
MMLIB100	43.05	3105.87	3275.61	3030.78	3540.87
MMLIB+	105.25	3560.98	3621.06	4687.42	5470.68

Table 1: Results for the instances of library PSPLIB and MMLIB executing ABFO.

4 Conclusions

In this work, we discussed the the multi-mode resource-constrained project scheduling problem considering the project's net present as the objective function. The interaction between the execution time and cash inflows or outflows is dynamic. Finding a balance between time/cost has been a priority for project managers. For this, it is necessary to have tools that help to make decisions. Here we proposed a bacterial adaptative foraging metaheuristic optimization for such purpose. In this algorithm, we have used operators that explore and exploit the solution space (that is, forward/backward improvement method, mode change and priority rule) to consider the positive and negative cash flow of the activities. The algorithm is tested on two benchmark datasets available in literature, namely the PSPLIB and MMLIB. Althought our proposal performs reasonably well in these instances, assessing its performance when other objective functions are used, and a possible extension for multi-objective functions are a work in progress.

References

[1] H. Chen, Y. Zhu, and K. Hu. Adaptive bacterial foraging optimization. In *Abstract and Applied Analysis*, volume 2011. Hindawi, 2011.

[2] R. Kolisch and A. Drexl. Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 29(11):987–999, 1997.

[3] R. Kolisch and A. Sprecher. Psplib-a project scheduling problem library. *European Journal of Operational Research*, 96(1):205 – 216, 1996.

[4] R. Panda and M. K. Naik. A crossover bacterial foraging optimization algorithm. *Applied Computational Intelligence and Soft Computing*, 2012:23, 2012.

[5] K. M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE control systems*, 22(3):52–67, 2002.

[6] V. Van Peteghem and M. Vanhoucke. An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research*, 235(1):62–72, 2014.

[7] J. Weglarz, J. Józefowska, M. Mika, and G. Waligóra. Project scheduling with finite or infinite number of activity processing modes: A survey. *European Journal of Operational Research*, 208(3):177 – 205, 2011.

A Harmony Search Approach for A Dynamic Cell Formation Problem

Laura Y. Escobar Rodríguez [1], Edwin A. Garavito Hernández[2], Leonardo H. Talero Sarmiento[3]

[1] Universidad Industrial de Santander
Carrera 27 con calle 9, Ciudad Universitaria, Bucaramanga, Colombia
laura.escobar@correo.uis.edu.co

[2] Universidad Industrial de Santander
Carrera 27 con calle 9, Ciudad Universitaria, Bucaramanga, Colombia
garavito@ uis.edu.co

[3] Universidad Autónoma de Bucaramanga
Avenida 42 No 48 - 11 Altos de Cabecera, Bucaramanga, Colombia
leonardo.talero@gmail.com

Abstract

This document addresses a harmony search approach for a dynamic cell formation problem presented as a mixed-integer linear model aiming to minimize the total costs associated with the production process. The mathematical model takes into account changing demand and part mix, machine relocation and machine sequence.

1 Introduction

The design of cellular manufacturing systems in a planning horizon has been called dynamic cell formation problem (DCFP). Given a set of part types, processing requirements, part type demand and available resources, the decision maker looks to form part families according to their processing requirements, then, group machines into manufacturing cells and finally assign part families to cells [1]. This general design is made for every period in a planning horizon, given place to machine location or system reconfiguration (Figure 1).

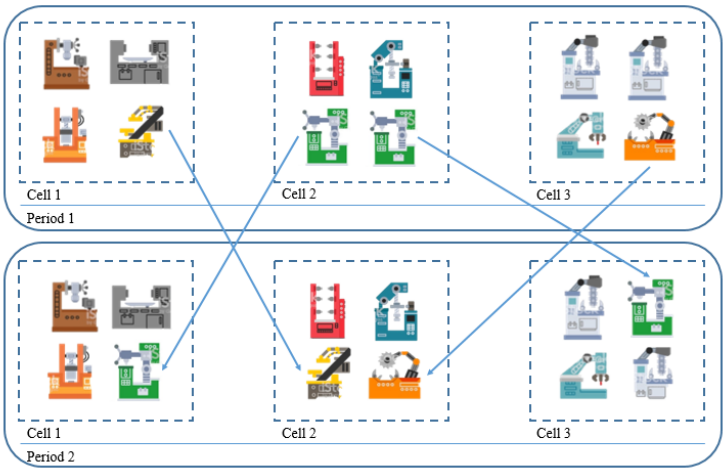


Figure 1: An illustration of machine relocation in a dynamic cellular manufacturing system

The objective of the DCFP is to minimize the total cost of acquisition, disposal, and relocation of machines, manufacturing, and inter-cell/intra-cell material handling. In this work we developed a harmony search algorithm which is presented by Zong, Kim & Loganathan [2], this algorithm is characterized by

intelligently manipulating and combining mechanisms for an adequate process of exploration and exploitation of a search space. This metaheuristic has been used in different disciplines, Cobos & Mendoza [3] presents a clustering algorithm based on the combination of the harmony search and K-means algorithms. Jaberipour & Khorram [4] proposed a harmony search algorithm for solving a sum-of-ratios problem with some applications in economy and engineering. Li, Li & Gupta [5] presented a hybrid harmony search to solve a multi-objective flowline manufacturing cell scheduling problem.

2 Proposed Harmony Search Algorithm for the Model

2.1 Problem Description

This work considers the cell formation problem in a dynamic environment with changing demand and part mix, machine relocation and machine sequence. The DCFP is presented as a mixed-integer linear model aiming to minimize the total costs associated with the production process, taking into account the following assumptions: the demanded quantity of each part type in each period is known, and production can meet demand; the capacity of each machine type is known and constant over the planning horizon; the cost of acquisition, disposal, and relocation of machines are known and constant over the planning horizon; the operation sequence of each part type is known; the unit inter-cell material handling cost is known and constant over the planning horizon.

2.2 The encoding of a solution

In the DCFP considered in this work, the encoding of the solution represents the assignment of the part types of every operation required to the machines in each cell in each period. The encoding of a solution for the above problem consists of J operations required, I part types, M machines, K cells and T periods from a planning horizon. Figure 2 illustrates the encoded solution which is a vector with $[I * J * M * K * T]$ elements.

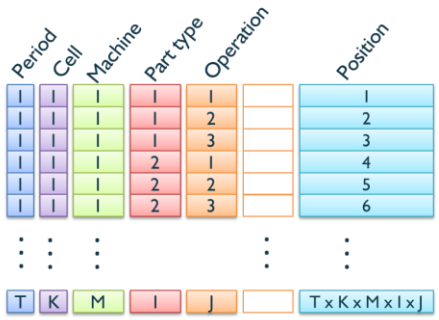


Figure 2: The encoding of a solution

2.3 Harmony search structure

The HS model can be presented as the solution that optimizes a vector which is constrained between upper and lower limits:

$$min\ (\ max)\ f(x)\ |\ LB_i < x_i < UB_i \qquad (1)$$

Where: $f(x)$ represents the objective function; x: N-length solution vector and (x_i) decision variables; LB_i, UB_i are the lower and upper limits of each variable.

2.4 Parameters

The values of the parameters are chosen from Garavito Hernández [6]. The Harmony Memory Size,

(HMS) represents the amount of sored solutions $\{100, 1000\}$. The Harmony Memory Consideration Rate ($HMCR$) is the probability of choosing a stored solution to be harmonized $\{50\%, 90\%\}$. The Pitch Adjusting Rate (PAR) is related to the probability of randomly generating the initial solution $\{10\%, 50\%\}$. The Maximum Number of Searches ($MAXIMPROV$) represents the termination criterion. $\{100,5000\}$.

Taking into account the search space diversification strategies proposed by Gao, Wang, Ovaska, & Zenger [7], we present another parameter called Number of Sub-harmonies (SUBARMO) with the purpose of generating multiple harmonic vectors $\{1,50\}$.

2.5 Solution improvement procedure

The optimization procedure of the HS meta-heuristic algorithm starts with the initialization of the optimization problem and the algorithm parameters. Then, the Harmony Memory (HM) is initialized and the initial harmony (solution vector) is generated (as many as HMS). The improvisation process of a new harmony from the HM is based on natural musical performance processes that happen when a musician searches for a better state of harmony. When a musician improvises one pitch, usually he follows one of three rules: (1) playing any one pitch from his memory, (2) playing an adjacent pitch of one pitch from his memory, and (3) playing totally random pitch form the possible sound range. Similarly, when each decision variable chooses one value in the HS algorithm, it follows any of three rules: (1) choosing any one value from the HS memory (defined as memory considerations, HMCR), (2) choosing an adjacent value of one value from the HS memory (defined as pitch adjustments, PAR), and (3) choosing totally random value from the possible value range (defined as randomization)[8]. When each decision variable chooses one value in de HS algorithm, it can apply one of the abovementioned rules. If the new vector in the harmony memory is better than the worst harmony vector in the HM, the new harmony vector will replace it. The procedure is repeated until the stopping criterion is satisfied (MAXIMPROV) [9].

References

[1] G. Papaioannou and J. M. Wilson, “The evolution of cell formation problem methodologies based on recent studies (1997-2008): Review and directions for future research,” *Eur. J. Oper. Res.*, vol. 206, no. 3, pp. 509–521, 2010.

[2] Z. W. Geem, J. H. Kim, and G. V. Loganathan, “A New Heuristic Optimization Algorithm: Harmony Search,” *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[3] C. Cobos and M. Mendoza, “A harmony search algorithm for clustering with feature selection Un algoritmo de búsqueda armónica para clustering con selección de características,” pp. 153–164, 2010.

[4] M. Jaberipour and E. Khorram, “Journal of Computational and Applied Solving the sum-of-ratios problems by a harmony search algorithm,” *J. Comput. Appl. Math.*, vol. 234, no. 3, pp. 733–742, 2010.

[5] Y. Li, X. Li, and J. N. D. Gupta, “Expert Systems with Applications Solving the multi-objective flowline manufacturing cell scheduling problem by hybrid harmony search,” *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1409–1417, 2015.

[6] E. A. Garavito Hernandez, L. H. Talero Sarmiento, and L. Y. Escobar Rodriguez, “Aplicación de meta heurísticas para solucionar el problema de formación de celdas de manufactura: una revisión,” in *II Congreso Colombiano de Investigación Operativa*, 2017.

[7] X. Z. Gao, X. Wang, S. J. Ovaska, and K. Zenger, “A hybrid optimization method of harmony search and opposition-based learning,” *Eng. Optim.*, vol. 44, no. 8, pp. 895–914, 2012.

[8] K. S. Lee, “A new meta-heuristic algorithm for continuous engineering optimization : harmony search theory and practice,” vol. 194, pp. 3902–3933, 2005.

[9] O. Abdel-raouf, “A Survey of Harmony Search Algorithm,” *Int. J. Comput. Appl.*, vol. 70, no. 28, pp. 17–26, 2013.

Complex Production Scheduling: case studies from various industries

Marco Better[1] and Fred Glover[2]

[1]OptTek Systems, Inc.
2241 Seventeenth Street, Boulder, CO 80302, USA
better@opttek.com

[2]ECEE, College of Engineering & Applied Science
University of Colorado, Boulder, CO 80309, USA
fred.glover@colorado.edu

Abstract

Manufacturing companies often involve processes that exhibit high levels of complexity and operate in an environment where optimal production schedules can provide a source of competitive advantage. Features commonly shared by these companies include: (1) production costs represent a significant portion of total product price, (2) multiple products share manufacturing infrastructure and resources, and (3) production schedules are required on a timely basis. We have developed a sophisticated production scheduling solution approach that combines mathematical programming, metaheuristic optimization, and simulation to craft optimal or near-optimal production schedules that perform in a reliable and effective manner. In this paper, we describe our approach and provide examples and computational results in a variety of industry case studies.

1 Introduction

Manufacturing and packaging companies typically face situations where production costs represent a significant portion of the price of their products, creating an environment in which optimal production schedules can provide a source of competitive advantage. For these companies, multiple products often share common manufacturing infrastructure and resources, and production schedules are required on a timely basis.

The production planning and production scheduling academic literature is quite vast and includes many simulation, pure optimization and hybrid optimization-based approaches to food production scheduling. Recent surveys (Jahangirian, et al. 2010) and (Smith 2003) identify more than 20 implementations involving discrete event simulation approaches, with more than a dozen others involving alternative types of simulation. Numerous other implementations involve mixed integer programming formulations (Wari and Zhu 2016) and metaheuristics (Abakarov, et al. 2009). However, these implementations characteristically make severe simplifications of the processes they are trying to represent, consider only portions of a complete process, or do not provide an integrated system capacity and job sequencing framework.

Existing solution approaches for production process scheduling commonly focus on two basic questions:

- When should a specific job be scheduled?
- What resources should be assigned to perform the job?

In many cases, companies attempt to answer these questions by applying simple, rule-based heuristics, such as sequencing tasks by earliest due date or by the magnitude of their processing times. More

complex rules can be derived by combining two or more simple rules into ratios or products, but the basic concept remains the same. Although appealing for their simplicity and intuitive nature, these methods usually produce inferior results because they tend to ignore critical task attributes such as penalties for tardiness, interactions with other tasks, availability of resources to perform all the work, changeover and setup times and costs, etc. To address these limitations, optimization-based approaches can be used. These methods use mathematical programming techniques to find the optimal solution to maximize or minimize some metric, like throughput, capacity utilization, makespan, or operating cost.

The complexity of most real-world systems often involves several decisions, including:

- How to size a task
- How to assign a task to a production line
- How to sequence the tasks on each production line

In high product mix environments, where product changeovers are sequence-dependent, the application of exact optimization methods is impractical (Lee, Bhaskaran and Pinedo 1997), either because the time to obtain the optimal solution is excessive, or because such systems are too complex to be mathematically formulated. In these cases, what is needed is a combination of mathematical methods combined with metaheuristic solution techniques and, often, simulation modeling approaches.

In this paper, we will describe a sophisticated production scheduling solution approach that combines mathematical programming, metaheuristic optimization, and simulation to craft optimal or near-optimal production schedules in a timely, reliable and effective manner. We will then demonstrate our approach on a variety of production environments, from food processing to construction materials to ecommerce.

Our approach is especially suited to very complex situations, with a high production volume and a high product mix, where sequence dependent constraints, multiple line assignments, scarce resources, and tight storage and work-in-process constraints may be present. In addition, this approach is very effective when a major disruption occurs in the plant, and there is an urgent need to quickly reoptimize the production schedule.

2 Methodology

Our production scheduling approach makes use of the appropriate technologies, either alone or in combination, depending on the situation. What every implementation has in common, however, is the technological framework. The framework can be described as a scheduling optimization engine, which makes use of a diverse set of techniques to obtain an optimal or near-optimal production schedule. These techniques include mathematical programming, metaheuristic optimization, and the combination of simulation and optimization. Figure 1 shows a high-level representation of the technology.

Even if highly complex, certain production environments are tractable enough that they can be solved with the Optimization Engine (OE) alone. Examples are operations where production lines are independent of each other (i.e., there is very little interaction between production lines), and resources (such as labor and physical assets) are dedicated to each production line. Most real-world systems, however, exhibit such complexity and interactivity that make it impossible to describe them with a set of equations or mathematical expressions. One instance of such a system is a dairy foods production facility. It may be relatively simple to optimize the schedule of the filler machines and the packaging facility, but such a schedule may conflict with the upstream operations in the process. Good coordination between processing and packaging is critical, so that the scheduling of the pasteurization step and the sterilization step, for example, are well-synchronized with the scheduling of the filling and

packaging operations. In these cases, we make use of a simulation model which we call the Schedule Evaluation Model (SEM). The SEM is a realistic model of the production facility, which allows us to iteratively evaluate schedules suggested by the optimization engine in a simulation optimization sense, as shown in Figure 1.

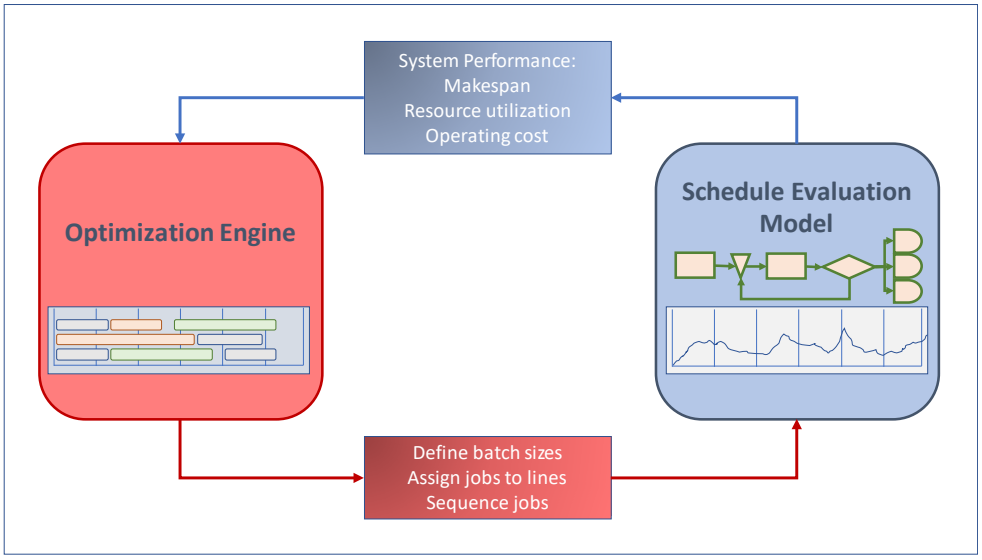


Figure 1: high-level representation of the technology behind OptPro

Our approach is not designed for situations where using straight-forward rules such as FIFO or EDD would suffice. It is for those situations where

- multiple products compete for common resources, such as production infrastructure and materials;
- a good production schedule can turn into a competitive advantage;
- significant gains can result by optimizing and automating a company's plant design and production schedule, and maximizing the benefit derived from operational processing decisions.

In manufacturing settings, the approach enables optimal decision making by simultaneously optimizing scheduling, sequencing, line-assignment, capacity and layout decisions to meet forecasted customer demands. We demonstrate the benefits of our proposed approach on a set of case studies based on real-world examples.

3 Case studies

We showcase our methodology on implementations in three different industries: (1) construction materials, with an example from a pipe insulation production facility; (2) a high-volume ecommerce operation; and (3) a liquid foods manufacturing facility. In all cases, we begin by formulating the production scheduling instance as a mixed integer programming problem that discloses its essential features. We then describe our solution approach, and report on results from our implementation. Finally, we identify the benefits our approach has afforded the company in its real-world setting, compared to the scheduling approach that had been in place prior to our involvement.

Revisiting the Set k-Covering Problem

Thiago A. Virgilio, Anderson S. Ramos, Amanda A. B. da Silva, Luciana S. Pessoa

Department of Industrial Engineering, PUC-Rio,
Rua Marquês de São Vicente, 225, Gávea - 22453-900 Rio de Janeiro, RJ, Brazil.
thiago.a.virgilio@hotmail.com, anderson.ramos@engenharia.ufjf.br, a.mandabatista@hotmail.com,
lucianapessoa@puc-rio.br

Abstract

The set k-covering problem is an extension of the well studied set covering problem in which, given a binary matrix 0-1 with costs linked to the columns, each row is required to be covered at least k times, by choosing a minimum cost subset of columns. Since the development of the first metaheuristic for this problem, in 2011, some progress has been reached for solving it and harder instances have been proposed. This short work-in-progress aims to bring a new application of the Iterated Local Search (ILS) metaheuristics to explore the set k-covering problem. Some experiments were carried out to show the direction to be followed in this research.

1 Introduction

Given a binary matrix $A = [a_{ij}]$, where a row $i \in I = \{1, \ldots, m\}$ and a column $j \in J = \{1, \ldots, n\}$, a column j covers a row i, if $a_{ij} = 1$. Additionally, each column j has a cost c_j associated to it. Equations (1) to (3) define an integer programming formulation for the set k-covering problem. The SC_kP is a NP-hard problem whose solution consists in finding a minimum-cost subset S of columns of matrix A, such that each row of the latter is covered by at least k columns in S [6].

$$\min \sum_{j=1}^{n} c_j x_j \tag{1}$$

subject to:

$$\sum_{j=1}^{n} a_{ij} x_j \geq k, \quad \text{i=1,...,m} \tag{2}$$

$$x_j \in \{0, 1\}, \quad \text{j=1,...,n} \tag{3}$$

where x_j indicates whether column j belongs to the solution (x_j = 1) or not (x_j = 0).

This problem was tackled initially by Pessoa et al. [4, 5] with the proposition of a Greedy Randomized Adaptive Search Procedure (GRASP) hybridized with path-relinking, as well as with a Lagrangean heuristic. A benchmark of 135 instances, with up to 400 rows and 4000 columns, were generated by deriving classical instances of the set covering problem [2]. Al-Shihabi [1] developed a hybrid Ant Colony-based algorithm that performs a linear relaxation of the set k-covering problem to classify the subsets based on reduced costs, aiming to restrict and guide the building of feasible solutions. Local searches are applied on the obtained solutions. Wang et al. [8] proposed a local search heuristic, called DLL_{CCSM} (Diversion Local Search based on Configuration Checking and Scoring Mechanism). This method is based on a strategy of neighborhood configuration analysis of each subset in each potential solution. Besides, a scoring mechanism directs the addition or removal of a subset at each iteration of the algorithm. Recently, Wang et al.[7] improved the subset configuration analysis strategy and the scoring mechanism present in the DLL_{CCSM} heuristic by proposing the MLQCC search algorithm. The updated method is composed by the scoring procedure ML (multilevel) and the configuration strategy QCC (Quantitative Configuration Checking). The authors also presented a pre-processing step applied to reduce the search space of the problem, based on the possibility that some rows could be covered by only k columns. The experiments presented by Wang et al. [8] and Wang et al. [7] were carried out on instances with maximal dimensions of 1000 rows and 10000 columns, in addition to those proposed by Pessoa et

al. [4], recording important advances in solving the set k-covering problem. The results obtained by the heuristics were compared with those found by running a mathematical model on a commercial solver.

2 Proposed Method

The proposed algorithm for solving the set k-covering problem is based on the Iterated Local Search (ILS)[3]. This is a two phase method where, initially a feasible solution is built by a greedy constructive procedure followed by a local search method that attempts to improve the incumbent solution. Then, this locally optimum solution is given to an iterative process of perturbation-local search-acceptance criterion.

The ILS customized for the set k-covering problem builds an initial solution by sorting the columns in the increasing order of their costs and, subsequently, inserting the smallest cost columns, one at a time, until all rows are k covered, i.e, until the subset of columns belonging to the solution is able to cover each row, at least, k times. Next, a local search composed by the combination of the $1,0$-exchange and $1,1$-exchange neighborhoods is applied. The $1,0$-exchange neighborhood search attempts to the remove redundant columns. On the other hand, the 1,1-exchange neighborhood replaces a column in the solution by a lower cost column which is not in the solution.

Within the cycle of perturbation-local search-acceptance criterion, the perturbation step is performed by removing a certain number of randomly chosen columns from the solution (which leads to an unfeasible solution), followed by the iterative process that inserts columns in the solution until it reaches the feasibility again. The number of columns removed, which identifies the strength of the perturbation, is calculated from a perturbation factor that multiplies the total number of columns belonging to the solution. The local search is the same applied on the initial solution, and the acceptance criterion only considers improving solutions.

3 Computational Results

In order to evaluate the performance of the proposed structure, experiments were carried out on some of the instances proposed by Pessoa et al. [4], who also defined three levels of coverage factors (k factor) required for each instance, as follows: $\mathrm{k}_{min} : k = 2$; $\mathrm{k}_{max} : k = \min \sum_{j=1}^{n} a_{ij}, i = 1, ..., m$; and, $\mathrm{k}_{med} : k = (k_{max} + k_{min})/2$.

For this study, seven test instances were selected from the OR-Library [2]. Each one with a different combination of the numbers of rows and columns, varying from 200 to 400 rows, and from 1000 to 4000 columns, as well as its density. Readers are referred to the work of Beasley [2], in order to obtain information about the instances scp41, scp51, scp61, scpa1, scpb1, scpc1 and scpd1. On each instance, the three above-described coverage factors were applied, so that a total of 21 test instances were considered in the experiment.

The computational experiments were performed on a Intel Core i7-8550 U CPU @ 1.80GHz running Linux Ubuntu 18.04. Each run was limited to a single processor. All codes were implemented in C++. The results were compared to the best known solutions (BKS) reported in [7]. In order to ensure a fair comparison, the time limits for the execution of the proposed method on each instance were adjusted to the used computer according to the time limits described by Wang et al. [7].

Given the importance of the perturbation strength, the aim of this experiment is to evaluate different values for such parameter. To evaluate each parameter value, eight runs were carried out for each instance, giving different initial seeds to the random number generator. The study focused on four perturbation factors, as showed in Table 1. First column brings the k_factor, each one encompassing seven out of the 21 instances. Then, below each perturbation factor are the %ADev and %BDev comparative metrics. %ADev refers to the average (over all instances within that coverage factor) percentage deviation from the average (over the eight runs) of the solution costs to the BKS. On the other hand, %BDev refers to the average (over all instances within that coverage factor) percentage deviation from the best (over the eight runs) solution costs to the BKS. Due to the increasing in the solutions quality observed

when perturbation factors smaller than 0.05 were applied, a strategy of a variable perturbation factor was examined. In the *Rand* strategy, at each iteration of the ILS, a perturbation factor is randomly chosen from the interval [0..0.05]. The benefits of applying the random approach can be observed, particularly, in the line *Total* which consolidates the results over the 21 instances. Although, this good performance is not evident for all coverage factors, this idea is worth further investigation.

Table 1: Results obtained with different perturbation factors for 21 instances, grouped by coverage factor

	Perturbation factor							
	0.05		0.03		0.01		Rand	
k_factor	%ADev	%BDev	%ADev	%BDev	%ADev	%BDev	%ADev	%BDev
k_{min}	3.42	2.11	2.24	0.99	4.82	4.28	2.52	1.23
k_{med}	6.28	5.74	6.87	6.37	4.82	4.31	5.33	4.75
k_{max}	5.85	5.47	5.44	4.95	4.79	4.43	4.83	4.25
Average	5.18	4.44	4.85	4.10	4.81	4.34	4.23	3.41

4 Next steps

The focus of this work was to investigate the application of an ILS metaheuristic for set k-covering problem. The results showed that the proposed algorithm was able to lead to solutions whose average deviation varies from less than 1.0% to about 6.5%, depending on the perturbation and the coverage factor. Besides, the strategy of using a randomly chosen perturbation factor showed promising results.

In the near future, greater emphasis will be given to the ILS self-tunning, by extending the experiments on the *Rand* strategy. Additionally, the components of the proposed method should be improved in order to reach a better performance when tackling the classical instances of the set k-covering problem.

References

[1] Sameh Al-Shihabi. A hybrid of max–min ant system and linear programming for the k-covering problem. *Computers & Operations Research*, 76:1–11, 2016.

[2] John E Beasley. OR-Library: distributing test problems by electronic mail. *Journal of the operational research society*, 41(11):1069–1072, 1990.

[3] Helena R Lourenço, Olivier C. Martin, and Thomas Stützle. *Iterated Local Search: Framework and Applications*, pages 129–168. Springer International Publishing, Cham, 2019.

[4] Luciana S Pessoa, Mauricio GC Resende, and Celso C Ribeiro. Experiments with lagrasp heuristic for set k-covering. *Optimization Letters*, 5(3):407–419, 2011.

[5] Luciana S Pessoa, Mauricio GC Resende, and Celso C Ribeiro. A hybrid lagrangean heuristic with grasp and path-relinking for set k-covering. *Computers & Operations Research*, 40(12):3132–3146, 2013.

[6] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[7] Yiyuan Wang, Chenxi Li, Huanyao Sun, Jiejiang Chen, and Minghao Yin. MLQCC: an improved local search algorithm for the set k-covering problem. *International Transactions in Operational Research*, 26(3):856–887, 2019.

[8] Yiyuan Wang, Minghao Yin, Dantong Ouyang, and Liming Zhang. A novel local search algorithm with configuration checking and scoring mechanism for the set k-covering problem. *International Transactions in Operational Research*, 24(6):1463–1485, 2017.

A sim-heuristic approach for the 3D irregular packing problem

Germán F. Pantoja[1], David Álvarez-Martínez[2]

[1] Universidad de los Andes
Carrera 1 Este # 19 A – 40, Bogotá D.C., Colombia
gf.pantoja10@uniandes.edu.co

[2] Universidad de los Andes
Carrera 1 Este # 19 A – 40, Bogotá D.C., Colombia
d.alvarezm@uniandes.edu.co

Extended Abstract

The 3D irregular packing problems are part of the combinatorial optimization problems (COP) [1], which have a high mathematical and computational complexity. In addition, these problems have a wide spectrum of applications in the industry where their solutions require to be of high quality and to be obtained in short computational times [2].

The problem to tackle in this work is to minimize the volume of a cuboid in which three-dimensional concave polyhedrons of different types are packed, and free rotation of the pieces is allowed. In the typology proposed by Wäsher et al. [3], this problem can be classified as:

- Dimensionality: three-dimensional.
- Kind of assignment: input minimization.
- Assortment of small items: this criterion may vary from weakly heterogeneous assortment to strongly heterogeneous assortment, depending on the nature of the instances.
- Assortment of large objects: one large object.
- Shape of small items: irregular.

To solve the COP, several methodologies have been proposed that usually involve heuristics, since efficient solutions are obtained with them. Lately, combinations of various techniques have received great popularity especially the mixture of heuristics and simulation called sim-heuristics, due to the increase in computational power and the development of simulation software.

In this work, a tool based on a sim-heuristic is developed using Unity®, a video game engine. Unity® incorporates colliders, a tool that can verify the overlapping between pieces and between piece-cuboid. Besides, Unity® allows programming heuristics based on movements, collisions, and forces that provide a convincing physical behavior of the elements (pieces and cuboid) due to the incorporated physics engines.

The approach of this work is to perform simulations of 3D irregular packing of pieces that will be subdued under forces (movements) and collisions that will change their position. When the positions of the pieces allow the cuboid to be smaller, the faces of the cuboid will move towards the pieces to compress the volume of the cuboid.

The proposed method considers three sections. First, the pieces are enclosed in a sphere using the algorithm presented in [6], this is done in order to facilitate their initial ubication inside the cuboid based on the biggest sphere. Second, the pieces are subdued under an increased gravity force and movements of the box that induce movement of the pieces within the cuboid. Third, the faces of the cuboid move towards the pieces in order to decrease the volume of the cuboid. The latter are framed in a Tabu Search, in such a way that the simulation does not return to already visited states [7]. Therefore, the transition mechanisms are the simulated accelerations and the search strategy is a simple Tabu Search algorithm.

In order to compare the performance of this approach with the results available in the literature, the instances used in this work are classical instances of concave pieces such as those found in [4], [1] and

[5]. The Figures 1 and 2 show the pieces of an instance of the problem and the proposed method, respectively.

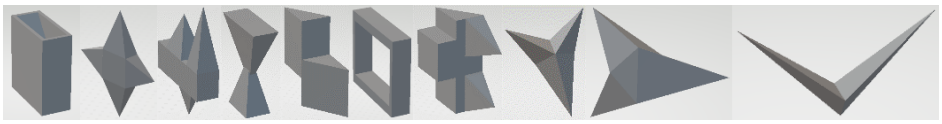
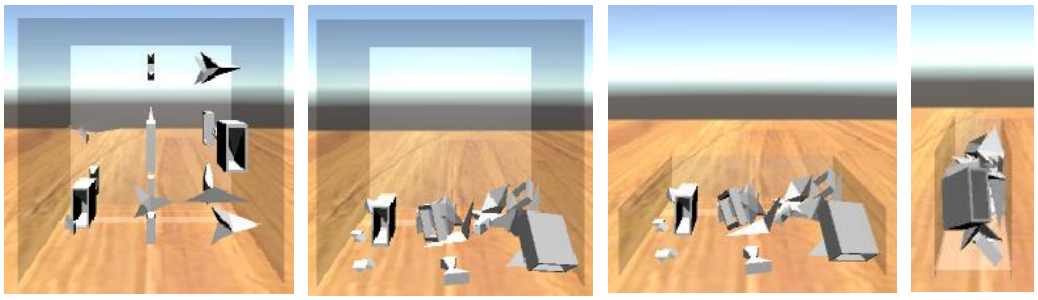


Figure 1: Type of pieces used in this work [5]



(a) (b) (c) (d)

Figure 2: Example of the proposed sim-heuristic for the instance 20 [5].

(a) shows the initial position of the pieces using the spheres of each piece.
(b) shows the pieces subdued under the gravity force.
(c) shows the movement of the top face of the box towards the pieces decreasing the volume.
(d) shows the final result of the sim-heuristic obtained after moving the box in several ways and moving the faces of the box towards the pieces to decrease the volume of the box.

Key words: Irregular 3D packing, sim-heuristic, concave pieces, Unity®.

References

[1] C. Liu, J.-m. Liu, A.-x. Cao & Z.-l. Yao, HAPE3D - a new constructive algorithm for the 3D irregular packing problem, *Frontiers of Information Technology & Electronic Engineering*, pages 380-390, 2015.

[2] A. A. Juan, J. Faulin, S. E. Grasman, M. Rabe & G. Figueira, A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems, *Operations Research Perspectives*, vol. 2, pages 62-72, 2015.

[3] G. Wäscher, H. Hausner & H. Schumann, An improved typology of cutting and packing problems, *European Journal of Operational Research*, vol. 183, pages 1109-1130, 2007.

[4] T. Romanova, J. Bennell, Y. Stoyan & A. Pankratov, Packing of concave polyhedra with continuous rotations using nonlinear optimization, *European Journal of Operational Research*, vol. 268, nº 1, pages. 37-53, 2018.

[5] Y. G. Stoyan, N. I. Gil, A. Pankratov & G. Scheithauer, Packing non-convex polytopes into a parallelepiped, Technische Universitat Dresden, 2004.

[6] K. Fischer, B. Gartner & M. Kutz, Fast Smallest-Enclosing-Ball Computation in High Dimensions, *Algorithms - ESA 2003: 11th Annual European Symposium*, Budapest, 2003.

[7] M. Gendreau & J.-Y. Portvin, Handbook of Metaheuristics, Second ed., New York: Springer, 2010.

An exact and heuristic approach for the sequencing cutting problem

Daniel Cuellar-Usaquen, Marcelo Botero, Alejandro Palacio, Emilia Ospina, and David Álvarez Martínez
Universidad de Los Andes
Carrera 1 Nº 18A - 12, Bogotá - Colombia

dh.cuellar, m.botero15, a.palacio10, e.ospina10, and d.alvarezm@uniandes.edu.co

Abstract

The material cutting process consists of two highly complex problems: First, it is necessary to find the optimal cutting pattern or location of the pieces within the plate such that the used area is maximized. Second, it is necessary to find the cutting sequence over the plate so that the pieces are extracted in the shortest possible time. The structure of the sequencing cutting problem can vary according to the technology used in the process. In the literature, this problem is addressed through exact and approximate optimization. In industries where material can be considered a commodity, the sequencing phase is decisive due to the need to achieve an economic and efficient operation. Due to this fact, the computing time required by the optimization methodology is included as part of the problem's total processing time. These types of minimization demand heuristic processes to be specialized in such a way that a good solution is guaranteed, and exact models to use non-conventional formulation techniques in search of computational efficiency. This study shows an exact and heuristic approach to solve the bidimensional sequencing cutting problem, under the use of an industrial manipulator integrated to a plasma cutter. Both approaches are compared through benchmarking, using the classic cutting instances from the literature. The possibility to analyze and include the quality of the pieces, given the cutting sequence, is left as a future work proposal.

1 Introduction

The problem of two-dimensional sequencing for material cutting processes originates when it is necessary to extract multiple smaller rectangular pieces of diverse dimensions from one large rectangular plate [1]. This particular problem is relevant in various industries that deal with various materials that must be cut and adjusted so that their dimensions meet the requirements of each customer. The objective is to find a cutting sequence for a given set of pieces such that the sum of the time of the cutting process and the computational time required for mapping such process is minimized, all with the purpose of increasing productivity and reducing the costs associated to machinery, workforce and infrastructure [2].

The problem of bidimensional sequencing for material cutting processes can be represented as a graph $G = (V, E)$ where the set of vertices V is composed of the extreme points of every piece that shall be cut, while the set of edges E is in turn divided in two subsets $(S, H) \subseteq E$. S represents the subset of mandatory edges (the pieces that must be cut exactly once by the industrial manipulator to guarantee and maintain the quality of the extracted piece) while subset H, of size $|V^2|$ contains all the possible aerial movements the industrial manipulator can make in order to reach any particular corner of the remaining pieces to be cut. Therefore, flowing over the edge $(i, j) \in S$ would mean to effectuate the cut between vertex$(i, j) \in V$; while flowing over the edge $(i, j) \in H$ would mean to travel between vertex i and vertex j through the air.

It is possible then to define route R as a feasible solution to the problem given that, starting in source node $(0,0)$, R flows exactly once over every edge $(i, j) \in S$. Considering that the objective of the study is to minimize the total processing time for a particular instance, it is necessary to define a function that computes total time of route R as $f(R) = g(R) + h(R)$, where $g(R)$ is equivalent to the sum of the cutting process and air movement times, and the computational time is $h(R)$.

Bogotá, March 6th, 2019

2 Proposed Methodologies

Linear Programming Model

For the first optimization methodology (MIP), the initial network must be modified. A routing network is defined by creating two additional nodes for each mandatory edge. These nodes correspond to the two possible ways of making a cut between point A and point B $(A-B, B-A)$, and are designated as cutting nodes. Node $(0,0)$ is connected to the starting node (n_I), and all nodes corresponding to a vertex are connected to the end node (n_F). The purpose of this network is to trace the cutting route. To accomplish this, the supply of n_I is set to 1 and the demand of n_F is set to -1. This parameter is set to 0 in every other node.

In addition to this, a second network called auxiliary network is designed. This network is very similar to the previously described routing network, but it is intended to ensure every cut within the problem; namely, it ensures that at least one of the cutting nodes is visited. To accomplish this, instead of connecting every vertex to an end node, each pair of cutting nodes is connected to a demand node. For this network, the supply of node n_I equals $|S|$ and the demand of all demand nodes is set to -1.

The proposed MIP uses both networks in a joint manner, aiming to create a unique uninterrupted route or cutting pattern that visits every pair of nodes that make up a mandatory edge. Given that each network is associated to a different flow variable x_{ij} and y_{ij}, additional constraints must be included to link the variables of both networks, ensuring a certain pair $(i,j) \in S$ is only assigned to the resulting route if the constraints for both networks are met.

Because of the nature of the solution of the proposed formulation, it is impossible to trace the order of the resulting cutting pattern without additional processing. Given the fact that the formulation ensures each node is entered and exited an equal number of times, the MIP solution corresponds to an Eulerian graph. Lastly the Hier-Holzer algorithm is executed over this graph, resulting the cutting sequence of the problem.

Metaheuristic approach

For the second proposed methodology a GRASP algorithm is adapted to the problem. The GRASP as presented in [3], is composed of two phases: construction and improvement, where the best solution obtained is saved after all iterations. After identifying all mandatory edges $(i,j) \in S$, a constructive algorithm must follow two procedures depending on the source node of the problem (n_0). If the position of the manipulator corresponds to a node that makes up a mandatory edge $(n_0 \in S)$, there are four possible edges to flow through to cut the material. In the opposite case, if the position of the arm corresponds to a vertex that makes up a non-mandatory edge $(n_0 \in H)$, the arm will have $|S|+1$ possible alternatives to move to a node that makes up a mandatory edge.

To decide which edge (n_0,n_j) must be used, a restricted list of candidates (RLC) is created with the most promising candidates for the objective function. Each candidate is evaluated to verify if it must be included to the RLC. Once the RLC candidates are properly identified, a random candidate is selected. This procedure is repeated until $S = \emptyset$, meaning that all mandatory cuts have been made.

It is necessary for the construction phase to link to the characteristics of the different instances of the problem. As such, parameter α, which limits the size of the RLC, is not fixed as suggested in [4]. A training phase corresponding to 30% of the total iterations is done to select the value of α, according to a subset of possible values and their respective results. In this way, the value of α for the remaining iterations is selected according to the probabilities associated to the performance obtained by the historical values of α. This reactivity is done to generate robustness in the algorithm and obtain better solutions.

In this study, an improvement movement is considered for the global solution of the generated route. This movement consists of a swap between the nodes of two edges that are included in the route, breaking two edges and repairing the route with two new edges. All possible swaps are evaluated, and only the best alternative is executed until there is no possible improvement.

Bogotá, March 6th, 2019

3 Results

The mathematical model described above was implemented in Python 3.6 and solved using Gurobi 8.1.0, while the algorithm GRASP was implemented in programming language C++. Both methodologies where executed with hardware Intel® Core™ i7-7700HQ CPU @ 2.80GHz, RAM16GB and operating system Windows 10 Enterprise 64 bits. The set of instances proposed by J. Beasley [5] was run and the average result for each of these, as calculated through both methodologies, is presented in Table 1. The GRASP algorithm was calibrated with a fixed number of 250 iterations while a maximum execution time of one hour was set in Gurobi ®. As can be observed in Table 1, the GRASP algorithm reaches the optimal operational material cutting time only for the set of instances defined as M. However, computational times are relatively inferior than those for MIP methodology leading to a GAP of -34.43% in the total processing time.

		APT	B	GCUT	HZ	M	U	UU
GRASP	Cutting Time (s)	80,87	101,35	39,97	4,18	**9,78**	218,91	34,63
	Computing Time (s)	0,48	0,52	0,21	0,08	0,04	0,46	0,11
	Total Processing Time (s)	81,34	**101,87**	40,18	4,26	**9,81**	219,37	**34,74**
MIP	Cutting Time (s)	77,27	101,11	37,59	3,95	**9,78**	213,70	34,45
	Computing Time (s)	1,03	0,98	0,44	0,29	5,19	1,31	0,39
	Total Processing Time (s)	**78,30**	102,09	**38,02**	**4,23**	14,97	**215,01**	34,84
	GAP (%)	3,89	-0,21	5,67	0,60	-34,43	2,03	-0,30

Table 1. GRASP and MIP results for classic cutting instances.

The results obtained by the MIP methodology are better in four groups of the proposed instances, this is due to the difference in the quality of the proposed solutions with specific regards to cutting process times resulting in a GAP between 0.6% and 6%. The GRASP algorithm has better performance in three of the trial instances, where it is evident that computational times are inferior, and the quality of the solution has slight differences. When analyzing the general GAP between both techniques it is possible to infer that the GRASP algorithm has better performance.

4 Conclusions

Two solution alternatives were implemented for the two-dimensional sequencing for material cutting problem: the GRASP algorithm and a MIP based on network flow modelling, aiming to compare the performance and quality of the solutions (measured in processing times). The results present a better general performance for the GRASP algorithm regarding computing time, even though it presents small differences regarding the cutting time. The network modelling done for the MIP allows for optimal results to be reached in considerable computing times, reducing processing times significantly.

As future work, an incorporation of the quality of the obtained pieces regarding the trajectory of sequencing is proposed, with aims to avoid reprocesses and loss of material.

References

[1] D. Álvarez Martínez, E. M. Toro, and R. A. Gallego, “Estudio comparativo de algoritmos basados en cúmulo de partículas para resolver el problema de empaquetamiento en placas,” *Ing. y Compet.*, vol. 13, 2011.

[2] G. Ghiani and G. Improta, “The laser-plotter beam routing problem,” *J. Oper. Res. Soc.*, vol. 52, no. 8, pp. 945–951, 2001.

[3] M. G. C. Resende and C. C. Ribeiro, “Greedy Randomized Adaptive Search Procedures: Advances, Hybridizations, and Applications,” *Handb. Metaheuristics*, vol. 146, no. 1989, pp. 283–319, 2003.

[4] R. Marti and J. Moreno-Vega, “Metodos multiarranque,” *Intel. Artif.*, vol. 19, no. 2, pp. 49–60, 2003.

[5] J. Beasley, “An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure,” *O. Res.*, v. 33, 1985.

Bogotá, March 6th, 2019

A Hybrid Decision Support System for Supplier Selection: An integration of Simheuristics and MCDM

Mohammad Dehghanimohammadabadi [1]

[1] Northeastern University
360 Huntington Ave, Boston, MA 02115
m.dehghani@northeastern.edu

Abstract

Supplier selection plays a key role in achieving the objectives of a supply chain system. Multiple strategic, operational, quantitative, and qualitative criteria influence the supplier selection process. A wide spectrum of criteria have been introduced, classified, and used by researchers and practitioners to evaluate the suppliers' performance; however, measuring and employment of all of these criteria is impractical in real-world scenarios due to the budget, time, and information limitations. In this study, a Decision Support System (DSS) is proposed to help managers identify the most significant criteria for the supplier selection process. This DSS is a two-fold integration of Multi-Criteria Decision Making (MCDM) and Simheuristic where a Metaheuristic algorithm is combined with a simulation module. In an iterative manner, the Metaheuristic algorithm changes the setting of the MCDM, until the best/good combination of the criteria for the supplier selection process are obtained. The simulation model is utilized as a cost function of the Metaheuristic algorithm, in which the performance of the generated solutions (MCDM settings) are tested.

1 Introduction

To improve the performance and focus on core business, numerous firms have been downsizing and outsourcing extensively [1]. In manufacturing industries, a large portion of sales revenue is spent on materials, parts, and equipment. According to [2], the raw material purchased from outside vendors constitutes 40%–60% of the unit cost of a product for the most U.S. firms. Manufacturers can reap large profits from a small percentage reduction in the cost of materials, which makes supply chain management a key competitive weapon [3]. A structured and rigorous supplier selection and evaluation could considerably improve organization performance and final product quality [4], and in return substantially create more cost savings for the supply chain [5]. Additionally, companies hope to establish a long-term relationship with suppliers to achieve cost competitiveness and excellences in services [6], which makes the importance of supplier selection procedure even more pronounced.

Essentially, supplier selection is a complicated multi-criteria decision-making (MCDM) process which contains a multitude of quantitative and qualitative metrics, aiming to reduce the initial set of potential suppliers to the well-rounded choices [7]. Finding appropriate and adequate criteria is a primary challenge in supplier selection and could fundamentally affect the overall outcome. Defining the most effective criteria is not trivial and requires careful analysis, which is key to getting a qualified supplier. It is obvious that the more criteria are involved, the more likely it is that decision is accurate. But, on the other hand, measuring the criteria requires a lot of effort, whereas changing customer preferences require a broader and faster supplier selection [8]. In some cases criteria are fairly intangible in which assigning trustworthy scores for intangible criteria is not intrinsically a straightforward task and is not easy to measure [9], [10]. Therefore, assigning exact numeric scores to represent preferences among criteria is a fundamental challenge for decision makers [10]. This issue becomes more critical for firms which periodically need to evaluate and re-select their supplier(s). Criteria selection and measurement could induce additional cost on these firms due to the efforts for obtaining suppliers information.

In addition, due to the associated complexities and uncertainties in supply chain systems, including all possible criteria to analysis may not necessarily lead to the best supplier selection. Each firm has a different strategy in the supply chain in terms of the characteristics of the product, and does not necessarily to include all the criteria into a final decision making [11]. As stated by Holm and Vo [12], there might be a mismatch between supplier selection practices and supplier evaluation criteria, and that the theoretical models provided by researchers may not be prominent in real world applications. Basically,

the effect of selected supplier(s) on the supply chain performance needs to be conducted in a real-world setting rather than numerical data-based problems. In other words, buying companies need to establish criteria relevant for own organization [12] which could reflect the real drivers of profit for the firm [13]. As a result, instead of considering all the criteria, it is more practical to quantify major criteria that cumulatively fulfill the supply chain requirements and have the major impact. Limiting the number of criteria to the most effective ones permits decision-makers to easily investigate suppliers performance in a reproducible and timely manner with minimal effort.
To address the above-mentioned issues, this paper introduces a threefold integrated framework consisting of optimization, MCDM, and simulation scheme. This hybrid model determines the most appropriate and practical criteria for supplier selection process and provides their corresponding weights. In an interactive manner with a simulated environment, the performance of the different combinations of suppliers is evaluated until desirable solution is found. Therefore, through this paper, the following contributions are made:

- Define, design, and develop a practicable decision support system for supplier selection
- Offer a method to distinguish important from unimportant criteria
- Calculate the weight of important criteria for decision making
- Evaluate the performance of selected supplier in accordance with a real-world system by using simulation

2 Framework Design and Implementation

This section presents a description and discussion of the design and technical implementation of the proposed Simheuristic framework integrated with MCDM. A detailed structure of the proposed framework is illustrated in Figure 1. To support the aforementioned goals for the supplier selection process, this DSS is built upon three modules including (i) optimization (Metaheuristics), (ii) multi-criteria decision making (MCDM), and (iii) simulation.

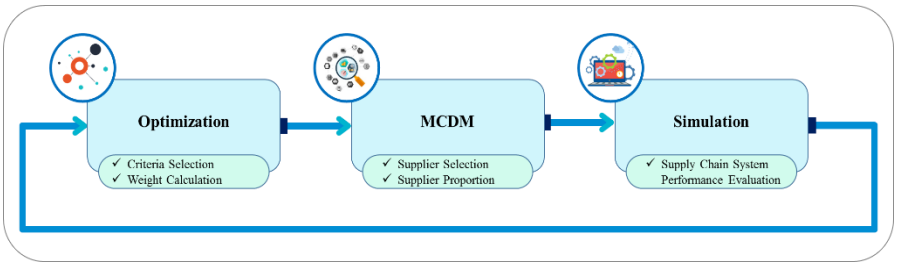


Figure 1: Design of the proposed simulation-optimization framework integrated with MCDM

In this recursive procedure, optimization module is the initial step, in which a shortlisted set of criteria and their corresponding weights are selected. Essentially, in every iteration of the framework, the optimization algorithm generates a new solution, which is decomposed into two different responses. The first response determines the list of selected criteria to be used by MCDM module, and the second one defines the corresponding weight or importance of criteria that are selected. Therefore, in every iteration of the model, the selected criteria and their weight are subject to change based on the generated solution given by the Metaheuristic algorithm.
The provided responses from the optimization module are transferred to the second stage to deploy a supplier selection process. At this point, the MCDM module takes the role of judging supplier alternatives using the provided criteria and weights from the optimization module. MCDM module uses a structured and numerical approach to find a set of top suppliers from the decision matrix. The obtained solution at this stage consists of a list of top suppliers and their proportion. The supplier proportion indicates how much percentages of the total demand is supplied by each supplier, and all add up to 100%.
The third component of this framework is the simulation module, which provides a detailed representation of the real-world supply chain system, with its dynamic processes and stochastic events. Simulation

offers a testbed to evaluate the impact of the selected supplier(s) from the decision-making module on the supply chain system performance. In each iteration, the simulation model considers the characteristics of the new selected suppliers while running the model. Therefore, simulation results effectively reflect the efficiency of the selected suppliers for the actual supply chain system. Once the simulation run is completed, the system performance measure is sent to the optimization module to generate the next solution. The cycle continues until the stopping criteria are met.

Depending on the supply chain system structure and business needs, any number of criteria or suppliers' alternatives could be included in this framework. A decision maker could easily consider a list of potential factors which intuitively might be crucial for the supplier selection process, and ultimately discover a limited set of the most important ones. Moreover, this approach removes the need of criteria weight configuration. The criteria weights are automatically calculated, and thereby, another layer of complexity is removed from the problem. Therefore, this framework identifies a shortened list of important criteria with their weights. This could benefit the decision makers by reducing the time and efforts required to perform the decision-making analysis in the future.

This framework could be applied either for a single-source or a multiple-source strategy supply chain system. If a single source strategy is desired, the model tries to find the most efficient supplier for the system; however, if the multiple-source strategy is attempted, the model provides a pool of suppliers with a supply proportion for each. The supply proportion indicates how much percentage of the demand is covered by each supplier(duplicate). Last but not least, using simulation model ensures that the selected supplier(s) has(have) a good fit for the real-world system.

Essentially, the developed framework is a simheuristic model, in which a metaheuristic algorithm is incorporated with a simulation model. As shown in Figure 2, the framework starts with the Initial Decision Matrix which includes all of the predefined criteria. The objective of the optimization module is to find the most relevant, practical, and useful criteria for the supply chain system. To do this, Genetic Algorithm (GA), which is a powerful evolutionary algorithm, is used as the optimizer of the model. Each chromosome (solution) derived from GA is translated into responses to obtain a pair of selected criteria and their corresponding weights. Then, the Initial Decision Matrix is updated, and un-selected criteria are removed. Therefore, the updated decision matrix is a cross table of supplier alternatives and the selected criteria. At this point, the decision-making algorithm which is TOPSIS is triggered to find the set of preferred suppliers and their proportion.

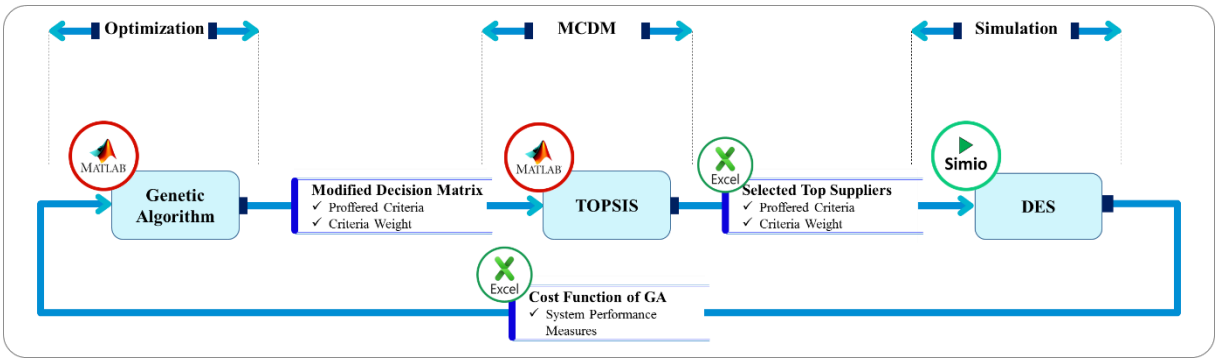


Figure 2: Software architecture of the proposed framework

In order to incorporate the suppliers' characteristics into the simulation model, a input data table is created in excel, which provides suppliers info for the Simio model. Whenever the combination of the suppliers' changes, the input table updates. Therefore, after each TOPSIS run, the simulation input data is adjusted according to the new selected suppliers. At this point, simulation model is ready to run. To trigger Simio from MATLAB, the developed module by [14] is used, which particularly calls Simio and runs experiments, and provides the simulation results. The provided results from Simio are used as a Cost Function for GA to determine the quality of the solution (selected criteria and weights). At each iteration of GA, the best solutions are retained, and the process repeats until the algorithm reaches its stopping criteria.

References

[1] B. Karpak, R. R. Kasuganti, and E. Kumcu, "Multi-objective decision-making in supplier selection: An application of visual interactive goal programming," *J. Appl. Bus. Res. JABR*, vol. 15, no. 2, pp. 57–72, 2011.

[2] E. Sucky, "A model for dynamic strategic vendor selection," *Comput. Oper. Res.*, vol. 34, no. 12, pp. 3638–3651, 2007.

[3] L. J. Krajewski, L. P. Ritzman, and M. K. Malhotra, *Operations management*. Addison-Wesley Singapore, 1999.

[4] M. Bozorgmehr and L. Tavakoli, "An EFQM-based supplier evaluation and verification methodology based on product's manufacturing policy and technology intensity: industrial gas turbine industry as a case study," *Int. J. Integr. Supply Manag.*, vol. 9, no. 4, pp. 276–306, 2015.

[5] D. Mohammaditabar, S. H. Ghodsypour, and A. Hafezalkotob, "A game theoretic analysis in capacity-constrained supplier-selection and cooperation by considering the total supply chain inventory costs," *Int. J. Prod. Econ.*, vol. 181, pp. 87–97, 2016.

[6] M. A. Khan, "Total quality management and organizational performance-moderating role of managerial competencies," *Int. J. Acad. Res.*, vol. 3, no. 5, pp. 453–458, 2011.

[7] F. R. L. Junior, L. Osiro, and L. C. R. Carpinetti, "A comparison between Fuzzy AHP and Fuzzy TOPSIS methods to supplier selection," *Appl. Soft Comput.*, vol. 21, pp. 194–209, 2014.

[8] L. De Boer, E. Labro, and P. Morlacchi, "A review of methods supporting supplier selection," *Eur. J. Purch. Supply Manag.*, vol. 7, no. 2, pp. 75–89, 2001.

[9] F. T. Chan and H. J. Qi, "Feasibility of performance measurement system for supply chain: a process-based approach and measures," *Integr. Manuf. Syst.*, vol. 14, no. 3, pp. 179–190, 2003.

[10] S. Nikghadam, B. L. Sadigh, A. M. Ozbayoglu, H. O. Unver, and S. E. Kilic, "A survey of partner selection methodologies for virtual enterprises and development of a goal programming–based approach," *Int. J. Adv. Manuf. Technol.*, vol. 85, no. 5–8, pp. 1713–1734, 2016.

[11] B. D. Rouyendegh and T. E. Saputro, "Supplier selection using integrated fuzzy TOPSIS and MCGP: a case study," *Procedia-Soc. Behav. Sci.*, vol. 116, pp. 3957–3970, 2014.

[12] T. L. Holm and T. T. T. Vo, "Supplier evaluation criteria: A comparative case stydy between the fashion retail industry and the subsea industry," 2015.

[13] "Clarifying the concept of product–service system." [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0959652601000397. [Accessed: 01-Mar-2017].

[14] Simio and MATLAB by Mohammad Dehghani, *Part 1 Calling MATLAB from SIMIO*. 2017.

Minimization of the picking time of a warehouse in the Batch Assignment and Sequencing problem with a simulation model

Liany Tobon[1], Daniel Mendoza[2]

[1] Universidad del Atlántico
Carrera 30 Número 8- 49 Puerto Colombia - Atlántico, Colombia
lltobon@mail.uniatlantico.edu.co

[2] Universidad del Atlántico
Carrera 30 Número 8- 49 Puerto Colombia - Atlántico, Colombia
danielmendoza@mail.uniatlantico.edu.co

Extended Abstract

This paper addresses the Batch Assignment and Sequencing problem considering multiple pickers and due dates of the orders that must be delivered in a warehouse. Order-picking systems in warehouses have been widely studied in the literature given their importance of making them more efficient due to the high costs and consumption of resources. In the process the orders are grouped in batches to be assigned and sequenced by the pickers equipped with a roll cage that will make tours through the warehouse where they must make stops in each of the storage locations of the items found in the batch they are recovering.

This research seeks to minimize the tardiness of the order picking process. The batches are organized considering the due dates of the orders to avoid incurring delays and cost overruns. The problem represents how and in what sequence the batches should be assigned to the available pickers to improve warehouse performance.

Batch Assignment and Sequencing problems are NP-hard, due to the fact that no exact algorithms are found that allow solving them for large instances in acceptable computational times. This has led different authors to treat the problem through the use of different metaheuristics such as Genetic Algorithms (GA), Variable Neighborhood Search, Ant Colony Optimization (ACO), Seed Algorithm, Taboo Search, among others.

The literature consulted does not evidence studies that provide a solution to the Batch Assignment and Sequencing problem through the use of Simulation Optimization techniques. The simulation has been a method of solution used ultimately. However, with the development of new technologies and the arrival of computers with more powerful hardware and specialized simulation software, they have made it possible to obtain solutions for large instances of NP-hard problems by providing shorter computational times for investigations.

This work, in progress, considers Batch Assignment and Sequencing problem to minimize the total tardiness of the order picking process considering due dates of the orders. Each batch is assigned in a single sequencing position that the pickers have, to then perform the corresponding routing. It is assumed that the batches have been previously processed by grouping orders whose sum of the number of items does not exceed the capacity of the roll cage with which the pickers have to carry out their tours.

In the use of the metaheuristic, each batch must be assigned to a single picker and have only one sequencing position. The pickers do not have a limit number of positions so they can do the collection of any number of batches existing in the warehouse following the established order. Subsequently, the picker proceeds to make the corresponding tour leaving the depot and returning when he has visited all the storage positions of the items contained in the batches. Therefore, it is understood that the processing time of the orders corresponds to the time elapsed since they are grouped until the picker returns to the depot with the batch in which they are assigned. In this way the tardiness of each order is the positive difference between the moment of the processing of the batch containing the order ended and its due date.

The warehouse that considers this research consists of a manual order picking system including two blocks, five selection aisles, three cross aisles and a depot where once the batches have been obtained and made their assignment and sequencing, the pickers are prepared to perform the tours

for the batches that they will recover.

In this sense, it is important to bear in mind that the disposition of the articles in the warehouse corresponds to an ABC classification of the products with 20 different types of articles. Thus, it should be noted that the warehouse has 500 storage locations, so a proportionality is assumed between the percentage of demand for which each class is responsible and its number of storage locations. Likewise, the items associated with the classes with the highest rotation are assigned to storage locations closest to the deposit.

From the above, class A is described, with 10% of all types of articles with the highest rotation, responding to 52% of the total demand. On the other hand, class B contains 30% of articles with an intermediate rotation responsible for 36% of the total demand and also, class C covers 60% of articles with the least rotation and, therefore, the lowest percentage of demand with 12%.

To minimize the picking time, in conjunction with the Flexsim 2019 simulation software, two metaheuristics are being used: A Seed Algorithm for the initial solution of the problem and a Variable Neighborhood Search (VNS) to improve the initial solution. In this way, from the seed algorithm orders are organized from lowest to highest due date for the assembly of batches that contain them without exceeding the capacity of the roll cage, which corresponds to 45 units, to be subsequently assigned and sequenced in the sequencing positions of the pickers.

Initially, said algorithm is tested with 90 and 180 orders for the assembly of batches. The articles of these orders are evenly distributed in a range of 5 to 10 for each order and were generated according to the demand percentages of each class established as previously described, that is, 52% of all articles of the generated orders correspond to class A, 36% to class B and the remaining 12% to class C.

For its part, the VNS algorithm is used to improve the initial solution in the assignment and sequencing problem, so the batches will not change the orders contained in them, but the processing times given by the simulation vary once this has been started to determine the total tardiness until the VNS can find a local optimum. This algorithm works through an exploration of the solution space through a sequence of neighborhood structures that go from N1 to Nn. The process begins with the initial solution s' and the best neighbor is established with solution s of N1 considering the value obtained for the objective function. Then, if s is a better solution than s', s becomes the new solution and the first neighborhood structure N1 is reconsidered. If the above does not happen, the next neighborhood is explored until the algorithm cannot find an improvement in the last structure of the neighborhood Nn, finding a local optimum with respect to the other neighborhood structures.

Consequently, it was found that with the use of Simulation Optimization it is possible to reduce the picking time between 10 and 20 percent when due dates are loose. However, in order to simulate a more realistic order-picking system, it is necessary to evaluate larger instances with more tight due dates as they are normally in the different warehouses. It is for this reason that this research is ongoing in order to achieve better solutions for the problem posed.

When addressing the problem of the way described above, a methodology is applied that allows dealing with more real instances considering stochastic aspects through the use of a simulation model. In this sense, with the implementation of simulation techniques it is possible to observe measures of system performance such as the percentage of resource utilization, service level, process response time and other variables that cannot be easily observed when only metaheuristics are used to find a solution.

In addition, it is worth mentioning that it is possible to extend the bases acquired when carrying out this research work to other future projects that present similar problems. In this sense, problems such as those treated in Which items should be stored together? A basic partition problem to assign storage space in group-based storage systems (Dominik Kress, Nils Boysen & Erwin Pesch, 2016) where a basic partition problem is considered, which when resolving it allows access to the minimum number of sets containing Stock Keeping Units (SKUs) to retrieve orders according to an established collection policy. For its part in Order picking with multiple pickers and due dates - Simultaneous solution of Order Batching, Batch Assignment and Sequencing, and Picker Routing Problems (Scholz, A., Schubert, D., & Wäscher, G., 2017) is regarded as future research to deal with the same problem presented in this article with narrow corridors and blockers of the selector, so when performing a simulation model it would be possible to visualize different aspects of the behavior of the order picking process according to the particularities presented by the different storehouses.

Keywords. Metaheuristic, order picking process, simulation, variable neighborhood search.

References

[1] De Koster, R., Le-Duc, T., & Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational*, 481-501.

[2] Jiménez B., M., & Gómez A., E. (2014). Mejoras en un centro de distribución mediante la simulación de eventos discretos. *Industrial Data*, 143-148.

[3] Scholz, A., Schubert, D., & Wäscher, G. (2017). Order picking with multiple pickers and due dates – Simultaneous solution of Order Batching, Batch Assignment and Sequencing, and Picker Routing Problems. *European Journal of Operational Research*, 461- 478.

[4] Vélez, L. (2016). Modelamiento eficiente de la preparación de pedidos en un almacén usando un metaheurístico de Búsqueda Tabú. Medellín: Universidad Nacional de Colombia.

[5] Wan, L.-F., & Shi, L.-Y. (2013). Simulation Optimization: A Review on Theory and Applications. *European Journal of Operational Research*, 1957-1968.

[6] Dominik Kress, Nils Boysen & Erwin Pesch. (2016). Which items should be stored together? A basic partition problem to assign storage space in group-based storage systems. *IISE Transactions 49, 13-30.*

A matheuristic for order picking problems in e-commerce warehouses

Mustapha Haouassi[1,2], Jorge E. Mendoza[3], Yannick Kergosien[1], Louis-Martin Rousseau[2]

[1] Université de Tours, LIFAT EA 6300, CNRS, ROOT ERL CNRS 7002
64 Avenue Jean Portalis 37200 TOURS
mustapha.haouassi@etu.univ-tours.fr
yannick.kergosien@univ-tours.fr

[2] Ecole polytechnique de Montréal, CIRRELT
2900 Boulevard Edouard-Montpetit H3T 1J4, Montréal (Québec), Canada
mustapha.haouassi@polymtl.ca
louis-martin.rousseau@polymtl.ca

[3] HEC Montréal
3000 Chemin de la cte Sainte Catherine H3T 2A7, Montral (Québec), Canada
jorge.mendoza@hec.ca

Abstract

Fast delivery is one of the most popular services in e-commerce retail. It consists in shipping the items ordered on-line in short times (1h, 2h, or same day). The customer orders thus come with due dates, and respecting this latter is pivotal to ensure a high service quality. We focus through this work on the order picking process. In a nutshell, order picking consists in regrouping orders into batches, assigning batches to order pickers, sequencing the batches assigned to each order picker, and designing the picking tours of each order picker to retrieve the assigned items. To deal with the time-critical orders, the e-commerce retailers often arrange their warehouses using a mixed-shelves storage policy, leading unit loads of an item to be located in several storage locations in the warehouse. We thus assume a mixed-shelves storage warehouse in our problem definition. Furthermore, we allow, unlike what is classically assumed in the picking literature, the items of a customer order to be assigned to different order pickers (splitting the customer orders). The objective of the problem is to design the picking tours of the order pickers to retrieve all ordered items and minimize the total tardiness of the customer orders. To solve the problem, we propose a two-stage matheuristic. In the first stage, the algorithm iterates a set of predefined procedures that construct a pool of promising picking tours. After N iterations, the first stage stops, and the second stage takes place. In this stage, we propose two mixed-integer programming (mip) formulations (time-based formulation and position-based formulation) to solve the set covering problem over the pool of picking tours designed in the first stage. The experiments are still in progress and will be presented in the talk.

1 Introduction

E-commerce grew up significantly in the last decades. To be competitive, retailers have to propose a high-level services to their customers. One of the most popular services is ”fast delivery“. Indeed, retailers (such as Amazon) propose their customers to deliver the items ordered on-line in short times (1H, 2H or same day). This service implies efficient picking and delivery processes. The delivery process is classically ensured by a third party (taxis, couriers, transport companies, ...) or by a fleet of couriers sufficient to deliver the customer orders in a short time. We thus focus in this work on the order picking process.

Our problem is defined on a warehouse with a “manual picker to part” system. In such a system, order pickers start from a **depot**, walk through **picking** and **cross aisles** to pick items stored in **shelves** (storage locations), and return to the depot. During a workday, a picker performs several picking tours in the warehouse. In each tour, he retrieves a batch of items using a pick list in which the storage location of each item is identified [2]. In addition, the items requested in each customer order have to be retrieved before a due date. In “fast delivery”, respecting the due date of the customer orders is pivotal to ensure a high service quality.

To optimize the picking process, researchers usually focus on: a) how to regroup the customer orders into batches (order batching problem); b) how to visit the picking locations included in a tour (picker routing problem); c) how to assign the batches to the order pickers and to sequence the batches assigned to each picker (batch assignment and sequencing problem). To the best of our knowledge, [2] is the only article that considers simultaneously the problems a), b) and c). Nevertheless, there are two aspects that make the difference between the problem defined in [2] and our problem: the storage assignment strategy and the splitting of the customer orders. The storage strategy assumed in [2] is the traditional storage assignment strategy (*i.e.*, unit loads of an item are stored in a single shelf). In our problem, we assume a mixed-shelves storage strategy (*i.e.* unit loads of an item are scattered in the warehouse). This latter is known to be well-suited to the business-to-consumer e-commerce, which is characterized by large assortment of items and time-critical picking orders [1]. It is worth noting that selecting from where to pick each ordered item becomes an additional problem to consider while assuming this strategy. The splitting of the customer orders (*i.e.*, to assign items of a customer order to different order pickers) is not allowed in [2] since it can lead to an unacceptable sorting effort. Nonetheless, computer systems used in the picking process grew up leading workers, in Amazon for instance, to become both pickers and sorters. We thus assume the splitting of the customer orders in our problem. In the following we give a formal definition of our problem:

Consider a complete and undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that models the warehouse layout, where $\mathcal{V} = \{0\} \cup \mathcal{N}$ is the node set and $\mathcal{E} = \{(i,j) \in \mathcal{E} : i,j \in \mathcal{V}, i < j\}$ the edge set. In $\mathcal{V}$, node 0 represents the depot while the set $\mathcal{N}$ represents the locations where the pickers stop to collect items around them (pick locations). A non-negative weight $t_{i,j}$ is associated with each edge (i,j), and it represents the travel time between nodes i and j. Figure 1 shows an example of a warehouse with a 2-block layout, the associated pick locations, and the storage locations available at each pick location. Let $\mathcal{O}$ be the set of customer orders. Each order $o \in \mathcal{O}$ consists of a set of items $\mathcal{M}_o$ that must be retrieved before a due date d_o. Let $\mathcal{M}$ be the set of all ordered items. Each item $m \in \mathcal{M}$ is available at a subset $\mathcal{L}_m$ of pick locations, and the quantity of items in each pick location $i \in \mathcal{L}_m$ is known to be $q_{m,i}$. We also associate a weight w_m to each item m. To retrieve the ordered items, a set of order pickers $\mathcal{K}$ performs picking tours using carts of capacity W. Each picking tour v is characterized by a set of pick locations $\mathcal{L}_v$ to visit in order to retrieve a subset of items $\mathcal{M}_v$ that satisfies partially (or fully) a subset of orders $\mathcal{O}_v$. To complete a tour, we need a fixed setup time β^s to prepare it; a search and pick time β^p for each item unit picked during the tour; and a travel time to walk from the depot to the first pick location, between picking locations, and from the last pick location to the depot.

We aim at finding a solution that defines a set Ω_k of sequenced picking tours for each order picker $k \in \mathcal{K}$ satisfying the following constraints: i) each item of each order is picked; ii) the number of items m picked in each picking location $\mathcal{L}_m$ does not exceed $q_{i,m}$; iii) the loads of each tour does not exceed the cart capacity W; and iv) each picking tour starts and ends at the depot. Among all feasible solutions, we seek one that minimizes the sum of the tardiness of the customer orders. The tardiness of an order o is defined as the non-negative difference between its completion time (the completion time of the last tour that retrieves an item in $\mathcal{M}_o$) and its due date.

2 A two-stage matheuristic

To tackle the problem defined in the section 1, we propose a "route-first assemble-second" approach. Algorithm 1 describes the general structure of this approach. The first stage of our matheuristic intends to construct a diverse pool Ω of promising picking tours. The second stage assembles a solution by selecting a subset of tours from Ω, assigning those tours to the set of order pickers $\mathcal{K}$, and sequencing the tours assigned to each picker with the objective of minimizing the total tardiness. Lines 1 to 7 depict the first stage of our algorithm. In a nutshell, this stage consists in invoking N times a set of predefined procedures that construct the pool Ω. At the n^{th} iteration, the algorithm starts by calling

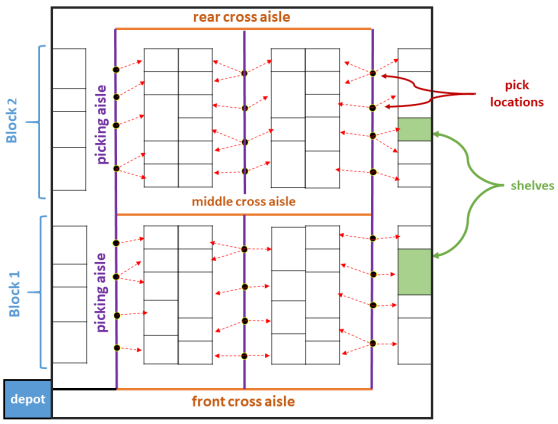


Figure 1: Two blocks layout

Algorithm 1 two-stages Matheuristic: general structure

Require: $\mathcal{G}, \mathcal{M}, \mathcal{O}, \mathcal{K}$

1: **for** $n \leftarrow 0$ **to** $N-1$ **do**
2: $\quad \mathcal{D}^n \leftarrow \texttt{buildVirtualDeadlines}(\mathcal{O})$
3: $\quad \mathcal{Z}^n \leftarrow \texttt{buildOrderedItemsList}(\mathcal{O}, R^n)$
4: $\quad \Omega^n \leftarrow \texttt{designTours}\ (\mathcal{G}, \mathcal{Z}^n, \mathcal{D}^n)$
5: $\quad UB \leftarrow min(UB, \texttt{fastSolutionValue}(\Omega^n, \mathcal{O}))$
6: $\quad \Omega \leftarrow \Omega \cup \Omega^n$
7: **end for**
8: $s^* \leftarrow \texttt{setCovering}\ (\mathcal{G}, \mathcal{M}, \mathcal{O}, \Omega, \mathcal{K})$
9: **return** s^*

the procedure $\texttt{buildVirtualDeadlines}(\mathcal{O})$ (line 2) that defines a set of virtual deadlines $\mathcal{D}^n$ by extending $n*B$ time units the due date of each order. This virtual deadlines will be used by the procedure $\texttt{designTours}$ to control the ordered items assigned to each tour. Furthermore, extending the due-dates at each iteration n intends to construct a diversified pool of tours. Then, the algorithm calls the procedure $\texttt{buildOrderedItemsList}(\mathcal{O}, R^n)$ that builds a list of all ordered items ($\mathcal{Z}^n = \{(m,o) : o \in \mathcal{O}, m \in \mathcal{M}_o\}$) sorted according to a priority rule R^n (line 3). In practice, several priority rules (earliest due-date, shortest distance to depot,...) are applied to ensure a diversification in the pool construction. Next, the algorithm uses the procedure $\texttt{designTours}\ (\mathcal{G}, \mathcal{Z}^n, \mathcal{D}^n)$ to design a set of picking tours $v \in \Omega^n$ (line 4). Each picking tour v is constructed by: regrouping randomly a subset of ordered items from $\mathcal{Z}$, selecting a pick location i from where to retrieve each ordered item, and applying a greedy heuristic to insert i in $\mathcal{L}_v$. Furthermore, the procedure is designed such that the constraints i), ii), iii) and iv) are satisfied at each iteration n. Next, the algorithm invokes a greedy procedure $\texttt{fastSolutionValue}\ (\Omega^n, \mathcal{O})$. This latter computes a fast solution of our problem by sorting the picking tours in Ω^n according the earliest due date (the due date of a tour v is defined as the due date of the earliest order in $\mathcal{O}_v$) and returns the associated objective value (line 5). The upper bound of our problem UB is then updated if necessary. At the end of the iteration n, the algorithm adds to the picking tours designed during the iteration n to the set Ω (line 6). After N iterations, the first stage stops and the second stage takes place. In this stage, the algorithm uses a commercial optimizer to solve the set covering problem (line 8). We thus propose two mip formulations (time-based formulation and position-based formulation) derived from the scheduling literature. To speed up the second stage, the algorithm uses the best upper bound UB found during the execution of the first stage. Note that we restrict our description of the two-stage matheuristic to few pointers. Full details will be exposed during the talk.

3 Computational Experiments

We implemented our two-stages matheuristic in C++ (Visual Studio 2013) and used Cplex Optimizer (version 12.6.0) to solve the set covering formulation (Algorithm 1, line 6). The experiments are still in progress. The test problem instances and results will be discussed during the talk.

References

[1] Nils Boysen, René de Koster, and Felix Weidinger. Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 2018.

[2] A Scholz and G Wäscher. Order batching and picker routing in manual order picking systems: the benefits of integrated routing. *Central European Journal of Operations Research*, pages 1–30, 2017.

Internal Logistics Routing Optimization

Marcelus Fabri[1], Helena Ramalhinho[2]

[1] Universitat Pompeu Fabra
Department of Information and Communication Technologies
R. Roc Boronat 138, 08018 Barcelona, Spain
marcelus.fabri01@estudiant.upf.edu

[2] Universitat Pompeu Fabra
Department of Economics and Business
R. Trias Fargas 25-27, 08005 Barcelona, Spain
helena.ramalhinho@upf.edu

Abstract

The internal logistics in a car manufacture company is an important activity that can lead to improve the efficiency of the production and cost reduction. In this work we describe an internal logistics routing problem in important automotive company and propose an Iterated Local Search algorithm to solve this problem in a realistic environment. This algorithm will be incorporated by the company change the actual static system to a more dynamic and automatic one. We study also the impact of this new system in the company enable the managers to make better decisions and make easier the implementation.

1 Introduction

The logistics in the automotive industry plays an important work. Two areas of Logistics can be taken into account, the external logistics to deliver the pieces and materials to the manufacture side and the logistics to deliver the produced cars to the selling points; on the other side there are other areas of logistics, known as internal logistics that it is related with the movement and deliver of pieces inside the manufacture company. This last one is less known that the external logistics but optimizing this internal logistics activities can lead to a significant improvement in the efficiency of the production as well as in the total production cost. In this work we focus on the optimization of the internal logistics in a car-assembling company, SEAT S.A, located at Barcelona.

The studied problem is an internal logistics routing problem, that consists in determine the best routes to deliver all pieces and material from the warehouse to the car-assembling lines. The company actually uses fixed routes that constantly visit the production lines and the logistics operator checks the level of stock on the line, decide to place or not an order and also deliver the products ordered before. Actually, the company is considering implementing an automatic order system that depending on the production schedule, the type of car scheduled to be produced and the pieces and material needed, the warehouse receives the orders automatically. Therefore, the logistics operator just needs to collect the orders and delivery to the car- assembling line. An important difference between both systems is that in the actual situation the logistics operator always does the same route, meanwhile in the new proposed system the routes can vary along the day.

The main objective of this work is to propose an algorithm that optimizes the routes in the new proposed system. This algorithm must be able to produce efficient solutions in a very short period of time, since orders can enter de system in short period of time. And afterwards, realize a computational experiment comparing both system using several Key Performance Indicators (KPI) suggested by the company to evaluate the potential application of the new system. These KPI consider costs related with the system, but also congestion in the car-assembling line as well the impact on the workers and adaptation needs that maybe need to be training.

To solve this problem, we propose an Iterated Local Search (ILS) [1], since this metaheuristic proved to be able to obtain very good solutions in a very short time. We will use real-data from the car assembly company to test in a real environment the solutions proposed in this work. For doing this, we have developed a simulation system that allows to incorporate all realistic aspect in the study and therefore represent as close as possible the real system.

The contribution of this work is twofold; for one side we studied a real important logistics problem in the car manufacture industry and for the other one we develop a metaheuristic algorithm and simulation system that incorporate innovative aspects that can be extended to other similar routing problems.

2 The internal logistics routing problem

The internal logistics routing problem in the consider company consists in defining the delivery routes from the warehouse to the workstations located along the car-assembling line. In the actual system each logistics operator working in a shift drive a specific vehicle that runs constantly along the production line always visiting the same workstations and doing the following operations at each one: checks the level of stock on the line, decide to place or not an order, and also deliver the products ordered before. This looks like a bus routing system. As mentioned, the company is studying to change the order system by an automatic one where based on the car production schedule, the characteristics of each car to be produced and the materials and pieces needed, the system can place automatically the order of the components to the warehouse in advance. These orders can then vary significantly along the shift or production period, and the logistics operator does not necessarily need to visit all workstation every time he or she drives the vehicle along the production line, because the workstation have different needs along the production period. For example, a workstation with small pieces maybe needs less visits since it can store more pieces in the workstation because it occupies less space, meanwhile a workstation that uses a larger piece maybe needs more visits because of the opposite reasons.

Being able to have dynamic routes, i.e. routes that can vary along the production period and depend on the automatic order system, is a very different system to the one used traditionally in the company and the company wants to be able to implemented it with the less disruption possible and also to be able to justify that an important improvement in terms of efficiency and cost reduction will be achieved.

The internal logistics routing problem can be seen as an extension of the well know Capacitated Vehicle Routing Problem, [2], with some additional constraints as the congestion on the aisles of the production line, the skills of the logistics operator and the dynamic automatic order system. This work is also an extension of the work [3]. In this last work, the authors design the fixed internal logistics routes to the internal logistics routing problem. These works share the car manufacture company but the approach is quite different since in the previous work the automatic order system is not consider.

3 The Iterated Local Search

To solve the internal logistics routing problem, we propose an Iterated Local Search (ILS) algorithm, [1]. The ILS is a popular single-solution based metaheuristics and it is recognized by many authors as a relatively simple yet efficient framework able to deal with complex combinatorial optimization problems (COPs).

LS-based algorithms have been successfully applied to provide near-optimal solutions to different problems in the areas of Logistics, Transportation, Production, Finance, Marketing, etc. ILS extends a problem-specific local search method by introducing a perturbation in each new local optimal solution before restarting the search for a new local optimal solution, therefore can been seen as a search algorithm that considers only local-optimal solutions. ILS is based on four procedures: Generation of an Initial Solution, Local Search, Perturbation, and Acceptance Criterion (see Figure 1).

Each component of the ILS is adapted and designed to consider the special characteristics of the

particular internal logistics routing problem described in the previous section.

```
Procedure Iterated Local Search
    s_0 = GenerateInitialSolution
    s* = LocalSearch(s_0)
    Repeat
        s' = Perturbation(s*, history)
        s'* = LocalSearch(s')
        s* = AcceptanceCriterion(s*, s'*, history)
    Until termination condition met
End
```

Figure 1: Iterated Local Search framework

4 Computational experiment

The main objective of the computational experiment is to evaluate and compare the actual system based on fixed routes with the new one, where the routes are dynamically calculated based on the automatic order systems. All data is provided by the car manufacture company and several Key Performance Indicators (KPY) will be used to evaluate the solutions. The KPY are: the total distance; the number of delay orders; the empty spots in the vehicles; and the congestion produced in the aisles. The system will be tested in different car-assembling lines with different number of workstations that can vary from 50 to 150 and also different types of materials that in some cases can be delivered from different warehouses. At this moment we are performing the computational tests, that it will be present in the conference. The preliminary results show that the algorithm is able to produce very casi-optimal solutions in a very short period of time and comparing with the actual system, it led to a significant improvement.

5 Conclusions

In this work, we consider a real internal logistics problem in a car manufacture company and the main objective is to study and evaluate the implementation of a new automatic order system. We have developed an Iterated Local Search to calculate in a short period of time the best routes, based on the orders received along the production time. We will present the complete computational study in the conference, that due the preliminary results obtained, we are confident that will produce good routes and it would improve in terms of quality and overall costs the actual system. Also, the system will help the managers to make good decisions and be able to implement the new system with less disruption possible in the business.

References

[1] H. Ramalhinho Lourenço, O. Martin, and T. Stützle, "Iterated Local Search: Framework and Applications," in *Handbook of Metaheuristics*, M. Gendreau and J. Y. Potvin, Eds. Springer International Publishing, 2019.

[2] G. Laporte, "Fifty Years of Vehicle Routing," *Transp. Sci.*, vol. 43, no. 4, pp. 408–416, Nov. 2009.

[3] M. F. Lima and H. Ramalhinho, "Designing Internal Supply Routes: a Case Study in the Automotive Industry," in *Winter Simulation Conference (WSC)*, 2017, vol. 91, pp. 399–404.

A Heuristic Oriented Racing Algorithm for the Fine-tuning of Metaheuristics

Eduardo B. M. Barbosa[1], Edson L. F. Senne[2]

[1] Brazilian National Institute for Space Research (Inpe)
Rod. Presidente Dutra, Km. 40 - Cachoeira Paulista, SP - 12630-000, Brazil
eduardo.barbosa@inpe.br

[2] Univ. Estadual Paulista (Unesp)
Av. Dr. Ariberto Pereira da Cunha, 333 - Guaratinguetá, SP - 12516-410, Brazil
edson.senne@unesp.br

Abstract

The metaheuristics have become a powerful tool to solve real-world optimization problems. Its ease adaptability, usually demands effort to correctly define its components (e.g.: problem representation, neighborhood structure, etc.) and parameters to achieve their best performance. Thus, this paper aims to present an approach on the fine-tuning of metaheuristics combining Design of Experiments and Racing algorithms. The key idea is a heuristic method, which explores a search space of parameters looking for candidate configurations near of a promising alternative and consistently finds the good ones. To confirm this approach, we present a case study for fine-tuning a VNS metaheuristic on the classical Traveling Salesman Problem, and compare its results against a well established racing method. In general, our approach proved to be effective in terms of the overall time of the tuning process.

1 Introduction

The algorithms for solving optimization problems, in special the metaheuristics, are highly adaptable to a wide set of problems. However, this feature usually demands a huge effort in the definition of its components (e.g.: problem representation, neighborhood structure, etc.) and parameters to achieve their best performance. Thus, since the last decades there has been a growing academic interests around the automated methods to assist in the fine-tuning of metaheuristics, highlighting the use of Design of Experiments [1, 2, 3], Racing algorithms [4], Neural Networks [5, 6], Fuzzy sets [7], statistical modeling [8, 9] and many others.

In this paper we present an approach on the fine-tuning of metaheuristics combining Design of Experiments (DOE) [10] and Racing algorithms [11, 4] in a heuristic method. The proposed approach brings together some characteristics from different fine-tuning strategies of the literature, such as CALIBRA [12], I/F-Race [13, 4] and ParamILS [14], in a single heuristic method with the ability to define a search space, and the efficiency to focus the search on candidate configurations within the aforementioned search space. To validate our approach, we present a case study with the VNS metaheuristic applied to the classical Traveling Salesman Problem (TSP).

The rest of the paper presents the problem of tuning metaheuristics and our proposed approach (Section 2). The Section 3 brings a case study on the fine-tuning of the VNS metaheuristic by means of different tuning approaches and presents its results. Our final considerations are in Section 4.

2 An Approach on the Fine-tuning of Metaheuristics

In this paper, the problem of fine-tuning of metaheuristics is formalized as: let M be a metaheuristic with a set of parameters, that must be tuned to solve a class of problems P. The parameters of M (e.g.: α, β, ..., ζ) admit a finite set of values (in general, discrete or continuous) and its cardinality can vary according to M and P studied. Let Θ be a set of candidate configurations, such that $\theta \in \Theta$ is any setting of M, then the problem can be formalized as a state-space:

$$S = (\Theta, P) \quad (1)$$

Broadly, the problem consists of knowing which is the best candidate configuration $\theta^* \in \Theta$ to optimize the performance of *M* on *P*. However, its determination is not always simple and, in the worst hypothesis, it can requires a full search in *S*.

2.1 Heuristic Oriented Racing Algorithm

To avoid the full search in *S* and still find a good setting of *M* to *P*, we consider an *agent* (e.g.: a heuristic method) whose the *actions* modify the *state* of *S* (e.g.: by creating candidate configurations).

Given an initial state (e.g: any alternative in *S*), the heuristic method explores (1) by creating new candidate configurations at the neighborhood of some best known alternative as a sequence of sets:

$$\Theta_0 \Rightarrow \Theta_1 \Rightarrow \Theta_2 \Rightarrow ...$$

From the step *k* to *k*+1 the set of candidate configurations is built possibly discarding some inferior alternatives, based on statistical evaluation of a wide range of problems. Given that some candidate configurations persist in that set, they are evaluated on more problem instances. Therefore, search a solution is equivalent to find a path in a graph, that from an initial state reaches the final state, that is, the best setting for *M*.

This approach is called Heuristic Oriented Racing Algorithm (HORA), due the way for which the alternatives are explored through a heuristic method, and its evaluation process by a racing algorithm.

3 Case Study

The HORA method proposed here can be applied on the fine-tuning of different metaheuristics, regardless its nature or parameters number. To illustrate it, we will use as example the configuration of a basic VNS metaheuristic [15, 16] on instances of the TSP.

The TSP is a NP-hard problem [17] extensively studied in the literature [18, 19], and a standard benchmark for new algorithm ideas. The VNS metaheuristic is a trajectory method widely applied to optimization problems, like the TSP. In summary, its strategy is based on dynamical changes in the neighborhood structure of an incumbent solution, moving to the next one if and only if an improvement is made. At each iteration three important stages must be done: shake, local search and move. A high-level pseudo-code is given in Figure 1.

```
procedure vns(S0, Kmax)
 S* ← S0
 repeat
  k ← 1
  repeat
   S' ← shake(S*, k)
   S'' ← localSearch(S')
   S* ← move(S*, S'', k)
  until k ≤ Kmax
 until termination criteria is met
```

Figure 1: Pseudo-code of the basic VNS metaheuristic.

The considered parameters for the fine-tuning process are in Table 1. The parameters levels (low and high) were chosen on early studies with the metaheuristic.

At the beginning of the tuning process we conduct *m* experimental studies[1] with DOE on different instances of the TSP from TSPLIB [20]. Its results let us refine the search space of parameters by

1 In this paper, we chosen arbitrarily *m* = 5, in order to promote diversity for the initial search space of parameters. Thus, at the end of the experimental studies we have five different results for each parameter.

defining a more restrict one, bounded by the maximum and minimum values of each parameter in the training set. Accordingly, the new parameter search space is: $n \subset [1621; 3640]$, $k \subset [3; 9]$ and $\delta \subset [2; 4]$.

Parameter	Description	Low	High
n	Number of iterations	1000	5000
k	Number of neighborhood structures	3	9
δ	Length between the neighborhood structures	1	5

Table 1: The parameters of the basic VNS.

The exploration of the new search space is done through HORA by creating alternatives at the neighborhood of some best known candidate configuration. For each of the alternatives, we run the target metaheuristic during 10s on an expanded set of instances[2]. This process was repeated 100 times and the result is presented in terms of mean and standard deviation ($\mu \pm \sigma$) in Table 2.

For comparisons, we chosen a robust fine-tuning method from the literature with a similar evaluation process to that implemented in HORA. Thus, the same tuning process was repeated with a racing algorithm inspired in I/F-Race method [21] (from here on called RACE). However, the following settings have been considered: $n = \{1621; 1909; 2198; 2486; 2775; 3063; 3352; 3640\}$, $k = \{3; 4; 5; 6; 7; 8; 9\}$ and $\delta = \{2; 3; 4\}$. Since of each possible combination of parameters leads to a different configuration for the metaheuristic, there are $8 \times 7 \times 3 = 168$ predefined settings for the VNS in the RACE scenario. The goal is to select an alternative as good as possible, such that it optimizes the performance of the metaheuristic. The result is in Table 2.

Parameter	HORA Settings ($\mu \pm \sigma$)	RACE Settings ($\mu \pm \sigma$)
n	2298 ± 562	2441 ± 634
k	4 ± 1	4 ± 1
δ	3 ± 1	3 ± 1

Table 2: The proposed parameter settings for the basic VNS.

3.1 Experimental Results

The case study results are similar in terms of parameterization (Table 2). However, we emphasize the differences of the tuning process, such as, the average time of the process, with HORA it takes 7872 seconds, whereas with RACE it demands 14392 seconds. The HORA also stands out in terms of the overall number of experiments performed, that is, 781 against 1410 from RACE. At the end of the tuning process, remains on average 8 surviving alternatives for HORA and 40 for RACE.

Even though it is not the main objective of this work, we run the basic VNS for each one of the proposed settings (Table 2) 15 times on 12 instances of the symmetric TSP with the number of cities varying between 300 and 800. Although the median performance from the metaheuristic tuned through HORA is slightly better, the results are statistically similar at the significance level of 5% to the t-Student and Wilcoxon tests.

4 Final Considerations

The HORA method, presented in this paper, combines DOE and Racing algorithm to efficiently search candidate configurations within an improved search space. As shown in the case study (Section 3), its better performance, relative to a classic strategy from the literature, can be explained by the way of it explores the search space. That is, by means of a heuristic method, which seeks for good alternatives in the neighborhood of some best known candidate configuration, and efficiently evaluates them with a racing method. The results achieved show that the proposed approach is a promising and powerful tool mainly when it is considered the overall time of tuning process. Additional studies shown the effectiveness of HORA with other metaheuristics and problems.

2 The expanded set matches 48 instances with less than 1000 cities of the symmetric TSP from TSPLIB.

References

[1] Coy, S. P.; Golden, B. L.; Runger, G. C.; Wasil, E. A. Using experimental design to find effective parameter settings for heuristics. *J. Heuristics*, 7(1): 77-97, 2001.

[2] Adenso-Díaz, B.; Laguna, M. Fine-tuning of algorithms using fractional experimental design and local search. *Oper Res*, 54(1): 99-114, 2006.

[3] Ridge, E.; Kudenko, D. Tuning the performance of the MMAS heuristic. In: Stützle, T.; Birattari, M.; Hoos, H. H. editors. *International Workshop on Engineering Stochastic Local Search Algorithms* (SLS 2007), pages 46-60, Heidelberg, Germany, 2007.

[4] Birattari, M.; Stützle, T.; Paquete, L.; Varrentrapp, K. A racing algorithm for config- uring metaheuristics. In: Langdon, W. B. editor. *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO 2002), pages 11-18, San Francisco, 2002.

[5] Dobslaw, F. A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks. In: *Sixth International Conference On Natural Computation*, pages 1-4, Cairo, 2010.

[6] Calvet, L.; Juan, A. A.; Serrat, C.; Ries, J. A statistical learning based approach for parameter fine-tuning of metaheuristics. SORT (*Statistics and Operations Research Transactions*), 40(1): 201-224, 2016.

[7] Ries, J.; Beullens, P.; Salt, D. Instance-specific multi-objective parameter tuning based on fuzzy logic. *European Journal of Operational Research*, 218: 305-315, 2012.

[8] Bartz-Beielstein, T.; Lasarczyk, C.; Preuss, M. Sequential parameter optimization. In: *Congress on Evolutionary Computation* (CEC 2005), pages 773-780, Piscataway, 2005.

[9] Hutter, F.; Hoos, H. H.; Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. In: Coello Coello, C. A. editor. *Learning and intelligent optimization*, 5th International Conference (LION 5), pages 507-523, Heidelberg, Germany, 2011.

[10] Montgomery, D. C. Design and analysis of experiments. 8th ed. John Wiley & Sons Inc, 2012.

[11] Maron, O., Moore, A. W. Hoeffding races: accelerating model selection search for classification and function approximation. *Advances in Neural Information Processing Systems*, 59-66, 1994.

[12] Adeson-Diaz, B., Laguna, M. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54(1): 99-114, 2006.

[13] Balaprakash, P., Birattari, M., Stützle, T., Dorigo, M. Improvement strategies for the F-Race algorithm: sampling design and iterative refinement. In: *4th International Workshop On Hybrid Metaheuristics*, pages 108-122, 2007.

[14] Hutter, F., Hoos, H., Leyton-Brown, K., Stützle, T. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36: 267-306, 2009.

[15] Hansen, P.; Mladenovic, N. An introduction to variable neighborhood search. In: Vob, S.; Martello, S.; Osman, I.; Roucairol, C. *Metaheuristics: advances and trends in local search paradigms for optimization*, Chap. 30, pages 433-458, 1999.

[16] Hansen, P.; Mladenovic, N. Variable neighborhood search: principles and applications. European *Journal of Operational Research*, 130: 449-467, 2001.

[17] Lenstra, J. K.; Rinnooykan, A. H. G.; Brucker, P. Complexity of machine scheduling problems. In: Hammer, P. L.; Johnson, E. L.; Korte, B. H.; Nemhauser, G. L. *Annals of Discrete Mathematics*, pages 343-362, Amsterdam, 1977.

[18] Matai, R.; Singh, S. P.; Mittal, M. L. Traveling salesman problem: an overview of applications, formulations, and solution approaches. In: Davendra, D. *Traveling salesman problem: theory and applications*, Chap. 1, pages 1-24, 2010.

[19] Applegate, D. L.; Bixby, R. E.; Chvátal, V.; Cook, W. J. The traveling salesman problem: a computational study, 2nd ed., *Princeton Series in Applied Mathematics*, Princeton Univ Press, 2007.

[20] Reinelt, G. TSPLIB: A traveling salesman problem library. *ORSA Journal on Computing*, 3(4): 376-384, 1991.

[21] López-Ibáñez, M.; Dubois-Lacoste, J.; Cáceres, L. P.; Birattari, M.; Stützle, T. The irace package: Iterated racing for automatic algorithm configuration. *Oper Res Perspectives* 3, 43-58, 2016.

Evaluation of Objective Function Designs Through an Auxiliary Heuristic Scheme

Li Lei, Raymond S. K. Kwan

University of Leeds
Leeds LS2 9JT, West Yorkshire, United Kingdom
{scll, r.s.kwan}@leeds.ac.uk

Abstract

Exact integer linear programming (ILP) models and their solvers have the advantage of delivering optimal results for complex real life scheduling problems. However, they are often computationally practical only for relatively small problem instances. Furthermore, it is not easy to establish high confidence in the effectiveness of an objective function design, which is vital for yielding good-quality schedules in practice. For example, suppose a train unit schedule needs 50 train units to cover a timetable. The workings of individual train units cannot be all efficient because of the timing connections of the trips served. The objective function is therefore a tradeoff amongst a number of schedule quality aspects. For train unit scheduling, an exact ILP solver called RS-Opt has been developed based on a multi-commodity network flow model. An iterative heuristic wrapper around RS-Opt called SLIM has also been developed for large problem instances. The SLIM heuristic aims at shrinking the graph size of the network flow model to as small as possible. An optimal graph for SLIM is a sub-graph of the original full graph for RS-Opt in which all the arcs are required in an optimal solution. At convergence, SLIM would have yielded a minimal graph required by RS-Opt to produce the optimal solution. Before convergence, the reduced graph at each iteration is sub-optimal but is of a very small size such that RS-Opt can be executed very quickly. The exact solution of the sub-optimal graph is used as a basis to derive the reduced graph for the next iteration.

In experiments where RS-Opt alone can find the optimal solution $\mathcal{S}$ without using SLIM, the percentages of graph arcs in $\mathcal{S}$ that are also present in the reduced graphs derived in SLIM iterations have been compared. It was observed that such percentages may not be high even when the iterations were leveling off with objective values quite close to the optimal objective value yielded by RS-Opt. Although the branch-and-price process in the RS-Opt solver may have influenced the graph arc selection, the design of the objective function may have contributed to the observed behavior and alternative designs may have to be considered. This paper reports on further investigations focusing on alternative objective function designs for RS-Opt, the SLIM graph reduction heuristic, and in particular testing the use of the SLIM heuristic scheme as a new methodology for enhancing logical and domain expert reasoning for the evaluation of objective function designs.

1 Introduction on RS-Opt and SLIM

The exact ILP solver RS-Opt for the train unit scheduling problem (TUSP), where train units are assigned to cover trips with respect to passenger demands, is driven by a pre-generated directed acyclic graph (DAG, $\mathcal{G} = (\mathcal{N}, \mathcal{A})$) [4, 5]. The different unit types are defined as multi-commodities. $\mathcal{N} = N \cup \{0, \infty\}$ is the node set where 0 and ∞ are the source and sink respectively, and N is the trip node set. $\mathcal{A} = A_0 \cup A_\infty \cup A$ presents the arc set where $A = \{(i, j)|i, j \in N\}$ is the trip-to-trip arc set, and $A_0 = \{(0, i)|i \in N\}$ and $A_\infty = \{(i, \infty)|i \in N\}$ are the sign-on/off arc set respectively. While $\mathcal{N}$ is fixed for a problem instance, $\mathcal{A}$ represents potential connections and therefore its size could be very large. A path p represents a *unit diagram* defining the trip serving sequence from source to sink of a corresponding unit. To extend the network flow model to prevent blockages within stations, an iterative approach of incrementally inserting cuts against station level conflicts identified in RS-Opt is proposed in [2, 3]. The optimizing result from RS-Opt is a set of unit diagrams for one-day workload of every unit.

Experiments have found the exact solver RS-Opt struggle to solve large problem instances (more than 500 trips). Considering that the complexity is caused by huge arc combinations in the original graph $\mathcal{G}$ and the optimizing result is a subgraph of $\mathcal{G}$, the heuristic wrapper SLIM is therefore designed to reduce the graph

size with a built-in arc controller governed by a set of heuristics [1]. Instead of considering the entire DAG as input, SLIM iteratively runs RS-Opt with a reduced graph $\mathcal{G}'$ yielding a local optimal solution $\mathcal{G}^*$ in which an improved solution may have been obtained based on the objective function value. A new $\mathcal{G}'$ will be formed for the next iteration by the heuristic arc controller based on $\mathcal{G}^*$. The arc controller aims at constructing a reduced graph which is sufficient for RS-Opt to produce a near optimal solution that is very close to the solution produced by RS-Opt in terms of objective function value and subgraph.

2 Discussion of experiments

The convergence of SLIM relies on an objective function that indicates whether the given reduced graph contains a better solution or not compared to the last iteration. Through our experimental observation, SLIM and RS-Opt can usually turn out similar objective function values, but the solution arc sets are dissimilar. For some dataset, e.g. GWR-EMU, the overlapping arc percentages are approximately 40 %. The observation from SLIM iterations shows that many sub-optimal solutions are yielded with similar and sometimes same objective function value but with very different subgraphs. The most likely reason could be that the objective function is not sensitive enough to differentiate the qualities of different sets of arc selection. This fails SLIM to distinguish one subgraph from another. Besides, considering that SLIM only works on size limited reduced graphs but RS-Opt itself works on the entire graph, SLIM may not be able to improve the objective function value for many iterations and hardly converge to an arc set that is highly overlapped with the solution found by RS-Opt solely.

For the exact ILP solvers, the optimal solution is delivered for a certain objective function. A sophisticated objective function is vital to generate high-quality solutions in practice. However, finding an objective function that is capable of separating similar solutions is not trivial. The framework of SLIM can assess the sensitivity of an objective function. This is expected to significantly distinguish one solution from another solution that is generated using SLIM iterations. Ideally, a good-quality objective function is able to converge SLIM to a subgraph which is highly overlapped with the subgraph produced solely by RS-Opt. The design of some alternative objective functions will be discussed in the next section. Testing the use of the SLIM heuristic scheme as a new methodology for enhancing logical and domain expert reasoning for the evaluation of objective function designs will be presented at the conference.

3 Alternative objective function designs

The incumbent objective function sums weighted multiple optimizing targets, for instance fleet size, mileage, coupling/decoupling activities, etc., which makes it hard to observe how each component affects the entire solving process. Some areas of potential improvements for the current objective function have been identified, e.g. (1) A short-slack-time arc may cost more compared to a long-slack-time arc since the arc cost is formed by fixed arc type and mileage cost. Despite the aforementioned, short arcs are preferred in real-world day time schedule except for some special cases such as maintenance. (2) No unit type preference is featured. This leads RS-Opt to use more higher capacity units. (3) Certain aspects of the solution, e.g. fleet size, have been featured in more than one of the objective function terms and the overall effect could be better controlled. (4) Trading off between fleet size optimization and coupling/decoupling activities is hard to balance.

From the perspective of real-world scheduling, there may be more optimization goals. First, for a timetable that is served by only one single unit type, the unit number should play the most important role in the objective function. Second, when multiple unit types are involved, the operators may care more about the balance among different unit types according to what they possess currently. Finally, instead of minimizing the number of units, some schedulers may prefer to optimize the number of coaches used to cover all the trips, or the total seat over-provision on the railway network and so on.

An objective function is difficult to satisfy all real-world optimization perspectives. The analysis of

real-world scheduling preferences can help to design a pool of alternative objective functions and their sensitivity in terms of differentiating subgraphs can be tested by SLIM. A more sensible arc cost strategy can be constructed by extracting mileage cost from the incumbent arc cost and considering slack-time as the superior arc cost. A rough mileage can be pre-calculated and considered as a soft constraint to satisfy. Besides, hard constraints of lower bounds for every unit type can be set by experienced schedulers. This allows more flexibility in balancing the usage of different unit types. In practice, fleet size usually plays the most important role for operators followed by other targets. A two level optimizing strategy can be applied. The first stage optimizes the fleet-size oriented alternative objective functions, i.e., the number of units under different circumstances. The second stage optimizes sub-targets with respect to the first-stage result, i.e., mileage, compact unit diagram, etc.

4 Remarks and ongoing research

This paper presents the potential of using the SLIM framework to assess objective function designs and derive highly overlapped subgraphs with similar objective values using RS-Opt and SLIM. Some alternative objective functions can be designed based on the incumbent objective function and real-world expectations. Once an objective function is proved as sensitive enough to differentiate solutions, SLIM can be applied to tackle large instances. Another benefit obtained from this study is the possibility to construct some new heuristics for the arc controlling process, for instance an arc cost based heuristic method. The setup and configuration for testing and refining the models, in particular the implementation of new objective functions and constraints in RS-Opt, are a very substantial ongoing task, which has to be carefully analyzed with many real datasets. More details and results will be reported at the conference.

Acknowledgements Li Lei is funded by a Ph.D. grant of the School of Computing, University of Leeds. And we would like to thank First Group, and Tracsis Plc for their kind and helpful collaboration.

References

[1] P Copado-Mendez, Zhiyuan Lin, and R Kwan. Size limited iterative method (SLIM) for train unit scheduling. In *Proceedings of the 12th Metaheuristics International Conference*. Barcelona, Spain, 2017.

[2] L Lei, R Kwan, Z Lin, and P J Copado-Mendez. Station level refinement of train unit network flow schedules. In *8th International Conference on Computational Logistics*. Southampton, UK, 2017.

[3] L Lei, R Kwan, Z Lin, and P J Copado-Mendez. Resolution of Station Level Constraints in Train Unit Scheduling. In *14th International Conference on Advanced Systems in Public Transport and TransitData*. Brisbane, Australia, 2018.

[4] Zhiyuan Lin and Raymond S K Kwan. An integer fixed-charge multicommodity flow (FCMF) model for train unit scheduling. *Electronic Notes in Discrete Mathematics*, 41:165–172, 2013.

[5] Zhiyuan Lin and Raymond S K Kwan. A branch-and-price approach for solving the train unit scheduling problem. *Transportation Research Part B: Methodological*, 94:97–120, 2016.

Deterministic Multi-Objective Fractal Decomposition Algorithm

Léo Souquet[1,2], Amir Nakib[1], El Ghazali Talbi[3]

[1] Université Paris-Est, Laboratoire LISSI, 122 Rue Paul Armangot, 94400
Vitry sur Seine, France
nakib@u-pec.fr

[2] Data ScienceTech Institute, DSTI Labs,
950 Route des Colles, Les Templiers, 061410 Biot, France
leo@dsti.co

[3] CRISTAL UMR CNRS 9189 and INRIA Lille Nord Europe,
Parc Scientifique de la Haute Borne 4 avenue Halley Bt.A, Park Plaza
59650 Villeneuve d'Ascq, France
el-ghazali.talbi@univ-lille1.fr

Abstract

This paper presents a new deterministic multiobjective optimization called "Multiobjective Fractal Decomposition Algorithm" (Mo-FDA). The original FDA was designed for mono-objective large-scale continuous optimization problems. It is based on a "divide-and-conquer" strategy and a geometric fractal decomposition of the search space using hyperspheres. In our work, a scalarization approach is used to deal with MO problems. The performance of Mo-FDA was compared to state of the art algorithms from the literature on classical benchmark of multi-objective optimization.

1 Introduction

In multiobjective optimization problems (MOP) the goal is to optimize at least two objective functions. This paper deals with these problems by using a new decomposition-based algorithm called: "Fractal geometric decomposition base algorithm" (FDA). It is a deterministic metaheuristic developed to solve large-scale continuous optimization problems [5]. We defined large scale problems by the problems having dimension greater than 1000. In this article, we are interested in using FDA to deal with MOPs because in the literature decomposition based algorithms have been successfully applied to solve these problems. Mainly FDA decomposes the variable decision space of a single objective problem into a set of hyperspheres (regions of the search space) and each is evaluated. In other terms, FDA is based on "divide-and-conquer" paradigm where the sub-regions are hyperspheres rather than hypercubes on classical approaches. In order to identify the Pareto optimal solutions, we propose to extend FDA using the scalarization approach. We called the proposed algorithm Mo-FDA.

The rest of the of paper is organized as follow. The next Section 2 presents a description of the proposed algorithm. In Section 3 the obtained results and a comparison to the competing methods are presented. Finally, Section 4 presents the future work.

2 Proposed Mo-FDA

Fractal Decomposition Algorithm uses a "Divide-and-Conquer" strategy across the search space to find the global optimum, when it exists, or the best local optimum. The hypercubes are the most used forms in the literature. However, this geometrical form is not adapted to solve large scale problems or high dimensional problems because the number of vertices increases exponentially. FDA [5] uses hyperspheres to divide the search domain as this geometrical object scales well as the dimension increases allowing FDA to solve large scale problems. In addition, the fractal aspect of FDA is a reference to the fact that the search domain is decomposed using the same pattern at each level until the maximum fractal depth k (fixed by the user). While searching for the optimum, FDA uses 3 phases: initialization phase; 1)

exploration phase, 2) and exploitation phase 3). During the initialization phase, at level 0, the current hypersphere is decomposed into $2 \times D$ sub-hyperspheres with D being the problem's dimension.

Once the initialization phase is completed, FDA starts the exploration phase to identify the sub-hypersphere that potentially contains the global optimum or best local optimum (or global optimum if it is known), to decompose it again in $2 \times D$ sub-hyperspheres. This operation is repeated until the maximum fractal depth, k is reached. k has been experimentally determined and set to 5. When the $k - level$ is reached, FDA enters in the exploitation phase. The aim of this phase is to find the best local optimum inside the current sub-hyperspheres. This procedure is called *Intensification local search (ILS)*. Each instance of ILS starts at the center of the sub-hypersphere being exploited. This local search is moving along each dimension, evaluating three points on each one and only the best is considered for the following dimension (more details are in [5]). Then, the second *k-level sub-hypersphere* is exploited using ILS. This process will stop when the maximum number of evaluations is reached. If all *k-level sub-hypersphere* have been exploited without FDA stopping, it backtracks to decompose the second best hypersphere at the level $k - 1$.

In a multi-objective problem, different objective functions are being optimized at the same time. Scalarization techniques allows to combine the different objective functions into one, allowing the approach to solve it as a mono-objective problem. Different scalarization methods were proposed such as Weighted Sum and Weighted Tchebycheff Method [4]. In this work, Tchebycheff approach were considered for Mo-FDA and is defined in equation 1:

$$\begin{array}{rl} Minimize & \max\limits_{i=1,\ldots,k} \quad [\omega_i(f_i(x) - z_i^*)] \\ \text{Subject to} & x \in X \end{array} \tag{1}$$

with k the number of objective functions to optimise and z_i^* the optimum of function f_i. In addition, the sum of weights ω_i must be equal to 1. In Mo-FDA, n different mono-objective problems are solved with different combination of weights ω_i. As each combination produces one solution, Mo-FDA produces a Pareto-Front composed of n points. To improve the speed at which Mo-FDA solves a multi-objective problem, the n independent instances are launched simultaneously using containers. The overhead produced by the different containers' management can be neglected compared to the benefit from the parallel implementation of the n instances. Furthermore, one can see that the algorithm can benefit from running on a multi-node environment without having to change the implementation.

3 Results and Discussion

In order to test the performance of Mo-FDA a set of 8 functions, 5 from the ZDT family problems [2] and 3 from the DTLZ sets [9] have been used. The results obtained were compared to the well regarded algorithms NSGA-II, NSGA-III and MEOA/D as well as state-of-the-art approaches GWASFGA [8], and CDG [1]. To conduct the different experiments, jMetal 5.0 [6], a popular Java-based framework in the literature has been used. The principal experiments settings described in [3].

In the context of multi-objective problems, many metrics can be used as discussed in [7]. In order to have a better overview of the performances of Mo-FDA compared to other algorithms, we have selected four different metrics. The first one, the Hypervolume metric. It measures the size of the portion of the objective space that is dominated by an approximation set. The Generational Distance metric (GD) computes the average distance from a set of solutions obtained by an algorithm to the true Pareto-Front. The Inverted generational distance (IGD), measures both convergence and diversity by computing the distance from each point known in the true Pareto-Front to each point of a set of solutions found by the executed algorithm. The Spread metric measures the extent of the spread achieved among the obtained solutions. It is important to notice that the goal is to maximize the first metric to minimize the others.

Moreover, to compare the results obtained by the different algorithms we used the Friedman Rank sum method and the obtained results are presented in Table 1. One can see that Mo-FDA is the most efficient algorithm among three metrics out of four. Looking at the other algorithms, MEOAD/D is efficient

Algorithms	Mo-FDA	NSGA-II	NSGA-III	MEOA/D	GWASGFA	CDG
Hypervolume	1.875 (1)	2.25 (2)	5.625 (6)	2.625 (3)	4.125 (4)	4.25 (5)
GD	2.25 (1)	2.875 (2)	4.875 (6)	3.25 (4)	3.125 (3)	4.625 (5)
IGD	2.125 (2)	2.25 (3)	5.5 (6)	2 (1)	4.25 (4)	4.875 (5)
Spread	1.75 (1)	3.5 (3)	4.25 (4)	1.75 (1)	4.25 (4)	5.5 (6)

Table 1: Ranking using Friedman Rank sum (and their global rank) of all algorithms on the different metrics for all tested functions.

on the IGD and Spread but not on the GD and the Hypervolume. Consequently, the importance of using multiple criteria highlights strengths and weakness of each algorithm. On average, the computation time required to solve one function at dimension *D=30* is 0.8 seconds. The machine used for experimentations has the following characteristics: a processor Intel Xeon Platinum 8000 with 144GB of RAM.

4 Conclusion and future work

In conclusion, Mo-FDA was tested on 8 different functions and compared to 5 other well regarded and state-of-the-art metaheuristics. Its performances to find good Pareto-front is proved using four popular metrics in the literature. For future work, we aim to adapt Mo-FDA to many-objectives problems as well as real world applications.

References

[1] X. Cai, Z. Mei, Z. Fan, and Q. Zhang. A constrained decomposition approach with grids for evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 22(4):564–577, 2018.

[2] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. *Scalable Test Problems for Evolutionary Multiobjective Optimization*, pages 105–145. Springer London, London, 2005.

[3] S. Jiang, J. Zhang, and Y. Ong. Multiobjective optimization based on reputation. *Information Sciences*, 286:125 – 146, 2014.

[4] K. Miettinen, F. Ruiz, and A. P. Wierzbicki. *Introduction to Multiobjective Optimization: Interactive Approaches*, pages 27–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[5] A Nakib, S Ouchraa, N Shvai, L Souquet, and EG Talbi. Deterministic metaheuristic based on fractal decomposition for large-scale optimization. *Applied Soft Computing*, 61:468 – 485, 2017.

[6] A. Nebro, J. Durillo, and M. Vergne. Redesigning the jmetal multi-objective optimization framework. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pages 1093–1100, New York, NY, USA, 2015. ACM.

[7] N. Riquelme, C. Von Lcken, and B. Baran. Performance metrics in multi-objective optimization. In *2015 Latin American Computing Conference (CLEI)*, pages 1–11, Oct 2015.

[8] R. Saborido, A.B. Ruiz, and M. Luque. Global WASF-GA: An evolutionary algorithm in multiobjective optimization to approximate the whole pareto optimal front. *Evolutionary Computation*, 25(2):309–349, 2017.

[9] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8:173–95, 02 2000.

A task planning problem in a home care business

Isabel Méndez-Fernandez, Silvia Lorenzo-Freire, Ignacio García-Jurado, Luisa Carpente, Julián Costa

Grupo MODES
Departamento de Matemticas, Universidade da Coruña, Spain
Emails: isabel.mendez.fernandez@udc.es, slorenzo@udc.es, igjurado@udc.es, luisacar@udc.es, julian.costa@udc.es

Abstract

In this work we present a task planning problem for a home care business. The company has a set of nurses in charge of visiting the users' homes and it wants to solve the problem that consists of scheduling the nurses working days in order to correctly attend all the clients. The real cases the company faces are of great size and cannot be solved in an exact way, and therefore we propose a heuristic algorithm to deal with the issues (like the registration of an user) the company addresses, in short computational times. In order to validate the behaviour of the heuristic algorithm we use it to solve a battery of examples, both small-sized instances and real-like cases.

1 Introduction

Home health care is a resource that allows elderly and/or dependent people to continue living at their homes despite being in a situation of dependency, or in need of assistance to carry out different daily tasks. The company we are working with has been providing home care services since 1997, both in the city of A Coruña and in the neighboring municipalities. This company faces a scheduling problem, in which we have to determinate the routes the employees must follow and set the exact time at which the caregivers must perform the services assigned to them.

1.1 The problem

The users of the service provided by the company are those who need health care support or require help in carrying out certain tasks and, to improve their quality of life, hire the company's services. These users must specify the number of services they require, detailing the day in which they must be completed, taking into account that a user can only require up to four services each day. They should also indicate the available (when the service must be completed) and optimal (when the user prefers to be attended, even though the fulfilment of this condition is optional) time windows of each service.

To provide its services the company has a set of caregivers who are in charge of visiting users' homes (at the time they have been instructed to do so) and carrying out the tasks assigned to them (for example: cooking meals, monitoring medication, cleaning chores, etc.). The number of available caregivers may increase or decrease depending on the volume of work to complete, which is the number of active users and the services they require. In order to maintain users' satisfaction, for each caregiver and user, the company sets a level of affinity that establishes how suitable the caregiver is to attend the user.

Therefore, the company must carry out a work plan where, for each caregiver, the services to be carried out are specified (establishing the tasks to be performed in each of them) and the schedule in which those services must be completed (always upholding the available time windows established by the users). This work plan arranges the whole week, so the schedule is repeated over time until the need to modify it arises. In order to carry out this task, the company has a number of supervisors who are in charge of manually managing the working days of the caregivers. Each supervisor works with a set of users and caregivers, according to the areas or neighbourhoods she operates in.

The company wants to obtain a software tool that automatically modifies the caregivers' previous plan, in order to solve the following issues:

- Registration of a user.

- Discharging a user.
- Increasing the number of services required by a user.
- Decreasing the number of services required by a user.
- Alteration of any of the parameters of a service.
- Repetition and/or combination of the previous incidents.

The new schedules, obtained to solve the abovementioned incidents, must be achieved considering the next objectives:

1. The new plans should minimize time lost between services.
2. The new plans should maximize the affinity levels between users and the caregivers who visit them.
3. We must try not to change the previous planning in excess.
4. The new plans should uphold, as much as possible, the optimal time windows of the services.

The purpose of this work is to provide the company, employing different operational research techniques, with a software tool that allows them to automatically modify the caregivers' schedules, in order to solve the considered incidents.

2 Problem solution

We have modelled the problem presented by the company as an integer linear programming problem, but for real instances, such as those faced daily in the company, this problem presents a large number of variables and constraints. Therefore, the integer linear programming problem cannot be used to obtain the optimal schedule of real examples.

Because of this, the task planning problem must be solved in an approximated way, through the use of heuristic techniques, in order to obtain admissible solutions in short computational times.

2.1 Algorithm

The algorithm designed to solve the company's planning problem is based on the simulated annealing method, introduced in [1], it consists of re-scheduling the services and optimizing the working days of the caregivers, and it intends to solve the incidents the company works with, avoiding excessive modification of the previous planning. The aim followed is to make the work plans feasible while trying to ensure that the caregivers have no gaps in their working hours. The most important steps of the algorithm are presented below:

Preparation of elements The algorithm starts with an initial solution, which is the previous schedule of the caregivers. The services that need to be removed from this solution are deleted (users' discharge, reduction of services) and, at the same time, we select the services that need to be rescheduled (modification of parameters) and the ones that need to be included in the work plans (users' registration, increasing of services).

Service planning First, each of the services that must be scheduled is assigned to the best available caregiver. To establish how suitable a caregiver is to assist a user, we arrange them on a list considering the following aspects: (i) the affinity level between them, (ii) the difference between the hours worked by the caregiver and those specified in her contract, and (iii) the travel time between that user's home and all the clients the caregiver is already attending. After that, the service is scheduled in such a way that it induces the least possible amount of overlap and break

gaps (between the new service and the ones already assigned to the considered caregiver). In some cases, we do not have services to plan, so the algorithm goes directly to the optimization phase.

Optimization After assigning each service to a caregiver, the schedule is optimized, using the simulated annealing method, to eliminate possible overlaps or breaks in the schedule. If, after this optimization, there are any overlaps in the schedule, the considered service must be assigned to the next best available caregiver. In the cases where there are no services to be scheduled, the caregivers' plan is optimized in an attempt to reduce the possible break gaps in it (note that in such circumstances there is no overlap between services).

End When all the services have been correctly scheduled, the overlaps have been eliminated and the breaks have been optimized, the algorithm ends, returning the optimized planning as the new solution.

The movements we applied during the implementation of the simulated annealing method are: rescheduling one service, rescheduling several services at once, exchanging the time at which two selected services are carried out, exchanging the caregivers that conduct two selected services, and changing the caregiver that performs a service.

The objective function considered to evaluate the schedules obtained in the simulated annealing method is a lexicographic function, in which we want to minimize the following elements: (i) the overlap time between services, (ii) the time lost between services, and (iii) the time that the services are being carried out outside their optimal time windows.

2.2 Numerical results

To check the algorithm's performance we designed a battery of 50 small-sized examples, that can be solved optimally though the integer programming problem, and we compare the optimal solutions with those given by the heuristic algorithm. The collected results show that our algorithm provides solutions analogous to the optimal ones, needing significantly less computational times.

We also designed a series of real-like examples, in which we take into account all the different incidents that the company deals with, and we solve them using the heuristic algorithm. Based on the schedules reached, we can conclude that the algorithm's behaviour is acceptable, since it correctly solves all the incidents considered while employing low computational times.

3 Conclusions

In this work, we analyse the scheduling problem presented by the company and we design a resolution method based on the heuristic technique of the simulated annealing. The purpose of the algorithm is to solve the incidents considered by the company, trying to modify the original schedules as little as possible.

The good performance of the algorithm is checked by solving first, a battery of small-sized instances (in order to compare the solutions obtained with the optimal ones) and second a set of real-like instances (in order to test if the algorithm solves the incidents addressed by the company). In both cases the results show that the heuristic algorithm exhibits a good performance.

References

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

A solution approach to the multi activity combined timetabling and crew scheduling problem considering a heterogeneous workforce with hierarchical skills and stochastic travel times

Brian A. Benavides [1], Diego R. Vásquez[2], David Barrera[3], Carlos E. Montoya[4],

[1] Pontificia Universidad Javeriana
Calle 40 No 5-50, Ed. José Gabriel Maldonado S.J. Piso, Bogotá D.C., Colombia
c_montoya@javeriana.edu.co

Abstract

This paper introduces an extension to the Multi-Activity Combined Timetabling and Crew Scheduling problem which considers heterogeneous personnel with hierarchical skills, multiple work shifts and stochastic travel times. The goal of this problem is to define the number of workers and to stablish a schedule that fulfills customer's requirements in order to minimize a total cost objective function. As a solution approach, a Genetic Algorithm (GA) was proposed for the deterministic version of the problem. Thereafter, a sim-heuristic was developed for solving the problem with stochastic travel times. The algorithms were tested on randomly generated instances. Obtained results show a good performance of the GA for the deterministic version of the problem. On the other hand, it were validated the benefits in terms of services level of considering an approach that combines simulation and optimization for the solving stochastic version of the problem.

1 Introduction

The Multi-Activity Combined Timetabling and Crew Scheduling Problem (MCTCSP) consists of visiting a set of customers, minimizing the number of resources involved and balancing the workload. This problem was solved by Barrera et al. [1], in the context of health promotion and prevention programs in children's schools. In this work, authors argue about the NP-Hard nature of the problem, since it combines features of the Crew Scheduling Problem (CSP) and the Timetabling Problem (TTP), which both have been classified as N-Hard [1].

This paper addresses an extension of the MCTCSP that considers: (i) heterogeneous workers with hierarchical skills, which implies that workers could have different levels of expertise for a single skill; ii) service requirements for different shifts (Morning, Afternoon and Night); and iii) stochastic travel times between customers locations. The resulting problem from including the mentioned extensions in the MCTCSP, consists on finding the amount of workers of type k, $k \in K$ and to schedule such workers, according to certain legal constraints, in order to fulfill customers requirements. Each customer$i$, $i \in \mathcal{C}$ demands a set of services $\mathcal{S}_i$, from a portfolio of services $\mathcal{S}$, $\mathcal{S}_i \subset \mathcal{S}$ within a set of working hours $\mathcal{H}$ and a set of days D. Each type or worker k has a predefined salary and is able to provide certain services; where the corresponding service times could vary depending on his level of expertise (beginner, average, expert). Each customer is characterized by a geographical location and defines a set of desirable schedules $\mathcal{N}_i$ where his required services could be provided. Each worker takes a certain time to move between clients, according to their specific geographical location. To satisfy customers' demands there is a set of available workers $\mathcal{T}_k, \mathcal{T}_k \subset \mathcal{T}$. The purpose of this problem is to build a set of feasible lines of work (routes) that minimizes the amount of salaries paid in order to guarantee satisfaction of demand during a planning horizon.

Regarding the original MCTCSP, Barrera et al. [1] proposed a mixed integer programming model and a two phase heuristic. Thereafter, Novoa et al. [2] introduced a GRASP-based approach for the MCTCSP with heterogonous workforce. Concerning the consideration of hierarchical skills, heterogeneous workforce and multiple skills, different solution methods such as goal programming, integer programming models, and genetic algorithms, have been explored for related personnel scheduling problems. Moreover, the feature of stochastic travel times, have been mainly considered in vehicle routing problems.

As can be inferred, this paper introduces a complex problem which have not been explored before,

since it takes into account several features that has not been considered altogether in the MCTCSP. In order to solve the problem at hand, first, the deterministic version of the problem was addressed by means of a Genetic Algorithm (GA). Thereafter, a sim-heuristic was developed for solving the problem with stochastic travel times. Different variability scenarios were tested for evaluating the impact on service levels and costs. The algorithms were tested on randomly generated instances. Obtained results show a good performance of the GA for the deterministic version of the problem. On the other hand, it were validated the benefits in terms of services level of considering an approach that combines simulation and optimization for the stochastic version of the problem.

2 Solution Approach

In order to solve the problem at hand, first a solution approach is proposed for the deterministic case, where travel times between nodes are fixed. Subsequently, a solution method that combines simulation and optimization is implemented for solving the stochastic version of the problem, where travel times between nodes are uncertain. All the proposed solution methods are described as follows.

2.1 Solution technique for the deterministic version of the problem

The solution for the deterministic version of the problem is based on a genetic algorithm which includes the utilization of a heuristic for ensuring the construction of feasible schedules for workers. For comparison purposes an initial solution is built by means of a constructive procedure. This initial solution is also included in the initial population of the GA.

2.1.1 Initial Solution

The proposed procedure for the initial solution consists of: (i) identify a subset of type of workers $L, L \subset \mathcal{T}$, which is composed by those type of workers, that are mandatory for the fulfillment of certain services; (ii) add a new worker that belongs to the subset L; (iii) build a feasible schedule (route) for the worker added in the previous step (this implies assigning such a worker for providing services for certain customers in certain hours and days during the planning horizon according to certain legal constraints); (iv) once there are not services that can only be fulfilled for the type of workers in subset L, add a worker that belongs to the worker type with the lowest salary; (v) build a feasible schedule (route) for the worker added in the previous step. Finally, steps (iv) and (v) are repeated until all services demands are fulfilled.

2.1.2 Genetic Algorithm

Regarding the GA, first it was necessary to define the chromosome structure, in which, each gene corresponds to the number of workers of a given type k that could be assigned for fulfilling customers' demands. The number of workers stablished in each gene should be lower or equal to the maximum number of workers available of its corresponding type. Once a chromosome is generated, an assignment heuristic is used for building feasible routes. This procedure defines an order in which the different types of workers should be scheduled. For this purpose, the next priority criterion U_k is calculated for each type of worker k:

$$U_k = \frac{\#\ of\ services\ that\ could\ be\ provided\ by\ worker\ k}{Wage\ of\ worker\ k} \quad (1)$$

Considering this criterion, a single worker h that belongs to the worker type with the largest value of U_k is selected to be scheduled. Once, a feasible route for h is build, the next worker with the largest value of U_k is selected to be scheduled. This last procedure is repeated until all customers' demands are satisfied. Considering that it is possible to fulfill all requirements and constraints without using all the workers defined in each gene, this values are adjusted according to the amount of scheduled workers of each type. Finally, the objective function is calculated according to the salaries of the workers that were scheduled after applying the heuristic procedure.

After stablishing the structure of the chromosome, the initial population was build. According to the work of Göçken et al. [, M. Yaktubay and F. Kiliç [22] a set of 200 chromosomes were generated. One

chromosome was obtained by means of the initial solution previously explained, while the remaining ones were created by randomly generating the number of workers of its respective genes. Moreover, the classic roulette method and single crossover operator were applied for generating new chromosomes.

2.2 Solution technique for the stochastic version of the problem

In the stochastic version of the problem, travel times between a pair customers i, j are randomly generated with a normal distribution function with a given mean $\mu_{i,j}$ and $\sigma_{i,j}$. For solving this problem, first, the GA is used by considering the expected values of the normal distribution ($\mu_{i,j}$) as travel times between clients. Thereafter, the chromosome that obtained the best objective function is fixed. Afterwards, the proposed sim-heuristic focuses on rescheduling workers in order to minimize the average tardiness (in hours) per service required. In a single iteration of the sim-heuristic, the average tardiness of the current schedules is calculated for 100 scenarios, where in each scenario travel times were randomly generated. Subsequently, a local search procedure is proposed for rescheduling the previous assignment of workers in order to minimize the average tardiness. Each time a new schedule is obtained by means of the local search, the average tardiness is calculated considering the 100 scenarios. The sim-heuristic stops until reaching a given number of iterations without improving the average tardiness.

3 Results and Conclusions

Regarding the deterministic version of the problem, 51 instances were randomly generated for validating the impact of the proposed solution approach. These instances considered between 30 and 100 number of services within a planning horizon of 7 days. Obtained results with the GA where compared against the initial solution and the ones obtained with an adaptation of the Grasp procedure proposed by Novoa et al. [2]. After running the 51 instances, it was possible to obtain an average improvement of 25% and 7% of the objective function with the GA in comparison to the proposed initial solution and the GRASP respectively. In addition, when comparing against the solutions of the mixed integer programming model which was also implemented, found results shows that the GA was able to find the optimal solution for the 7 instances in which the solver was able to find an optimal solution within one hour of execution. For the remaining 44 instances the GA was able to reach an improvement of 25% of the best objective function obtained with the solver within a time limit of one hour.

Regarding the stochastic version of the problem, results were evaluated in 45 instances, which considered between 30 and 100 nodes respectively. Solutions found with the proposed sim-heuristic were compared with the ones obtained with the GA when fixing the expected values of the normal distribution as the travel times. Solutions obtained with both approaches were validated in 100 scenarios where in each scenario travel times were randomly generated. As a result, the sim-heuristic was able to reach an average reduction of 0,17 hours of the average tardiness per service required in comparison to the one found with GA in the first place.

References

[1] D. Barrera, N. Velasco, and C. A. Amaya. A network-based approach to the multi-activity combined timetabling and crew scheduling problem: Workforce scheduling for public health policy implementation. *Comput. Ind. Eng.*, vol. 63, no. 4, pp. 802–812, 2012.

[2] D. Novoa, C. Olarte, D. Barrera, and E. M. González-Neira. A GRASP-based approach to the multi activity combined timetabling and crew scheduling problem considering a heterogeneous workforce. *Int. J. Ind. Eng. Comput.*, vol. 7, no. 4, pp. 597–606, 2016.

A Scalable Method to solve the Call Center Staffing with Service-Level Agreement under Uncertainty

Renzo Benavente[1], Julio C. Casas[2], Susara Van Den Heever[3], Gianmaria Leo[3], Victor J. Terpstra[3]

[1] Pontificia Universidad Católica del Perú
Av. Universitaria 1801, San Miguel 15088, Peru
renzo.benavente@pucp.pe

[2] IBM Global Markets Cloud Sales
Av. Javier Prado Este 6230, La Molina 15024, Peru
jcasas@pe.ibm.com

[3] IBM Data Science Elite
590 Madison Avenue, New York, NY 10022, United States
svdheever@fr.ibm.com, gianmaria.leo@ibm.com, vterpstra@us.ibm.com

Abstract

The call center staffing is a complex planning problem that many firms deal with. The entire decision-making process usually aims to achieve the right trade-off between cost efficiency and a proper workforce planning that ensures the committed service level. This practice turns out to be challenging due to the presence of relevant stochastic factors impacting decisions, like arrival and duration of calls, waiting time and abandonment. Another important challenge arises from the restricted lead time to deliver the staff plans, whereas common solution approaches are usually computationally intensive or require exhaustive testing to validate assumptions and parameters. Our work focuses on a call center line managed by a major Bank of Peru. We introduced an effective simheuristic leading to a scalable framework that handles configurable assumptions. We validated our method by using a benchmark tool adopted by the Bank, and we compared it with a more typical approach adopted in production. The new method provides comparable solutions in terms of quality and accuracy, by reducing the computational time from hours to few minutes.

1 Introduction

This work focuses on a call center line managed by a major Bank of Peru. The business goal is to minimize the total cost of the workforce, by ensuring that staffing and shift scheduling meet the committed Service Level Agreement (SLA). The literature widely addresses call center problems, see [3, 4]. Authors of [11] present a stochastic programming approach binding together staffing and shift scheduling, where the queueing system is modeled as the Erlang A: a MIP formulation is introduced and solved with L-shaped decomposition. In [9], authors explain call arrivals with a doubly stochastic Poisson process that combine random arrivals and a random busyness factor, while the service rate follows an exponential distribution. The authors use stochastic and robust optimization techniques, then reformulate the problem as a MIP. In [5], a joint chance constrained stochastic program is introduced, where the forecasting error of call arrivals is assumed to have a continuous probability distribution. The problem is reformulated as a deterministic MIP, which can be effectively solved over 1-week instances. More recently, authors of [2] investigate a two-stage stochastic formulation for a multiclass service system, which they solve with Benders decomposition strengthened by MIR inequalities. Test instances focus on a bank call center: a 1-day planning horizon is set, and call arrivals are assumed to be Poisson random variables with random rate. Similar instances are solved in [10], where a two-stage robust program is presented and solved with a Benders-like method.

Our work investigates a novel simheuristic approach, given the known advantages of these techniques [7] towards the Bank's business needs. The contributions of this paper follow. First, our method allows to configure probability distributions and related parameters without invalidating the approach. In this way, business users and analysts can benefit from increased flexibility when they test hypotheses and tune distribution parameters from the analysis of solutions. Second, the introduced simheuristic scales

over computational resources: a fast method can improve the productivity of business users and allows executives to evaluate more efficiently budget and employment contracts. Third, our method is easy to implement, since it does not rely on sophisticated algorithms, which typically characterize this application. Hence, the implementation of new features, as well as regular troubleshooting and maintenance, can lead to reduced delivery time with improved time-to-market. Finally, the introduced method can be adopted to solve more general staffing problems focusing on multi-server queueing system with SLA constraints. This paper is structured as follows: in Sec.2 we define the optimization problem; in Sec.3 we describe the compared methods; in Sec.4 we summarize computational experiments and results.

2 The Call Center Staffing Problem of the Bank

Let P, $S \subset \mathbb{N}$ be resp. the sets of periods and shifts. Let $A \in \{0,1\}^{|P|\times|S|}$ be a binary matrix such that $a_{ij} = 1$ iff period $i \in P$ is covered by shift $j \in S$. Let c_j, c^{m}, $c^{\mathrm{p}} \in \mathbb{R}_+$ be resp. the cost (or salary) of any agent working in shift $j \in S$, and the costs of any manger and principal over P. Let x_j, x^{m}, $x^{\mathrm{p}} \in \mathbb{Z}_+$ be integer decision variables representing resp. the number of agents to be allocated to shift $j \in S$, and the numbers of managers and principals to be hired. Moreover, let $\mathcal{C}(x, x^{\mathrm{m}}, x^{\mathrm{p}})$ be a polyhedron expressing some staffing rules between agents, managers and principals. Finally, let $y_i \in \mathbb{Z}_+$ be a decision variable representing the number of agents available in each period $i \in P$. Now, more notation has to be introduced to define the SLA. Let $\varphi := (\tau, \lambda, \vartheta, \zeta)$ be such that: τ is the time threshold within calls have to be answered to meet SLA, and λ, ϑ, ζ are tuples of parameters of some (distinct) probability distributions representing resp. arrivals, duration and abandonment of customer calls. Moreover, let d_i be the predicted number of inbound calls in $i \in P$; let $Q \subseteq P$ and $l(Q) \in [0,1]$ be resp. a time window and a targeted SLA value over Q. Given φ, the SLA can be defined as a function $g_\varphi : \mathbb{N} \times \mathbb{Z}_+ \to [0,1]$ of the number of inbound calls and the number of agents available over Q. Furthermore, let $z \in \{0,1\}^{|P|}$ be a decision variable such that $z_i = 1$ only if $g_\varphi(d_i, y_i) \geq l(\{i\})$ for each $i \in P$. Finally, the formulation follows:

$$\min_{\substack{(x,x^{\mathrm{m}},x^{\mathrm{p}})\in\mathcal{C}\cap\mathbb{Z}_+^{|S|+2},\\ y\in\mathbb{Z}_+^{|P|},\, z\in\{0,1\}^{|P|}}} \quad \sum_{j\in S} c_j x_j + \sum_{h=\mathrm{m,p}} c^h x^h \tag{1}$$

$$\sum_{j\in S} a_{ij} x_j \geq y_i \qquad \forall i \in P \tag{2}$$

$$g_\varphi(\sum_{i\in Q} d_i, \sum_{i\in Q} y_i) \geq l(Q) \qquad \forall Q \in \mathcal{Q} \tag{3}$$

$$g_\varphi(d_i, y_i) \geq l(\{i\}) z_i \qquad \forall i \in P \tag{4}$$

$$\sum_{i\in P} z_i \geq \alpha |P| \tag{5}$$

where $\mathcal{Q}$ is a family of time windows and $\alpha \in [0,1]$. Constraint (2) defines the allocation of the agents to the shifts; constraint (3) ensures that the SLA is met over defined observation windows; constraints (4)-(5) together balance the SLA over the entire horizon P. Finally, let us observe that constraints (3)-(4) cannot be expressed in closed form, since g_φ depends on random variables.

3 Methods to solve the problem

The Bank adopts in production a method denoted **HeuP**: the main idea is exploiting simulation to estimate a discrete set of SLA values for some combinations of inbound calls and available agents. The approach generates a finite set of relevant valid SLA inequalities by exploiting an event-drive simulator, similarly to the separation oracle introduced in [1]. More details follow. Let N, $M \subset \mathbb{N}$ be two finite subsets representing the numbers of resp. inbound calls and available agents for some $i \in P$. Given an event-driven simulator $\hat{g}_\varphi$, any SLA value can be estimated empirically, i.e. we can compute a matrix

Method	Cost (PEN)	# Agents	# Managers	# Principals	Score (%)
HeuP	308,935	59	4	2	83.2
HeuN	313,364	60	4	2	80.9

Table 1: Summary of solutions.

Method	Sim time (min)	Opt time (sec)	Opt gap (%)
HeuP	229.91	20.04	4.08
HeuN	10.71	0.16	0.00

Table 2: Summary of computational times.

$B \in [0,1]^{|N|\times|M|}$ such that $b_{nm} := \hat{g}_\varphi(n,m)$ for each $n \in N$, $m \in M$. Now, constraints (3)-(4) can be reformulated as linear inequalities, by using $\hat{g}_\varphi$ and introducing a set of suitable binary variables that bind each relevant staffing level (number of agents) to each SLA value, in any period. In conclusion, **HeuP** consists of two consecutive steps, which are the computation of B and the resolution of a pure MILP program.

The method we introduce is denoted **HeuN**: the main idea is mixing combinatorial optimization and simulation to retrieve deterministic uncertainty sets, which aim to model the minimum number of agents to meet the SLA. Given a number of inbound calls over a time window, **HeuN** generates a finite set of deterministic scenarios representing possible SLA-compliant phone records. In particular, given φ (as defined in Sec. 2), d inbound calls and a targeted SLA value $\hat{l}$, any deterministic scenario $\sigma := (\varphi, d, \hat{l})$ is a set of $d' \sim d$ time intervals representing attended calls, such that $\hat{l}$ is met under the assumptions in φ. Let $n(\sigma)$ be the minimum number of agents to meet the SLA in σ, hence $n(\sigma)$ can be computed in polynomial time[1]. Repeating the generation of σ and the computation of $n(\sigma)$ for k times, the deterministic uncertainty set $U(\sigma,k) \in \mathbb{N}^k$ can be obtained by $(n(\sigma))_{h=1,\dots,k}$. Finally, the original formulation can be reduced to a Set Covering Problem by removing constraints (3)-(5) and fixing $y_i = \lfloor E[U(\sigma,k)] \rceil$ for each $i \in P$.

4 Implementation Notes and Summary of Computational Experience

Both the methods are characterized by two relevant phases: simulation (**sim**) and optimization (**opt**). In **HeuP**, phase **sim** performs the computation of matrix B: the simulator $\hat{g}$ is developed in Java with the library `SSJ`[8] and the computation of its entries is parallelized by using Bash. In **HeuN**, **sim** retrieves a family of uncertainty sets of agents: the procedure is implemented in Python with `numpy.random` and `networkx`, and the execution is parallelized with `concurrent.futures`. Phase **opt** is implemented in Python for both the methods, by using IBM Decision Optimization for Watson Studio[6] and the IBM `docplex` APIs. The computational experience has been performed over an instance provided by the Bank, based on a call center line that handles credit and debit cards blocking requests. The instance corresponds to the month of August 2018, composed by 527 operational periods: it includes a volume of over 300K predicted inbound calls and 44 feasible shifts derived from the employment contracts. The Bank also provided assumptions and parameters about probability distributions and targeted SLA, and a benchmark tool to evaluate the acceptance of solutions, which determines rejection whether the reached score is less than 80%. Experiments have been carried out on a cluster with 8 CPU cores at 2.7GHz and 64 GB of RAM. Tab.1 reports solutions in terms of cost, staff planned and acceptance score. Tab.2 summarizes the computational performance in terms of elapsed real time for each phase, and the relative optimality gap for the phase **opt** (20-seconds time limit is set).

Finally, we can conclude that the two methods return comparable results in terms of solution quality and acceptance score, while **HeuN** outperforms **HeuP** in terms of computational effort by reducing the time from almost 4 hours to less than 11 minutes.

[1]Let $G_\sigma := (V,E)$ be the the interval graph whose vertices are inbound calls and edges are agent conflicts. Any agent allocation is in bijection with a vertex coloring of G_σ; then, $n(\sigma) = \chi(G_\sigma)$, where $\chi(G_\sigma)$ is the chromatic number of G_σ.

References

[1] J. Atlason, M.A. Epelman, and S.G. Henderson. Optimizing call center staffing using simulation and analytic center cutting-plane methods. *Management Science*, 54(2):295–309, 2008.

[2] M. Bodur and J.R. Luedtke. Mixed-integer rounding enhanced benders decomposition for multi-class service-system staffing and scheduling with arrival rate uncertainty. *Management Science*, 63(7):2073–2091, 2017.

[3] P. De Bruecker, J. Van den Bergh, J. Beliën, and E. Demeulemeester. Workforce planning incorporating skills: State of the art. *European Journal of Operational Research*, 243(1):1 – 16, 2015.

[4] M. Defraeye and I. Van Nieuwenhuyse. Staffing and scheduling under nonstationary demand for service: A literature review. *Omega*, 58(C):4–25, 2016.

[5] M. Excoffier, C. Gicquel, and O. Jouini. A joint chance-constrained programming approach for call center workforce scheduling under uncertain call arrival forecasts. *Computers & Industrial Engineering*, 96:16 – 30, 2016.

[6] IBM. IBM Decision Optimization for Watson Studio Local 1.2.2, IBM Watson Studio Local - content hub, 2018.

[7] A. Juan, J. Faulin, S. Grasman, M. Rabe, and G. Figueira. A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2, 03 2015.

[8] P. L'Ecuyer. SSJ: Stochastic simulation in Java, software library, 2016.

[9] S. Liao, C. van Delft, and J.-P. Vial. Distributionally robust workforce scheduling in call centres with uncertain arrival rates. *Optimization Methods and Software*, 28(3):501–522, 2013.

[10] S. Mattia, F. Rossi, M. Servilio, and S. Smriglio. Staffing and scheduling flexible call centers by two-stage robust optimization. *Omega*, 72:25 – 37, 2017.

[11] T.R. Robbins and T.P. Harrison. A stochastic programming model for scheduling call centers with global service level agreements. *European Journal of Operational Research*, 207(3):1608–1619, 12 2010.

An hybrid VNS and Mathematical Programming Algorithm for a Public Bicycles-Sharing System

Anibal Álvarez[1], Pablo Maya[2], and Guillermo Cabrera-Guerrero[1]

[1] Pontificia Universidad Católica de Valparaíso, Chile
anibal.alvarez.g@mail.pucv.cl;guillermo.cabrera@pucv.cl
[2] Universidad de Antioquia, Colombia
pablo.maya@udea.edu.co

Abstract. A matheuristic approach that combines the well-known variable neighbourhood search (VNS) algorithm and a mathematical programming (MP) solver to solve a novel model for a Public Bicycles-Sharing System is presented. The problem is modelled as an integer programming problem. While the VNS algorithm aims to find the set of optimal repositioning centres, the MP solver computes the optimal allocation network for a given set of repositioning centres. The proposed approach obtains very promising results, specially for those instances where the solver by itself is not able to find feasible solutions within acceptable times.

Keywords: Variable Neighbourhood Search · Public Bicycles-Sharing System · Matheuristic.

1 Introduction

The Bicycle Sharing Systems (BSS) have positioned as public and sustainable alternative in urban mobility. One of the biggest challenges of the BSS operators is to face the asymmetric demand, highly spatial and temporal depended, that causes that bicycles accumulate to certain stations while others are empty. The most common strategy to deal with this problematic is the use of repositioning vehicles to transport bicycles between stations. This problematic has attracted the attention of the research community in the last decade with the number of publications and conference risen considerably in the last couple of years [2].

Most of the research on repositioning bicycles in BSS focuses on operative decisions, such as the routing of the vehicles. However, this paper describes a model and a solution strategy to tackle the districting problem, a tactical decision, that operators faced when they have to partition the operation area of the system in a set of zones to be covered by each of the repositioning vehicles. Furthermore, this approach takes into account not only distance and connectivity when defining the districts of the BSS but also criteria such as demand patterns and stations hierarchy.

1.1 Mathematical Model

Let $\mathcal{E}$ be the set of stations, $\mathcal{C}$ a subset of $\mathcal{E}$ that contains the candidate stations to be the center of a repositioning zone, and $\mathcal{P}$ a set of importance levels that defines the priority that each station is granted for the repositioning strategy. We define r_i^+ and r_i^- as the number of bicycles and parking docks, respectively, required by station i at the peak hour, d_{ij} the distance between station i and j, c_{ij} a binary indicator of the connectivity between stations i and j, p_{il} a binary parameter that indicates whether the station i is assigned priority l, and k the number of repositioning zones to be defined. The model considers two set of decision variables. The binary variable y_j indicates whether the candidate station j is designated to be the center of a repositioning zone, while the variable x_{ij} indicates whether the station i is assigned to the zone centered in the station j.

$$\min \quad \sum_{i\in\mathcal{E}} \sum_{j\in\mathcal{C}} c_{ij}x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{C}} x_{ij} = 1 \qquad \forall i \in \mathcal{E} \tag{2}$$

$$x_{ij} \leq c_{ij}y_j \qquad \forall i \in \mathcal{E}\ \forall j \in \mathcal{C} \tag{3}$$

$$\sum_{j\in\mathcal{C}} y_j = k \tag{4}$$

$$\left| \frac{\sum_{i\in\mathcal{E}} r_i^+ x_{ij} - \sum_{i\in\mathcal{E}} r_i^- x_{ij}}{\sum_{i\in\mathcal{E}} r_i^+ x_{ij} + \sum_{i\in\mathcal{E}} r_i^- x_{ij}} \right| \leq \alpha \quad \forall j \in \mathcal{C} \tag{5}$$

$$\left| \sum_{i\in\mathcal{E}} p_{il}x_{ij} - \left\lfloor \frac{\sum_{i\in\mathcal{E}} p_{il}}{k} \right\rfloor \right| \leq \beta \tag{6}$$

$$x_{ij}, y_j \in \{0,1\} \qquad \forall i \in \mathcal{E}\ \forall j \in \mathcal{C} \tag{7}$$

Objective function (1) minimizes the sum of the distances of each station to the center of the repositioning zone to which the station was assigned. It aims at generating compact zones. It would be preferable to consider a function that minimizes the maximum distance between each pair of stations within the same repositioning zone. However, in order to keep the model computationally tractable, we opted for the first objective function and involve a maximum coverage distance in the definition of the connectivity c_{ij}. Constraints (2) to (4) establish the number of repositioning zones and ensure that each station is assigned to one of the created zones. Constraints (5) aim at balancing the bicycle and parking docks demand within each zone, such that at peak hours not all the stations of a zone demand only bikes or only parking docks which would force the vehicle to move to adjacent zones. The parameter α is the maximum tolerable percentage of unbalance. Similarly, constraints (6) aim at distributing homogeneously the stations among the zones respect to the priority levels. That is, that the most critical stations do not concentrated in a reduced set of zones. The parameter β is the maximum difference allowed between the number of stations of a given priority level and the ideal value within each zone. A different value of β might be used for each level of priority. Finally, constraint (7) defines the variables to be binary.

2 Proposed VNS Algorithm

In this paper we implement a matheuristic algorithm which combines a simple yet efficient Variable Neighbourhood Search (VNS) [3] algorithm and mathematical programming. While the VNS algorithm determines the center of each repositioning zone the mathematical programming solver finds the optimal allocation for the remaining stations. One distinctive aspect of our approach is that it includes some ad-hoc considerations in order to reduce the search space for the allocation problem. For instance, we implement a grid that allows us to uniformly distribute zones centers. Some constraints based on the grid are also used in the allocation phase to enforce the the mathematical programming solver to allocate stations to their adjacent zones, leading to clusters that are much more compact. In this paper, two neighbourhood movement, namely $\mathcal{N}_1$ and $\mathcal{N}_2$ are implemented. While the first one helps us to better *explore* the search space, the second one helps us to *exploit* promising zones of the search space. Let $s \in \mathcal{S}$ be a vector where $s_j \in s$ is 1 if the j-th stations is the centre of its zone and 0 otherwise. We say that $s' \in \mathcal{N}_1(s)$ iff $s_j = s'_j$ for all but one $j = \{1, \ldots, \mathcal{C}\}$. Similarly, we say that $s' \in \mathcal{N}_2(s)$ iff s' for all but three $j = \{1, \ldots, \mathcal{C}\}$.

3 Experiments and Future Work

To run the experiments, we use information from EcoBici, a system that operates in Mexico City and involved, at the moment of the study, 452 stations and more than 600 bicycles. A total of 2.353.389 travel records, for the months with higher level of transactions (September November 2016) were used. The peak and valley hours, station priorities and requirements are defined following the methodology described in [1]. The operation time-frame is divided in intervals of 30 minutes due to the fact that 92% of the trips last less than 32 minutes. Three peak periods were identified. However, we focus on the period of higher demand during the afternoon (e.g., around 18:00) as it accounts for around 30% of the total demand of the system. The system is divided in 15 repositioning zones while the stations are classified in four levels of priority.

References

1. Gaviria, B., Gómez, M., Rodríguez, M.: Demand caracterization of EnCicla bike sharing system. Master's thesis, Universidad de Antioquia (6 2016)
2. Laporte, G., Meunier, F., Calvo, R.W.: Shared mobility systems. 4or **13**(4), 341–360 (2015)
3. Mladenovi, N., Hansen, P.: Variable neighborhood search. Computers Operations Research **24**(11), 1097 – 1100 (1997)

Nested genetic algorithm to collaborative school buses routing problem

Marcela Torné [1], Juan P. Orejuela [2], Claudia C. Peña[3],

[1]Universidad del Valle
Cl. 13 #100-00, Cali, Valle del Cauca, Colombia
marce0314@gmail.com

[2] Universidad del Valle
Cl. 13 #100-00, Cali, Valle del Cauca, Colombia
juan.orejuela@correounivalle.edu.co

[3] Universidad Autónoma de Occidente
Cl. 25 #115-85, Cali, Valle del Cauca, Colombia
ccpena@uao.edu.co

Abstract

This document presents a methodology based on nested genetic algorithms to solve collaborative school buses routing problem. In this problem, there are a set of parents who are willing to send their children to a set node to be picked up by the bus. This set node represents the location of a sub-set of parents that are trust and are willing to receive the children and take them to the route. Parents who send and receive the children have a discount in the transport rate as a collaboration strategy, which reduce the number of nodes to visit and the total journey time. It is necessary to establish the number of collaborations which maximize the global benefit because the parents received a benefit which is actually a cost to the system.

1 Introduction

People in countries and cities are facing long journeys time, high costs in mobility, traffic jams and recently it has been reported the impact of mobility issues on the quality of life of families and their members, despite efforts to improve mobility and the investments in infrastructure [1]. This issue is presented in the school transport since the longer the journeys to arrive to school, the earlier families have to start their day in order to fulfil with the academic schedules and other home duties, impacting the academic performance of the students and the time they share as a family [2]. Based on the above, the aim of this article is to propose an approach to the school routing problem in a collaboration context, considering an efficient planning of the routes and the decrease of the travel time. A mathematical model was developed which included collaboration strategies between the agents, guarantying their interaction and collaboration as a common welfare. The model was solved with a two phase's nested genetic algorithm. The first phase defines who collaborates and the second one, solves the resulting routing problem.

1.1 School routing problem

The school routing problem aims at planning an efficient schedule for school bus fleet which each bus pick students up at various stops and take them to the corresponding schools, satisfaying different constraints such as maximum load of the bus, maximun travel time of a single student and time window of school [3]. Some authors agree [4-5] that the school routing problem is complex because it involves multiple factors: increase on the number of students, several stops that are necessary to cover and constraints related to capacity of the buses. According to [6], this problem is also impacting parents and children wealfare, because they both need to start their day earlier in orden to fulffil with the schedules; also it increases the tiredness due to long travel, it decrases the time to share with the family and also it affects the academic performance of the students in the schools.

1.2 Collaborative school routing buses

Currently, cooperation has been a key research topic in the supply chain management [7]. Working together with partners or stakeholders allows to improve effectivity and efficiency in the supply chain [8]. A collaboration proposal is presented to transport students in Bogota where a set of schools are located

in a zone, but there is only a road that connects those schools with the neighborhoods [8]. Similarly, [9] present an approach to arch routing problem with a collaboration strategy. Additionally, [10] show a routing problem in multiple centers which utilize collaboration in sending and receiving the children, so as to minimize the operational cost and the total number of vehicles within the network.

2 The problem

2.1 Description of the problem

For risk reasons in some countries, the students should be picked up at home which represents several stops in a same sector; this is different from some other countries where there are common stops for the students [3]. The common stop is key to the collaboration strategy, however to minimize the risks, the common stops are proposed at homes where parents are willing to receive to other children from the same school and these parents will get an economic benefit for providing this service to the system. Other parents are willing to send their children to some common stops represented by other students´ homes and they also will receive an economic benefit. The problem consists in stablishing which and how many students will be allocated to each home, from all the homes that are available to send and receive and also it is necessary to define which routes should take the buses to solve the resulting route, so as to maximize the system benefit.

2.2 Mathematical formulation

General assumptions

- Buses fleet is homogenous
- Stop time is minimum and it is not considered in the model.

Collaboration assumptions

- If children from home *i* are sent to home *j*, parents from both families trust each other and children are friends.
- Nodes that are willing to receive, are not willing to send; and vice versa
- **Variables**

$$A_{i,j} = \begin{cases} 1. \text{ If children from home } i \text{ are sent to home } j \\ 0. \text{ Otherwise} \end{cases}$$

$$X_{k,i,j} = \begin{cases} 1. \text{ If the vehicle } k \text{ visits } j \text{ after visiting } i \\ 0. \text{ Otherwise} \end{cases}$$

- **Constraints**
- Only a single vehicle can visit the active nodes
- Constraints to calculate the costs
- Constraints to guarantee smooth flow
- Constraints to control capacity

3 Solution strategy

It is integrated with two genetic algorithms. One of them, aims at allocating from node to node according to the collaboration strategy mentioned before. In this strategy, there are parents who are willing to receive children at their homes and there are parents who are willing to send their children to those homes. Therefore, this allocation algorithm solve the problem related to how children will be allocated, deciding who will be sent and received. Total incomes of the system are calculated with this allocation. The second genetic algorithm is aim at routing the buses according to the stops defined previously with the allocation algorithm. The results from this part show the nodes that will be covered by the buses and the order of precedence to be visited from the parking lot until the arrival to the school. This routing makes possible to calculate the total costs of the system.

4 Results

This proposal demonstrates that the lower the collaboration, the longer the traveled distances, therefore higher the time a child is within the route. There is a 21% difference in the total traveled distance by comparing the results from the first generation with the second one. This represents 210,28 meters less and confirm the goodness of collaboration to diminish the transport time and the buses travel distances. This is similar to ideas exposed by [8] who solved the school bus routing problem with collaboration.

References

[1] Nocera, S., & Cavallaro, F. (2014). The ancillary role of CO2reduction in Urban transport plans. Transportation Research Procedia, 3(July), 760–769. https://doi.org/10.1016/j.trpro.2014.10.055

[2] Hinrichs, P. (2011). When the bell Tolls: The effects of school starting times on academic achievement. Education Finance and Policy, 6(4), 486–507. https://doi.org/10.1162/EDFP_a_00045

[3] Park, J., & Kim, B. I. (2010). The school bus routing problem: A review. *European Journal of Operational Research*, *202*(2), 311–319. https://doi.org/10.1016/j.ejor.2009.05.017

[4] Arias Rojas, J. S., Jiménez, J. F., & MontoyaTorres, J. (2012). Solving of school bus routing problem by ant colony optimization. Revista EIA Rev.EIA.Esc.Ing.Antioq, 193–208.

[5] Pérez-Rodríguez, R., & Hernández-Aguirre, A. (2016). Probability model to Solve the School Bus Routing Problem with Stops Selection. International Journal of Combinatorial Optimization Problems & Informatics, 7(1), 30–39. https://doi.org/10.6036/8204

[6] Taylor, K. L. (2017). School Start Times: When Should the Bells Ring? Retrieved from http://www.carolinaparent.com/CP/School-Start-Times-When-Should-the-Bells-Ring/

[7] Skippari, M., Laukkanen, M., & Salo, J. (2017). Cognitive barriers to collaborative innovation generation in supply chain relationships. Industrial Marketing Management, 62, 108–117. https://doi.org/10.1016/j.indmarman.2016.08.002

[8] Rodríguez Parra, G. R., Guerrero, W. J., & Sarmiento-Lepesqueur, A. (2017). Cooperation strategies featuring optimization in the school transportation system in Bogota. *Dyna*, *84*(202), 164–174. https://doi.org/10.15446/dyna.v84n202.65391

[9] Fernández, E., Fontana, D., & Speranza, M. G. (2016). On the Collaboration Uncapacitated Arc Routing Problem. Computers and Operations Research, 67, 120–131. https://doi.org/10.1016/j.cor.2015.10.001

[10] Wang, Y., Zhang, J., Assogba, K., Liu, Y., Xu, M., & Wang, Y. (2018). Collaboration and transportation resource sharing in multiple centers vehicle routing optimization with delivery and pickup. Knowledge-Based Systems, 160(July), 296–310. https://doi.org/10.1016/j.knosys.2018.07.024

An Effective Tabu Search Method with a Limited Search Space for Carpooling Optimization Problems

Kosei Takahashi[1] , Toshichika Aoki[1] , Takayuki Kimura[2], and Tohru Ikeguchi[3,4]

[1] Graduate School of Electronics, Information and Media Engineering,
Faculty of Engineering, Nippon Institute of Technology
[2] Department of Electrical, Electronics and Communication Engineering, Faculty of Fundamental Engineering,
Nippon Institute of Technology
4–1 Gakuendai, Miyashiro, Minami-Saitama, Saitama, 345-8501, Japan
[3] Department of Management Science, Graduate School of Engineering, Tokyo University of Science
[4] Department of Information and Computer Technology, Faculty of Engineering, Tokyo University of Science,
6–3–1 Niijuku, Katsushika, Tokyo 125–8585, Japan
2198016@stu.nit.ac.jp, 2188001@sstu.nit.ac.jp, tkimura@nit.ac.jp, tohru@rs.tus.ac.jp

Abstract

Carpooling systems are one of the effective means of reducing traffic congestion. Recently, many carpool matching services, such as Carpool Global and Share Your Ride, have arisen to match passengers as carpool members. Such carpool matching services require high-speed matching of carpool groups, while considering the distances between drivers and the pick-up and drop-off points of passengers. Constructing an efficient route in a carpooling system is called the carpooling optimization problem (COP). To solve the COP, an approximate solution method has been proposed using tabu search. However, this method fully searches all routes between the passengers and drivers in possible carpooling groups in each iteration, and this procedure requires considerable computational costs. In this study, we improve this grouping procedure by employing an evaluation function that automatically determines the carpooling groups of drivers and passengers. Numerical experiments demonstrate that the tabu search method using our proposed procedure efficiently determines better carpooling groups, and also drastically reduces the calculation time by approximately 300 seconds compared to the conventional method.

1 Introduction

In modern society, the numbers of active vehicles in traffic networks have been considerably increasing, causing heavy traffic congestion, particularly in developing countries. To resolve this problem, carpooling systems, which share routes of drivers and passengers between their residential and working areas, have gained considerable attention as an effective means of reducing traffic [1]. In addition, such systems can successfully prevent air pollution.

Many matching services, such as Carpool Global and Share Your Ride, are already available to realize carpooling systems [3]. These carpool matching services require high-speed matching of carpooling groups, and should search for shorter routes for groups considering the distances between drivers and the pick-up and drop-off points of passengers. Constructing an efficient route in a carpooling system is called the carpooling optimization problem (COP). The previous study [4] proposed a tabu search method [2] to solve the COP. However, this method searches all the candidate routes for drivers and passengers in each possible group. In addition, the number of possible candidate routes increases exponentially if a large number of passengers participate in a carpooling group, so that this searching of routes requires considerable computational costs. From this perspective, in this study, we improve the tabu search method using an evaluation function that automatically determines the carpooling groups of drivers and passengers. Numerical experiments demonstrate that the tabu search method using our proposed procedure rapidly constructs better carpooling routes.

2 Carpooling Optimization Problem (COP)

We refer to the departure points of drivers and passengers in carpooling groups as "residences" and the destinations as "workplaces." Furthermore, these locations are expressed by nodes.

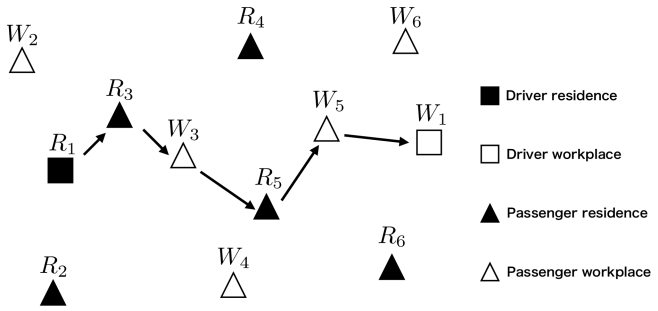


Figure 1: Example of a carpooling route [4]. In this figure, black and white squares represent the residence and workplace of the driver, respectively; and black and white triangles represent the residences and workplaces of passengers, respectively. Arrows illustrate the order of the pick-up and drop-off points of passengers.

The objective of the COP is to find routes with the minimum distances for all carpooling groups of drivers and passengers. The objective function of the COP is then defined as follows:

$$\min Z = \sum_{i=1}^{2N} \sum_{j=1}^{2N} \sum_{r=1}^{P} d_{ij} x_{ijr}, \tag{1}$$

where N is the total number of candidates; $P = 2E + 1$ is the total number of routes, where E is the number of passengers; and d_{ij} is the shortest distance between the ith and jth nodes. In our model, $1 \leq i, j \leq N$ represent the residences and $N + 1 \leq i, j \leq 2N$ represent the workplaces, where the residence i corresponds to the workplace at $i + N$.

Furthermore, x_{ijr} is a decision variable defined as follows:

$$x_{ijr} = \begin{cases} 1 & \text{(if the } r\text{th path has an edge from the } i\text{th to the } j\text{th nodes),} \\ 0 & \text{(otherwise).} \end{cases}$$

3 Evaluation function for distances between drivers and passengers

To construct an evaluation function for the carpooling groups, we first prepare a set of drivers D and a set of customers C. Next, we randomly select four customers from C, as in this study we assume that each vehicle has space for at most four passengers. The set of selected customers is denoted by C_{selected}. Next, we attempt to find the best C_{selected} with the shortest distance from a driver $l \in D$ by minimizing the following function:

$$\min Q_l(C_{\text{selected}}) = \sum_{n \in C_{\text{selected}}} \left\{ q_l(n) + q_l(n+N) \right\}, \; l \in D, \tag{2}$$

where $q_l(i) = d_{s(l)i} + d_{it(l)}, \; l \in D$ is the combined distance from the residence $s(l)$ of the driver l to the ith node and from the ith node to the workplace $t(l)$ of the driver. The reason for writing the second term in Eq. (2) as $q_l(n + N)$ (adding N) is that the workplace corresponding to the node n is expressed as $n + N$.

The previous tabu search approach memorizes the selected routes, namely x_{ijr} for each route r. However, the proposed method memorizes the combination of passengers C_{selected} in the tabu list to find

the optimum set C_{selected} with the minimum $Q_l(C_{\mathrm{selected}})$. After finding combinations of passengers, we then conduct a route calculation using a shortest path search method such as the Dijkstra algorithm.

4 Numerical experiments

We considered the Guangzhou city road network in China [4] for these experiments. The carpooling sets, including drivers and passengers, were randomly generated in the same manner as described in Ref. [4]. In addition, we randomly selected four passengers from C to construct the initial set C_{selected}, and searched 500 times for combinations of C_{selected} with the minimum value of Eq. (2) using the tabu search. We compared the performance of the conventional method [4] with that of our method. The tabu tenure of the tabu search was set to 10.

Table 1: Experimental results (objective function/calculation time)

	$N = 100$	$N = 200$	$N = 300$
conventional	$95.40/342.20$	$92.74/342.19$	$90.99/342.19$
proposed	$\mathbf{67.73/0.50}$	$\mathbf{89.78/0.48}$	$\mathbf{53.18/0.48}$

Table 1 shows the objective function and calculation times for the conventional method [4] and our proposed method. It can be observed that the objective function for the tabu search using our proposed grouping procedure decreases by 71.0 %, 96.8 %, and 58.4 % for the cases with $N = 100, 200$, and 300, respectively, compared to the conventional method. In addition, the proposed method also reduces the calculation time by approximately 300 seconds compared to the conventional method for every problem.

5 Conclusion

In this study, we propose a tabu search solution method for efficient carpooling grouping in the COP. Whereas the conventional tabu search method calculates all the routes between drivers and the pick-up and drop-off points of passengers for all possible grouping cases, our method efficiently forms carpooling groups using an evaluation function that automatically determines drivers and passengers. Numerical experiments demonstrate that the proposed method successfully reduces the objective function value and calculation time compared to the conventional method. In this study, we evaluate problems in which at most four passengers exist in each group. However, the number of candidate routes drastically increases if the number of passengers in a group can be larger than five, such as in the vanpool problem. In future work, we plan to propose a method to quickly search good solutions for vanpooling optimization problems.

The research of T. K. was supported by JSPS KAKENHI Grant Number 16K21327. The research of T. I. was supported by JSPS KAKENHI Grant Numbers 15TK0112 and 17K00348.

References

[1] Ronald Fagin and John H Williams. A fair carpool scheduling algorithm. *IBM Journal of Research and development*, 27(2):133–139, 1983.

[2] Fred Glover. Tabu search—part I. *ORSA Journal on computing*, 1(3):190–206, 1989.

[3] Shih-Chia Huang, Ming-Kai Jiau, and Chih-Hsiang Lin. Optimization of the carpool service problem via a fuzzy-controlled genetic algorithm. *IEEE Trans. Fuzzy Systems*, 23(5):1698–1712, 2015.

[4] Jizhe Xia, Kevin M Curtin, Weihong Li, and Yonglong Zhao. A new model for a carpool matching service. *PLOS ONE*, 10(6):e0129257, 2015.

A matheuristic for the Multi-period Electric Vehicle Routing Problem

Laura C. Echeverri[1], Aurélien Froger[1], Jorge E. Mendoza[2], Emmanuel Néron[1]

[1] Université de Tours, LIFAT EA 6300, CNRS, ROOT ERL CNRS 7002
64 avenue Jean Portalis, Tours 37200, France
laura.echeverriguzman@etu.univ-tours.fr,{aurelien.froger,emmanuel.neron}@univ-tours.fr

[2] HEC Montréal
3000 Chemin de la Côte-Sainte-Catherine, Montréal (Québec) H3T 2A7, Canada
jorge.mendoza@hec.ca

Abstract

The Multi-period Electric Vehicle Routing Problem (MP-E-VRP) consists on designing routes to be performed by a fleet of electric vehicles (EVs) to serve a set of customers over a planning horizon of several periods. EVs are charged at the depot at any time, subject to the charging infrastructure capacity constraints (e.g., number of available chargers, power grid constraints, duration of the charging operations). Due to the impact of charging and routing practices on EVs battery aging, degradation costs are associated with charging operations and routes. The MP-E-VRP integrates EV routing and depot charging scheduling, and has coupling constraints between days. These features make the MP-E-VRP a complex problem to solve. In this talk we present a two-phase matheuristic for the MP-E-VRP. In the first phase it builds a pool of routes via a set of randomized route-first cluster-second heuristics. Routes are then improved by solving a traveling salesman problem. In the second phase, the algorithm uses the routes stored in the pool to assemble a solution to the MP-E-VRP. We discuss computational experiments carried out on small-size instances.

1 Introduction

We define the Multi-period Electric Vehicle Routing Problem (MP-E-VRP) as follows. Let $\mathcal{P} = \{0, \ldots, P_{max} - 1\}$ be the set of periods within the planning horizon. Let $\mathcal{I}$ be the set of visit nodes and 0 a node representing the depot. Each node $i \in \mathcal{I}$ represents a customer with a service time $v_i > 0$ and a period $p_i \in \mathcal{P}$ in which the service must take place. A finite set of homogeneous electric vehicles (EVs) denoted by $\mathcal{K}$ is available to visit the nodes in $\mathcal{I}$. Each EV $k \in \mathcal{K}$ has a battery capacity Q (in kWh). Travelling from one location i (the depot or a visit node) to another location j incurs a driving time $t_{ij} \geq 0$ (in hours) and an energy consumption $e_{ij} \geq 0$ (in kWh). The triangular inequality holds for driving times and energy consumptions. At period $p \in \mathcal{P}$, an EV cannot leave the depot before E_p or return after L_p (with $L_p > E_p$). In other words, time interval $[E_p, L_p]$ corresponds to the opening hours of the depot for period $p \in \mathcal{P}$.

At the beginning of the planning horizon each EV $k \in \mathcal{K}$ has an energy kWh_0^k (in kWh). The EVs can be fully or partially recharged at the depot at any time. There is no possibility to recharge an EV outside of the depot. The EVs can only perform one route per period and they are allowed to be recharged only once between two routes. Multiple chargers are available at the depot. Each charger is associated with a charging mode (e.g., slow, fast, moderate) belonging to set $\mathcal{M}$. The depot has A_m available chargers for each charging mode $m \in \mathcal{M}$ (usually, but not necessarily, $\sum_{m \in \mathcal{M}} A_m < |\mathcal{K}|$). During a charging operation, a charger of mode m consumes a power pw_m (in kW) from the grid. An EV can be recharged using any available charger, but at any time the power grid capacity PW_{max} must not be exceeded. Moreover, it is forbidden to switch the charging mode while charging an EV. Charging operations are also non-preemptive (i.e. they cannot be interrupted and resumed later). Each charging mode is associated with a piecewise linear charging function $g_m(\Delta)$ and a fixed cost per charge u_m.

Routing and charging decisions impact the lifespan of EV batteries. To take this into account, a cumulative wear cost is incurred when performing routes and charging operations. This cost is modeled as a piecewise linear function that maps the state of charge (SOC) of the battery σ to the sum of the wear

cost associated to charge the EV to every SOC from 0% to σ or discharge it from every SOC σ to 0%. The SOC of a battery is the percentage of available battery capacity. The cumulative wear cost function is derived from the model proposed by [2].

Feasible solutions to the MP-E-VRP satisfy the following conditions: 1) each node $i \in \mathcal{I}$ is visited by a single EV in period p_i, 2) each route starts and ends at the depot, 3) each EV can perform one route per period, 4) when an EV starts a route, it must have enough energy to complete it (no mid-route charging is allowed), 5) no more than $|\mathcal{K}|$ EVs are used on a single period, 6) EVs visit nodes in $\mathcal{I}$ in period $p_i \in \mathcal{P}$ only during time window $[E_{p_i}, L_{p_i}]$, 7) the power consumed by all chargers at any time does not exceed the maximum power PW_{max}, and 8) no more than A_m EVs simultaneously charge at the depot using mode $m \in \mathcal{M}$.

The objective of the MP-E-VRP is to determine for each period a set of routes visiting all the customers and to schedule the charging operations of the EVs while minimizing the cost of battery degradation. This cost is defined as the sum of the fixed costs of all charging operations and the cumulative wear costs associated with all charging operations and routes.

We also define a related problem that takes as inputs a set of pre-generated routes. The problem that consists on selecting a subset of routes from the pool, assigning EVs to them and scheduling charging operations while satisfying conditions 1)-8) is referred to as the Multi-period Electric Vehicle Route Assignment and Charge Scheduling Problem (MP-E-VRACSP).

2 Matheuristic for the MP-E-VRP

To tackle the MP-E-VRP we present a matheuristic based on the multi-space sampling heuristic (MSH) [3]. MSH has two phases: sampling and assembling. In the sampling phase the algorithm uses a set of randomized traveling salesman problem (TSP) heuristics to obtain a set N of TSP-like tours. Then, MSH extracts every feasible route that can be obtained without altering the order of the customers of each TSP tour, following the route-first cluster-second principle [1, 4]. MSH uses these routes to build a set $\Omega \subset \mathcal{R}$, where $\mathcal{R}$ is the set of all feasible routes. In the assembling phase MSH finds a solution s, from the set of all feasible solutions to the problem $\mathcal{S}$, by solving a set partitioning formulation on set Ω.

Algorithm 1 describes the general structure of our matheuristic. The procedure starts by entering the sampling phase (lines 1-1) for each period $p \leq |\mathcal{P}| - 1$. At each iteration $n \leq N$, the algorithm selects a sampling heuristic from a set $\mathcal{T}$ (line 1) and uses it to build a TSP tour tsp_p visiting the customers that must be serviced in that period ($i \in \mathcal{I} : p_i = p$). Then, the algorithm uses a tour splitting procedure (known as `split`) to retrieve a set of routes added to the set of routes for period p, $\Omega'_p \in \mathcal{R}$. Routes in Ω'_p respect a maximum duration, related with the opening hours of the depot ($L_p - E_p$), and the maximum energy consumption, Q. Then, the algorithm calls procedure `custSeqs`($\mathcal{R}_p$) - line 1. This procedure retrieves all the unique sets of customers present in set Ω'_p and generates the set of customer sequences $\mathcal{CS}_p$. Then the algorithm invokes procedure `tsp`($\mathcal{CS}_p$) - line 1. The latter solves a TSP for each set of customers in $\mathcal{CS}_p$ and stores in Ω_p the resulting routes. Routes in Ω_p are then stored in Ω. In the assembly phase (line 1), the algorithm invokes a procedure called `mpevracsp` to solve a MP-E-VRACSP by selecting routes from Ω to find a feasible MP-E-VRP solution. We propose a continuous-time mixed integer linear programming (MILP) formulation of the MP-E-VRACSP. This formulation uses arc-based tracking variables for time and energy. More details on the algorithm implementation and the MILP formulation will be discussed in the talk.

3 Computational experiments

We implemented our matheuristic in Java (jre V.1.8.0) and used Gurobi Optimizer (version 8.0.1) to solve the MILP formulation of the MP-E-VRACSP (Algorithm 1, line 1). We compare the results of out matheuristic with results obtaining from solving a MILP formulation of the MP-E-VRP. We set a time limit of 3 hours. We generated 100 small size instances of the MP-E-VRP. We consider 3 periods of 8 hours. We generated instances with 5, 10, or 15 customers in each period. Customers locations

Algorithm 1 Matheuristic: general structure

function MATHEURISTIC($\mathcal{P}, \mathcal{I}, \mathcal{K}, \mathcal{M}, \mathcal{T}, N$)
  $\Omega \leftarrow \emptyset$
  $n \leftarrow 1$
  for $p = 0$ to $p = |\mathcal{P}| - 1$ **do**
    $\Omega'_p \leftarrow \emptyset$
    while $n \leq N$ **do**
      for $l = 1$ to $l = |\mathcal{T}|$ **do**
        $h \leftarrow \mathcal{T}_l$
        $tsp_p \leftarrow h(\mathcal{I})$
        $\Omega'_p \leftarrow \Omega'_p \cup \texttt{split}(\mathcal{I}, tsp_p)$
        $n \leftarrow n + 1$
      end for
    end while
    $\mathcal{CS}_p \leftarrow \texttt{custSeqs}(\Omega'_p)$
    $\Omega_p \leftarrow \texttt{tsp}(\mathcal{CS}_p)$
    $\Omega \leftarrow \Omega \cup \Omega_p$
  end for
  $\sigma \leftarrow$mpevracsp$(\Omega, \mathcal{P}, \mathcal{I}, \mathcal{K}, \mathcal{M})$
  return σ
end function

are randomly generated in a geographic space of approximately 45×45 km. This represents an urban operation in which 16 kWh-EVs can perform routes without need for mid-route charging. For each number of customers per period we generated 5 sets of customer locations. Service time is fixed to 0.75 hours. We assume the EVs have an energy consumption rate of 0.125 kWh/km. The initial energy for all EVs is fixed to 8 kWh or 12.8 kWh. For instances with 5 customers per period we considered 2 EVs, for the rest of the instances, 4 or 5 EVs. We included two types of charging mode: $slow$ (with a power of 6 kW and a fixed cost per charge of $\$1.22$) and $moderate$ (with a power of 11 kW and a fixed cost per charge of $\$1.42$). For $slow$ mode, there is the same number of available chargers as for EVs, while for $moderate$ mode there is only one charger available. We considered instances with only one charging mode, $slow$, and instances with two charging modes, $slow$ and $moderate$. The power grid capacity is adjusted depending on the number of EVs, so that all chargers of both charging modes cannot be used at the same time. We fixed parameter $N = 1000$.

For the 5 customers per period - instances our matheuristic was able to find the optimal solutions. For the 10 customers per period - instances our method reported average improvements of 0.2% with respect to the solutions delivered by the solver running the MP-E-VRP. For the 15 customers per period - instances, the improvement is of 4.45%.

References

[1] N. Christofides and J. E. Beasley. The period routing problem. *Networks*, 14(2):237–256, 1984.

[2] Sekyung Han, Soohee Han, and Hirohisa Aki. A practical battery wear model for electric vehicle charging applications. *Applied Energy*, 113:1100–1108, 2014.

[3] Jorge E Mendoza and Juan G Villegas. A multi-space sampling heuristic for the vehicle routing problem with stochastic demands. *Optimization Letters*, 7(7):1503–1516, 2013.

[4] Christian Prins, Philippe Lacomme, and Caroline Prodhon. Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C: Emerging Technologies*, 40:179–200, 2014.

A matheuristic framework for profitable tour problem with electric vehicles and mandatory stops

David L. Cortés-Murcia[1], Caroline Prodhon[1], H. Murat Afsar[1]

ICD-LOSI, Université de Technologie de Troyes
12 Rue Marie Curie, CS 42060, 10004 Troyes Cedex France
david.cortes_murcia@utt.fr, caroline.prodhon@utt.fr, murat.afsar@utt.fr

Abstract

In this paper, a generalization of the capacitated profitable tour problem with electric vehicles is presented. The aim here is to synchronize the lunch break and the recharging activities by choosing clusters of restaurants where electric vehicles can be charged. Due to the fact that having an agreement with restaurant chains has an associated cost, the problem can also be seen as a problem with location aspects. This variant is pertinent especially in a city logistics context. A matheuristic framework is implemented which is able to solve instances with up to 100 customers and 6 clusters of restaurants in less than 3 seconds with an average GAP no larger than $0,3\%$.

1 Introduction

Nowadays electric vehicles (EVs) are considered a suitable alternative to replace conventional vehicles (CVs) in last mile operations [3]. The adoption of EVs responds to the need to reduce the negative environmental impact generated by transport operations. EVs not only represent one of the cleanest means of transportation in urban areas, but also contribute to noise reduction, permit a high tank-to-wheel efficiency, and offer lower operational cost with respect to CVs [7]. Thanks to government incentives and technological improvements their market share is increasing. Thus, some companies have included EVs into their delivery operations [6].

However, [6] highlights some challenges that must be faced by operators in implementation of EVs for now: limited range of vehicle models, limited mileage range, low vehicle speed, relatively long charging times, and limited payload. According to [5], in France and UK, the urban freight transport and service operators report limited range and risk of queuing at recharging stations (RSs) as technical and operational obstacles after having adopted EVs.

Studies on the Vehicle Routing Problems (VRP) which consider usage of EVs increase in recent years [1]. In general in those variants, the charging time turns out to be an idle time and driver's mandatory break times are not explicitly considered. Additionally, in the work of [5] operators mention that an eventual boost charging, allows to recharge up to 80% battery in about 30 minutes. It would enable the use of the EV throughout the day. Likewise, two operators highlight the importance of being able to synchronize the recharging and lunch times as a way to mitigate the impact of recharge time and the risk of waiting for recharge.

In this paper, a solution approach to solve a variant of a routing problem with profits and EVs is presented. In this problem, it is possible to recharge the EVs during the lunch time. Lunch break is considered a mandatory stop. Additionally, it is assumed that, it is possible to have an agreement with a set of restaurants which provides a RS and a reservation for recharging during lunch. The aim is to benefit from idle time caused for the lunch break, and synchronizing with vehicle recharging. A fixed cost is associated with selection of a cluster of restaurants. It represents the cost established for each restaurant chain to provide food and recharging service. So, selecting a restaurant can be seen as a location decision. Likewise, an estimated profit for visiting customers is defined. The objective is to maximize the total operational benefit. Thus, the proposed problem generalizes the capacitated profitable tour problem (CPTP), where decision related to visit or not customers, clusters of restaurants and restaurants must be made.

2 Problem description

The problem is defined on a graph $G = (V', A)$ with a set of vertices $V' = \{V \cup R\}$ and a set of arcs given by $A = \{(i, j)|i, j \in V', i \neq j\}$. Let be V composed of customers and depot, R the set of restaurants, and F the set of clusters of restaurants. A homogeneous EV fleet of size U is available at the depot. For each visit to a restaurant, the lunch duration is considered constant. Also, we assume that during that time the EV is charged to its maximum level B. Energy consumption through an arc (i, j) is described as a linear relation between distance d_{ij} and the consumption rate cr. Each customer $i \in V$ has a positive profit p_i. Each cluster $j \in F$ have a fix cost C_j associated to food and recharge service in their restaurant chain. Each restaurant $j \in R$ has a hard time window $[e_j, l_j]$. It represents the lunch time established by the restaurant schedule or by the company policies. Lastly, a maximum tour duration is represented by $Tmax$.

The problem can be formulated as a route-based form as follows. Let Ω be the set of feasible vehicle routes. In other words, the set of arcs in A issued from the depot, visiting one restaurant, going back to the depot, satisfying capacity, battery, restaurant time windows constraints, and visiting at most once each customer. Let ρ_k be the income of the route $r_k \in \Omega$. Let a_{ik} be a binary parameter equal to 1 if route r_k visits node i, 0 otherwise. Let γ_{fk} be a binary parameter equal to 1 if route r_k visits a restaurant that belongs to the cluster $f \in F$, 0 otherwise. Binary variable θ_k is equal to 1 if route r_k is part of the solution, 0 otherwise. Binary variable μ_f indicates if the cluster of restaurants f is visited or not. Each cluster $f \in F$ has a fix service cost c_f when it is selected in the solution.

Thus, the model can be described with the following set-packing model (SPM):

$$(SPM) \qquad max \sum_{r_k \in \Omega} \rho_k \theta_k - \sum_{f \in F} c_f \mu_f \tag{1}$$

$$\sum_{r_k \in \Omega} \theta_k \leq U \tag{2}$$

$$\sum_{r_k \in \Omega} a_{ik} \theta_k \leq 1 \qquad \forall i \in V \tag{3}$$

$$\sum_{r_k \in \Omega} \gamma_{fk} \theta_k \leq U \mu_f \qquad \forall f \in F \tag{4}$$

$$\theta_k \in \{0, 1\} \qquad \forall r_k \in \Omega \tag{5}$$

$$\mu_f \in \{0, 1\} \qquad \forall f \in F \tag{6}$$

The objective function (1) aims to maximize the total operational income computed as the sum of the profits minus the operational cost associated with the total distance and minus total cost associated with the restaurant service. Constraints (2) limits the number of vehicles that are used. Constraints (3) guarantee that customers can be served at most once. Constraints (4) indicates if a restaurant is visited by the solution. Finally, domain variable definitions are defined in constraints (5) - (6).

3 Matheuristic framework

As a solution method we implement a matheuristic framework which is composed of two phases. The first phase is a heuristic route generation, where promising routes are generated by executing a neighborhood based metaheuristic. In the second phase a restricted version of the SPM presented above is solved using the routes generated in first phase as set of routes Ω. The advantage of this method is that phase 1 can be designed to improve customer visits and routing decisions, while the second phase is in charge of visiting clusters decisions. Good solutions can be reached if a large diversity of routes during first phase is provided.

In the ongoing work, for the first phase a GRASP algorithm is implemented. First, we build an initial solution by inserting randomly a restaurant to each route and then inserting customers with a greedy

algorithm. Second, we develop a Variable Neighborhood Descend as local search strategy [4]. It follows a best improvement strategy using neighborhoods based on Insert, Remove, Relocate and Swap operators over visited and unvisited nodes.

4 Preliminary results and conclusions

A set of instances of this problem is created based on the set of instances proposed by [2]. The matheuristic is implemented in C++ and SPM is solved by CPLEX 12.8. The algorithm is executed on a computer with Intel(R) Core(TM) i5-5300U CPU @2.30GHz and 8 GB RAM.

In the preliminary results we test the instances that are solved to optimality by the Branch-and-Price method introduced in [2]. The instances have three types of batteries: B_{small}, B_{medium}, and B_{large}. Preliminary results are presented in Table 1. n is the number of available customers, c is the number of clusters of restaurants, $nbIns$ is the number of instances tested, $\%GAP$ the average GAP percentage with respect to the optimal solutions, and CPU is the average computational time (in seconds).

			B_{small}			B_{medium}			B_{large}		
Set	n	c	$nbIns$	$\%GAP$	CPU(s)	$nbIns$	$\%GAP$	CPU(s)	$nbIns$	$\%GAP$	CPU(s)
Set-1	30	3	54	0,27	0,62	54	0,00	0,84	54	0,00	0,75
Set-2	19	3	33	0,00	0,91	33	0,00	0,82	33	0,00	0,88
Set-3	31	3	60	0,00	0,98	60	0,00	1,21	60	0,00	1,22
Set-4	98	5	38	0,04	1,94	31	0,08	2,41	33	0,05	2,68
Set-5	64	4	72	0,00	1,48	65	0,28	1,80	65	0,30	1,84
Set-6	62	4	42	0,00	0,99	39	0,05	1,25	38	0,00	1,26
Set-7	100	6	54	0,05	1,83	45	0,03	1,95	44	0,04	2,03

Table 1: Computational results with different B values

Table 1 shows that the proposed algorithm is capable to solve instances from 19 customers and 3 clusters up to 100 customers and 6 clusters, in less than 3 seconds and with a optimality GAP no larger than $0,3\%$ on average. We expect to test the algorithm in instances adapted from routing problems with EVs, and to test different route generation heuristics to evaluate the performance.

References

[1] Anagnostopoulou Afroditi, Maria Boile, Sotirios Theofanis, Eleftherios Sdoukopoulos, and Dimitrios Margaritis. Electric vehicle routing problem with industry constraints: trends and insights for future research. *Transportation Research Procedia*, 3:452–459, 2014.

[2] David L. Cortés-Murcia, H. Murat Afsar, and Caroline Prodhon. A branch and price algorithm for the electric capacitated profitable tour problem with mandatory stops. In *IFAC-PapersOnLine series*. Elsevier, August 2019. (in press).

[3] Brian A Davis and Miguel A Figliozzi. A methodology to evaluate the competitiveness of electric delivery trucks. *Transportation Research Part E: Logistics and Transportation Review*, 49(1):8–23, 2013.

[4] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100, 1997.

[5] Eleonora Morganti and Michael Browne. Technical and operational obstacles to the adoption of electric vans in france and the uk: An operator perspective. *Transport Policy*, 63:90–97, 2018.

[6] Nina Nesterova and Hans Quak. State of the art of the electric freight vehicles implementation in city logistics - update 2015. Technical report, TNO, 2015.

[7] Samuel Pelletier, Ola Jabali, and Gilbert Laporte. 50th anniversary invited articlegoods distribution with electric vehicles: review and research perspectives. *Transportation Science*, 50(1):3–22, 2016.

An $n\ log\ n$ Heuristic for the TSP

Éric D. Taillard[1]

Univ. of Applied Sciences of Western Switzerland
HEIG-VD
Rte de Cheseaux 1, CP 521, CH-1401 Yverdon-les-Bains, Suisse
eric.taillard@heig-vd.ch

Abstract

An $n\ log\ n$ randomized method based on POPMUSIC metaheuristic is proposed for generating reasonably good solutions to the travelling salesman problem. The method improves a previous work which algorithmic complexity was in $n^{1.6}$. The method has been tested on instances with billions of cities. Few dozens of runs are able to generate a very high proportion of the edges of the best solutions known. This characteristic is exploited in a new release of the Helsgaun's implementation of Lin-Kernighan heuristic (LKH) that is also able to produce rapidly extremely good solutions for non Euclidean instances.

1 Introduction

The travelling salesman problem (TSP) is certainly the most studied NP-hard combinatorial optimisation problem [4, 9, 13]. Now, we are able to exactly solve instances up to several thousands of cities and to find solutions for instances with millions of cities at a fraction of percent above the optimum [1, 3, 5, 6, 8, 10].

A key point for implementing a fast and efficient local search is to use a neighbourhood of limited size containing all the pertinent moves. For the travelling salesman problem, the most efficient neighbourhoods are based on Lin-Kernighan moves. In order to speed-up the computation, only a subset of moves are evaluated. This article propose a method with a reduced algorithmic complexity for generating a limited subset of pertinent edges that must be used in the solution tour. This technique for reducing the complexity of the local search was called *tour merging* by [2].

Previous works [11, 12] proposed a 2-levels method for generating good candidate edges with a complexity in $n^{1.6}$. The method is based on 2 main steps: first, an initial tour is built with a recursive randomized procedure, with an algorithmic complexity in $n\ log\ n$. Then this initial tour is improved in linear time with a fast POPMUSIC metaheuristics.

The algorithm is discussed in Section 2. Section 3 presents numerical results showing firstly that the empirical complexity of the method is quasi-linear, as expected, and secondly that it generates a very large proportion of edges belonging to the best solution tours.

2 The $n\ log\ n$ Heuristic for the TSP

2.1 Building an initial tour

For generating an initial tour that can be further improved with a fast POPMUSIC algorithm, an arbitrary city is duplicated. Let us call c_0 and c_n this duplicated city and let $C = \{c_1 \dots c_{n-1}\}$ be the other cities of an instance. The path $P = c_0, c_1, \dots c_{n-1}, c_n$ defines a tour. Let t be the only parameter of the method (in the numerical results presented in Section 3, $t = 15$). If $n \leq t^2$, then a very good path passing once through all cities of P, starting at city c_0 and ending at city c_n can be found, for instance with a local search using Lin-Kernighan moves. A good tour is built and the method stops.

Otherwise, if $n > t^2$, a sample S of t cities is randomly chosen from C. Let $s_1 \in S$ be the city the closest to c_0 and $s_t \in S \setminus \{s_1\}$ the city the closest to c_n. A good path P_S through all the cities of sample S, starting at city s_1 and ending at city s_t can be found with a local search. Let us rename the cities of S so that $P_S = s_1, s_2, \dots, s_{t-1}, s_t$

Let us decompose all the cities of C into t clusters $C_1 \dots C_t$, so that cluster C_i contains all the cities that are closer to s_i than to the other cities of S ($i = 1 \dots t$). The initial path P can be reordered so that it starts with city c_0, then it has all the cities of cluster C_1, then those of C_2, then ..., then those of C_t, and ends with c_n.

At this step, the order of visit of the cities of a cluster is arbitrary (as it was for P at the beginning of the procedure). This order can be improved by calling recursively the procedure for each sub-path C_i ($i = 1 \dots t$), the last city of C_{i-1} playing the role of c_0 (or being actually c_0 for C_1) and the first city of C_{i+1} playing the role of c_n (or being actually c_n for C_t).

When all recursions stop, the cities are ordered in such a way that it can be successfully improved with a POPMUSIC-based heuristic. If t is considered as a fixed parameter (not depending on the problem size n), the algorithmic complexity of this procedure is $n\ log\ n$.

2.2 Improving the intial tour with a fast POPMUSIC

In [11], POPMUSIC metaheuristic is adapted for the TSP by optimizing sub-paths containing R consecutive cities of the tour, where R is a parameter. The optimization procedure is a local search based on Lin-Kernighan moves. The optimizations are repeated until there is no subset of R consecutive cities in the tour that can be improved with the Lin-Kernighan neighbourhood.

For getting good candidate edges by the tour merging technique, it is not required to run POPMUSIC until all sub-paths of R consecutive cities have been optimized. Instead, we propose to speed-up the method by optimizing the tour in 2 scans. A first scan optimizes non-overlapping sub-paths of R cities starting with the first city. Then, the tour is shifted by $R/2$ cities and a second scan optimizes sub-paths involving $R/2$ cities for each of two adjacent sub-paths in the first scan.

If R does not depends on n (we have chosen $R = t^2$ to have a method with a unique parameter), then the algorithmic complexity of the improvement with POPMUSIC is linear.

3 Computational results

3.1 Computational times

The main goal of this work was to produce moderately good solutions to TSP instances with a short computational time. In figure 1, we provide the computational time of our method as a function of the problem size. The problem instances were randomly generated in the unit square and toroidal distances are considered (as if the square was folded so that opposite border are contiguous). We provide the time for generating the initial solution and for improving it with the fast POPMUSIC presented above. In this figure, we have included the computational time for generating a tour with the previous POPMUSIC implementation of [11].

The main limitation of our method is due to the memory required for storing the problem data and the solution. A personal computer with 64Gb of main memory was able to deal with instances with 2G cities.

3.2 Solution quality

The optimal solution of randomly generated solutions in the unit square with toroidal distances was estimated by [7] as $(0.7124 \pm 0.0002)\sqrt{n}$. The quality of the solutions produced for such instances is given in Figure 2. The method proposed is able to produce solution at about 10% above optimum for all instance size. Let us mention that the nearest neighbour heuristic produces solution about 22% above optimum.

In Figure 3, we give, for few instances of the literature, the proportion of missing edges of the best solution known as a function of the number of runs of our method. We see in this figure that the union of 50 solutions produced by the method contains more than 99.9% of the edges of the best solution known.

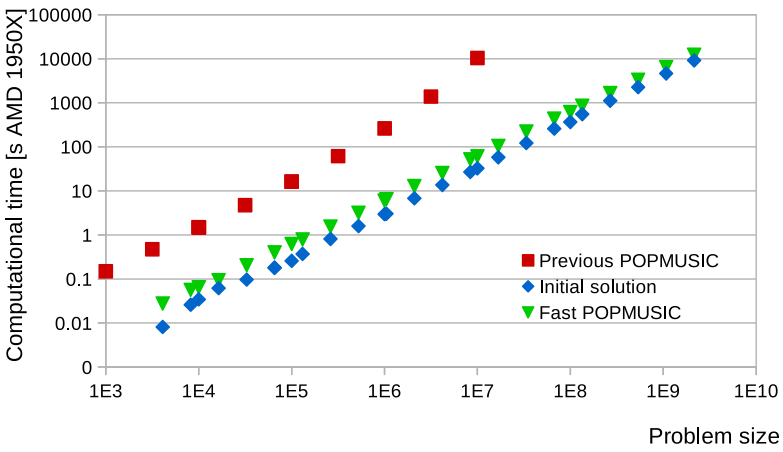


Figure 1: Computational time for producing one solution as a function of problem size.

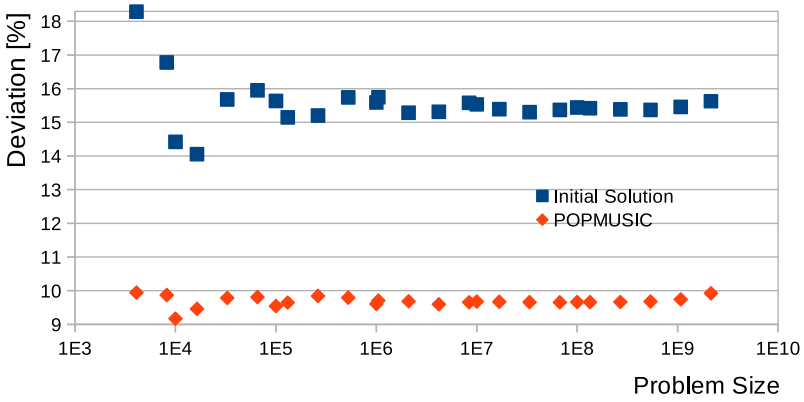


Figure 2: Quality of solutions (expressed in % above expected optimum) as a function of problem size. Uniformly distributed cities with toroidal distances in 2D.

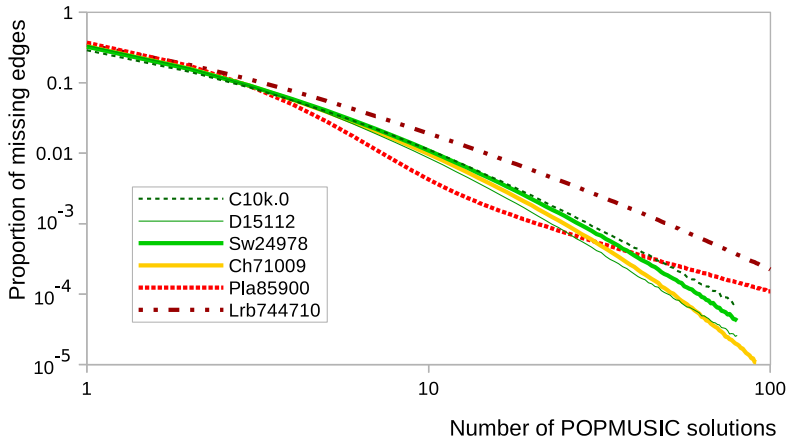


Figure 3: Proportion of missing edges as a function of the number of POPMUSIC solutions generated.

4 Conclusion

This work propose a method for generating moderately good TSP solutions in $nlogn$. The method does not make assumption about the problem structure. For the first time, to our knowledge, instances with more that a billion of cities have been tackled with a metaheuristic. The LKH 2.0.9 release includes the proposed method for generating candidate edge sets.

References

[1] David L. Applegate, Robert E. Bixby, Vasek Chvátal, and William J. Cook. Concorde: A code for solving traveling salesman problems, 1999.

[2] David L. Applegate, Robert E. Bixby, Vasek Chvátal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, USA, 2007.

[3] David L. Applegate, William Cook, and André Rohe. Chained Lin-Kernighan for Large Traveling Salesman Problems. *INFORMS Journal on Computing*, 15(1):82–92, 2003.

[4] William J. Cook. *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press, 2012.

[5] Keld Helsgaun. Best known solutions to Dimacs TSP instances. Last updated: October 6, 2014.

[6] Keld Helsgaun. General k-opt submoves for the lin-kernighan tsp heuristic. *Mathematical Programming Computation*, 1, 2009.

[7] David S. Johnson, Lyle A. McGeoch, and E. E. Rothberg. Asymptotic experimental analysis for the Held-Karp traveling salesman bound. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 341–350, 1996.

[8] Peter Merz and Jutta Huhse. An iterated local search approach for finding provably good solutions for very large tsp instances. In Günter Rudolph, Thomas Jansen, Nicola Beume, Simon Lucas, and Carlo Poloni, editors, *Parallel Problem Solving from Nature – PPSN X*, pages 929–939, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[9] Abraham P. Punnen and Gregory Gutin. *The Traveling Salesman Problem and Its Variations*. Combinatorial Optimization 12. Springer US, 1 edition, 2007.

[10] César Rego, Dorabela Gamboa, Fred Glover, and Colin Osterman. Traveling salesman problem heuristics: Leading methods, implementations and latest advances. *European Journal of Operational Research*, 211(3):427 – 441, 2011.

[11] Éric D. Taillard. TSP neighbourhood reduction with POPMUSIC. In *Metaheuristic Interantional Conference (MIC'17) proceedings*, 2017.

[12] Éric D. Taillard and Keld Heslgaun. POPMUSIC for the travelling salesman problem. *EURO Journal of Operational Research*, 272(2):420–429, 2019.

[13] Voigt, editor. *Der Handlungsreisende wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiß zu sein ; Mit einem Titelkupf.* Voigt, Ilmenau, Germany, 1832.

Optimization of Synchronizability in Power Grids Using a Second-order Kuramoto Model

Toshichika Aoki[1], Hideyuki Kato[2], Takayuki Kimura[3], and Tohru Ikeguchi[4,5]

[1] Graduate School of Electronics, Information and Media Engineering, Nippon Institute of Technology
4–1 Gakuendai, Miyashiro, Minami-Saitama, Saitama 345-8501, Japan
[2] Faculty of Science and Technology, Oita University
700 Dannoharu, Oita-city, Oita 870-1192, Japan
[3] Department of Electrical, Electronics, and Communication Engineering, Nippon Institute of Technology
4–1 Gakuendai, Miyashiro, Minami-Saitama, Saitama 345-8501, Japan
[4] Department of Management Science, Graduate School of Engineering, Tokyo University of Science
[5] Department of Information and Computer Technology, Faculty of Engineering, Tokyo University of Science
6–3–1 Niijuku, Katsushika, Tokyo 125-8585, Japan
2188001@sstu.ac.jp, h-kato@oita-u.ac.jp, tkimura@nit.ac.jp, tohru@rs.tus.ac.jp

Abstract

The synchronization of power grids is an important task for their safe operation. If the power grid enters an asynchronous state, its operation becomes unstable, which in the worst case may lead to cascading failures. Therefore, investigations on the types of power grid structures that are resistant to power outages or cascading failures are necessary to build reliable systems. A previous study investigated the optimum structure of centralized and/or decentralized power grids using the greedy algorithm; this structure is obtained by reconnections of edges while maximizing the synchronous area. However, the solution of this algorithm easily reaches the local minima. In this study, we found the optimum structure for various types of power grids using simulated annealing. Numerical experiments demonstrated that our method successfully obtains highly synchronizable power grids.

1 Introduction

Centralized power grids that provide electric power generated at centralized power plants, such as nuclear power plants, to surrounding customers are the most important infrastructure to maintain a population's comfortable lifestyle. However, renewable energy generators such as photovoltaic and wind turbine systems have rapidly and widely been introduced to the world. Consequently, power grid structures have been changing from centralized to decentralized. However, detailed investigations of robustness against attacks or failures in decentralized power grids are ongoing and various experiments have been conducted (for example, Ref. [6]). One of the experimental methods used to determine the robustness of power grids involves evaluating their synchronizability, which ensures safe operation. If a power grid enters an asynchronous state, its operations become unstable and, in the worst case, cascading failures may occur. Therefore, investigations on the types of power grid structures that are resistant to power outages or cascading failures are necessary to build reliable systems.

In power grids, a frequency that starts synchronization determines the safety of the operation. In other words, a larger synchronous area of the power grid leads to a safer operation. A previous study [4] investigated the optimum structure of centralized and/or decentralized power grids using the greedy algorithm, whereby the structure of the power grid is obtained by reconnections of the edges while maximizing the synchronous area. However, the solution found by this algorithm easily reaches the local minima. In this study, we investigate the optimum structure for various types of power grid models using simulated annealing (abbr. SA). Numerical experiments demonstrate that our method successfully obtains highly synchronizable power grids.

2 A power grid model with second-order Kuramoto model

In this study, we model a generator and consumers in the power grid using a second-order Kuramoto model [2, 6, 7]. In this model, the ith node ($i = 1, \ldots, N$) that corresponds to a generator or customer

has power P_i. If $P_i > 0$ ($P_i < 0$), the corresponding ith node is the generator (customer). In addition, the phase of the ith node is defined as follows:

$$\theta_i(t) = \Omega t + \phi_i(t), \tag{1}$$

where $\phi_j(t)$ is the phase difference to a reference frequency of the ith node at the tth time, and $\Omega = 2\pi \times 50[\mathrm{Hz}]$ or $\Omega = 2\pi \times 60[\mathrm{Hz}]$ are frequencies of the power grid.

By considering the kinetic energy and mechanical loss, we express an approximated equation of motion for the phase difference of the ith node as follows:

$$\frac{\mathrm{d}^2\phi_i}{\mathrm{d}t^2} = P_i - \alpha\frac{\mathrm{d}\phi_i}{\mathrm{d}t} + \sum_{j=1}^{N} K_{ij}\sin(\phi_j - \phi_i), \tag{2}$$

where P_i is the generating or consuming power of the ith node, α is a dissipation coefficient, and K_{ij} is defined as the transmission line capacity between the ith and the jth nodes. In this model, the electric power generated at or provided to the ith node is supplied to the jth node if a link exists between them. Furthermore, the amount of power exchanged is proportional to their phase differences, defined by $\sin(\phi_j - \phi_i)$.

To investigate the synchronous magnitude of the power grid, we used the order parameter r [3], defined as follows:

$$r = \frac{1}{N}\sum_{i=1}^{N} \mathrm{e}^{\mathrm{i}\phi_i}, \tag{3}$$

where r is the average position of oscillators on the complex plane, ψ is the average phase of the nodes, N is the total number of nodes, and ϕ_i is the phase difference of the jth node. The power grid easily synchronizes, if r is large. We defined the power grid as not synchronized if r is 0.

3 Finding the optimum power grid structure

To investigate the power grid's optimum structure, we first construct an initial solution for the SA using the following procedure.

1. We first build the ER type random networks [1] and the capacities of all transmission lines are set to the same value, namely, $K_{ij} = K$. We set K such that the order parameter r is larger than or equal to 0.99.
2. Next, we randomly select an edge and disconnect it from one of the attached nodes. The selected edge is then reconnected to a randomly selected node and r is evaluated for this reconnected power grid.
3. If the reconnected power grid has a higher r than the previous one, this structure is retained. Otherwise, the power grid returns to the previous structure.
4. Edge reconnection is repeated until r stops being updated.

A power grid obtained by the above procedure is further improved by the SA. Using SA, if the reconnection of an edge improves the r of the original power grid, the power grid is updated with the new r. Otherwise, the update of the power grid is approved according to the probability defined by $\exp(\frac{2\times10^5\times\Delta r}{T_{\mathrm{cur}}})$, which is based on an improved value $\Delta r = r_{\mathrm{improved}} - r_{\mathrm{original}}$ and a temperature parameter. We set the SA parameters as follows: $T_0 = 1$ as the initial temperature, $C = 0.995$ as the cooling rate, and $T_{\mathrm{end}} = 0.1$ as the final temperature.

4 Numerical experiments

We used two types of power grid models: large-scale and small-scale power plants. The power grid model comprised 100 nodes, which included 30 generators (10 large and 20 small power plants), 70 consumers, and 200 edges.

The parameters of Eq. (2) were set as follows: $P = 1$ for the reference power, $P_i = 5P$ for large power plants, $P_i = P$ for small power plants, $P_i = -P$ for power consumed by each consumer, and $\alpha = P$. The initial values of ϕ_i and $\dot{\phi}_i$ were set to 0. An improvement rate of the order parameter and obtained optimized structure of power grid using SA are shown in Fig. 1.

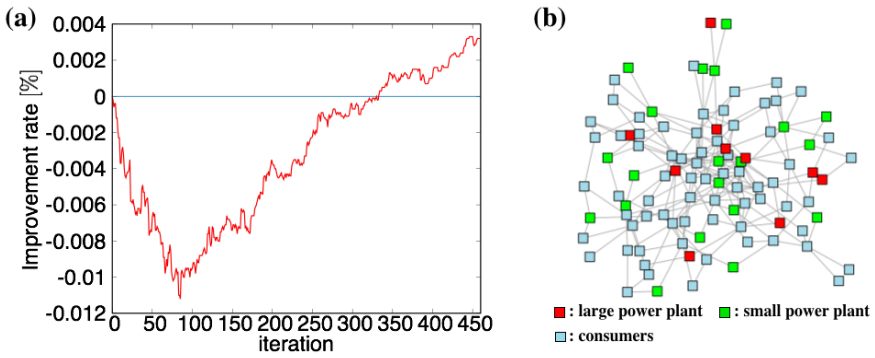


Figure 1: (a) Number of iterations versus improvement rate of the order parameter. (b) Obtained structure of the optimized power grid. Here, red squares represent large power plants, green squares represent small power plants, and sky blue squares represent consumers.

In Fig.1(a), our procedure improves the order parameter r by $3.5 \times 10^{-3}\%$ compared to the initial power grid. These results indicate that the optimized power grid has higher synchronizability than the initial one. We then investigate the topological differences between the initial power grid and the optimized one by using the clustering coefficient in the complex network theory. The cluster coefficient measures the ratio of including triangle relationships in the networks. Our investigation clarified that the optimized power grid increases the clustering coefficient by 3.0×10^{-3}. These results indicate that the optimized power grid has higher robustness because the triangles relationships have tolerance towards transmission line failures of the power grid [5]. In future works, we will propose a power grid structure optimization method based on centrality measures in complex network theory.

5 Acknowledgment

The research of T.K. and H.K. was partially supported by a Grant-in-Aid for Young Scientists (B) from JSPS (No.16K21327, No.16K16127) and the research of T. I. was supported by JSPS KAKENHI Grant Numbers JP15TK0112 and JP17K00348.

References

[1] P. Erdös and A. Rényi. *Publicationes Mathematicae Debrecen*, 6:290, 1959.

[2] G. Filatrella, A. H. Nielsen, and N. F. Pedersen. *The European Physical Journal B*, 2008.

[3] Y. Kuramoto. In H. Araki, editor, *International Symposium on Mathematical Problems in Theoretical Physics*, pages 420–422, 1975.

[4] Rafael S. Pinto and Alberto Saa. *Physica A: Statistical Mechanics and its Applications*, 463:77 – 87, 2016.

[5] M. Rohden, A. Sorge, D. Witthaut, and M. Timme. *Chaos*, 24(1):013123–9, 2014.

[6] B. Schäfer, D. Witthaut, M. Timme, and V. Latora. *Nature Communications*, 9(1):1–13, 2018.

[7] D. Witthaut and M. Timme. *New Journal of Physics*, 14(8):083036, 2012.

Planning of 'last-mile' delivery for a Colombian dairy company using a biased-randomized multi-start algorithm

Carlos L. Quintero Araújo[1], Carlos A. Vega-Mejía[1], Andrés Muñoz-Villamizar[1]

[1] Operations & Supply Chain Management Research Group, Universidad de La Sabana
Campus del Puente del Común, Km. 7, Autopista Norte de Bogotá. Chía, Cundinamarca, Colombia.
{carlos.quintero5 ; carlosveme ; andres.munoz1}@unisabana.edu.co

Abstract

This document focuses on the 'last-mile' distribution of dairy products of a company in Colombia. To improve the current operational and economic performance of the company a heuristic solution method is proposed to solve a Vehicle Routing Problem with Time Windows. This problem characterizes the 'last-mile' delivery problem that the dairy company faces on its day-to-day operation. The proposed heuristic method consists of a multi-start algorithm with biased randomization and is validated utilizing historical data from the Company. Computational results show that, employing the proposed heuristic method, a more accurate planning of delivery routes can be achieved in a shorter processing time than with the current manual procedure the company uses. The proposed solution procedure increases the percentages of space utilization of the vehicles required for the 'last-mile' delivery task. This increase represents monetary savings for the Company.

1 Introduction

The distribution or transportation of products is a critical component within the operations of a supply chain [1]. It is such a crucial element within supply chain management, that transportation has been considered an irreplaceable fundamental infrastructure for economic growth [2]. This study focuses on the 'last-mile' distribution of the dairy products of mass consumption produced by a company in Colombia. Currently, daily distribution is programmed in an empirical way mostly relying on the expertise of the company's employees in charge of designing the delivery routes. This planning methodology has generated operational and economic inefficiencies for the company, such as an increase of total costs of the logistics operation and assignment of high workloads to the crews of the vehicles. To resolve these issues a heuristic solution method is proposed to solve a capacitated Vehicle Routing Problem with Time Windows (VRPTW), which characterizes the 'last-mile' delivery problem the dairy company faces on its day-to-day operation. The proposed heuristic method consists of a multi-start algorithm with biased randomization and is tested with real data from the company under study. To test the performance and impact of the proposed heuristic solution, its computational results are compared with actual distribution plans of the company.

The paper is organized as follows. Section 2 provides a description of the Colombian dairy company and a brief literature review of VRPTWs. Section 3 presents the proposed heuristic solution methodology. Section 4 presents the computational results and analyzes the performance of the proposed heuristic method with real historic data from the dairy company. Lastly, Section 5 presents some concluding remarks.

2 Background

This section provides, first, a brief description of the Colombian company that was selected for this study. For confidentiality reasons the name of the company will remain undisclosed throughout the paper. Second, this section provides a brief review of recent and relevant studies available in the literature that are related to the VRPTW and to solution methods that have been proposed.

2.1 The Company

The company selected for this study produces dairy products and is one of the leaders in the Colombian market. Currently, the company has six production plants located in different regions of Colombia. Each plant has its own distribution center from where the distribution and commercialization of the products

is carried out. In addition, the company has seven satellite distribution centers located in other regions of the country. The distribution is carried out in two phases: a primary distribution and a secondary distribution. This study focuses on the latter. The secondary distribution, which encompasses the 'last-mile' delivery process, takes the product from a satellite distribution center and delivers it to customers (e.g. supermarkets, or grocery and convenient stores). The process of the secondary distribution is summarized in Figure 1.

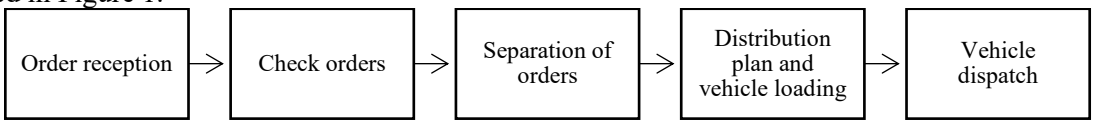


Figure 1: Secondary distribution of the Company

2.2 Literature review

The problem of the Company studied in this document can be represented as a capacitated VRPTW. The VRPTW is one of the many variants of the original VRP problem and is considered the core of the routing problems [3]. Given the NP-Hard nature of the problem [4], solution methods are often directed towards the development of heuristic procedures, although these cannot guarantee optimal solutions [5]. In this regard, [6] presented an Adaptive Large Neighborhood Search to solve a VRPTW with multiple objectives (i.e. minimization of distance and total costs). [7] proposed a Bee Colony Algorithm for minimizing the costs of labor and fuel of a VRPTW. [8] proposed a hybrid heuristic that combined Tabu Search and Variable Neighborhood Search to minimize the distance travelled in Rich VRPs. [9] proposed a hybrid heuristic that combined Tabu Search and Genetic Algorithms to solve a VRPTW with stochastic demands.

Other solution approaches are based on biased randomization, which is a method that has been utilized recently by several researchers [10]. Examples of such applications can be found in studies that solve VRPs with loading constraints, which can be considered a special case of Rich VRPs. For instance, [11] proposed a biased-randomized heuristic for solving a Location Routing Problem. Other studies applied a biased randomized version of the Clarke and Wright Savings Algorithm (CWSA) in the solution methodology for 2D-VRPs (e.g. [12]). Other solution methods for 2D-VRPs have combined this biased-randomized CWSA with other heuristics, such as Iterated Local Search (e.g. [13]), and Large Neighborhood Search (e.g. [14]). The proposed solution method for the Company presented in this study is based on the concept of Biased-Randomization.

3 Solution methodology

To solve the problem of the company a multi-start algorithm is proposed. This procedure consists of a biased randomization of the Nearest Neighbor heuristic. Figure 4 presents the proposed algorithm.

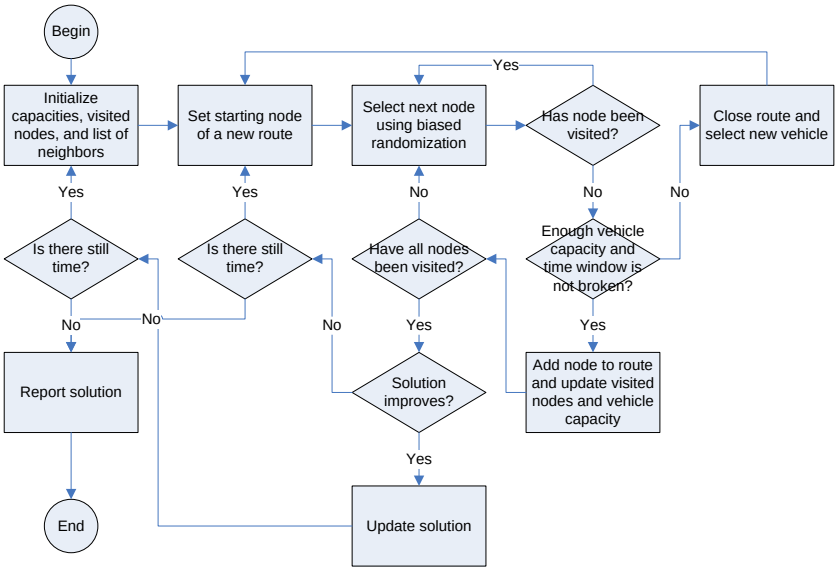


Figure 2: Proposed algorithm

The input parameters of the algorithm are: The maximum running time (in seconds) of the algorithm, the probability of skewed randomization, the maximum duration of the routes in minutes, and a random seed. The biased randomization process will assign a probability to the nodes not yet added to a vehicle route. This is employed to avoid always selecting the nearest node from those nodes that have not been visited yet. However, a higher probability is awarded to the nearest node, so that the nature of the Nearest Neighbor heuristic is somewhat maintained. It is also worth noting that the time window constraints are strictly enforced. This means that late deliveries are not allowed.

4 Computational results

The proposed algorithm was programmed using Visual Basic for Applications from Microsoft Excel. The maximum running time of the algorithm was set at 120 seconds. A geometric distribution was employed for the biased randomization process, and a probability of 0.8 was given to the nearest node. And the maximum duration for each delivery route was set at 660 minutes. The proposed algorithm was tested utilizing historical data from the Company, which corresponded to a week in the first semester of operations of 2018. The results, in terms of required vehicles and average utilization of the available space within the container of the vehicles, is presented in Table 1.

Day of operation	Under current plan		Using the proposed algorithm	
	Vehicles required	Space utilization	Vehicles required	Space utilization
1	33	57%	18	83%
2	34	47%	26	47%
3	38	42%	21	74%
4	32	49%	31	53%
5	32	57%	28	66%
6	38	55%	26	71%

Table 1: Computational results

5 Conclusions

The current programming process for ‘last-mile’ delivery of a dairy company in Colombia is performed manually and greatly depends on the experience and knowledge of the people in charge of planning the delivery routes. This form of planning hinders the possibility of finding efficient delivery routes for the Company. To address these issues, a heuristic solution procedure that employs biased randomization techniques has been proposed. The proposed multi-start algorithm was validated utilizing historical data from the Company and results showed that the proposed solution procedure increases the occupation percentages of the vehicles required for the ‘last-mile’ delivery task by approximately 30%. This translates to a higher average of products being transported per vehicle. It also represents monetary savings for the Company as the costs of renting transporting vehicles would be significantly reduced, as fewer vehicles are required to meet the demand from the customers.

References

[1] Tsao, Y.-C., & Lu, J.-C. (2012). A supply chain network design considering transportation cost discounts. Transportation Research Part E: Logistics and Transportation Review, 48(2), 401–414. http://doi.org/10.1016/j.tre.2011.10.004.

[2] Lin, C., Choy, K. L., Ho, G. T. S., Chung, S. H., & Lam, H. Y. (2014). Survey of green vehicle routing problem: Past and future trends. Expert Systems with Applications, 41(4), 1118–1138. http://doi.org/10.1016/j.eswa.2013.07.107.

[3] Schneider, M., Schwahn, F., & Daniele, V. (2017). Designing granular slution methods for routing problems with time windows. *European Journal of operation Research*, 16.

[4] Cordeau, J. F., Desaulniers, G., Desrosiers, J., Solomon, M., & Soumis, F. (1999). *The VRP with*

Time Windows. Montreal: Gerad.

[5] Zanakis, S. H., & Evans, J. R. (1981). Heuristic "Optimization": Why, When, and How to Use It. *Interfaces*, 84-91.

[6] Goeke, D., & Schneider, M. (2015). Routing a mixed fleet of electric and conventional vehicles. European Journal of Operational Research, 245(1), 81–99. http://doi.org/10.1016/j.ejor.2015.01.049.

[7] Adaptive memory artificial bee colony algorithm for green vehicle routing with cross-docking. Applied Mathematical Modelling, 40, 9302–9315. http://doi.org/10.1016/j.apm.2016.06.013.

[8] Sicilia, J. A., Quemada, C., Royo, B., & Escuin, D. (2016). An optimization algorithm for solving the rich vehicle routing problem based on variable neighborhood search and tabu search metaheuristics. Journal of Computational and Applied Mathematics, 291, 468–477. http://doi.org/10.1016/j.cam.2015.03.050.

[9] Liao, T.-Y. (2017). On-line vehicle routing problems for carbon emissions reduction. Computer-Aided Civil and Infrastructure Engineering, 32(12), 1047–1063. http://doi.org/10.1111/mice.12308.

[10] Faulin, J. A., Ferrer, J., H.R., A. L., & Barros, B. (2013). MIRHA: Multi-start biased randomization of heuristics with adaptive local search for solving non-smooth routing problem. *TOP*, 109 - 132.

[11] Quintero-Araujo, C. L., Caballero-Villalobos, J. P., Juan, A. A. and Montoya-Torres, J. R. (2017). A biased-randomized metaheuristic for the capacitated location routing problem. Intl. Trans. in Op. Res., 24: 1079-1098. doi:10.1111/itor.12322.

[12] Guimarans, D., Dominguez, O., Juan, A. A., & Martinez, E. (2016). A multi-start simheuristic for the stochastic two-dimensional vehicle routing problem. In T. Roede, P. Frazier, R. Szechtman, & E. Zhou (Eds.), 2016 Winter Simulation Conference (WSC) (pp. 2326–2334). Washington, US: IEEE Publishing. http://doi.org/10.1109/WSC.2016.7822273.

[13] Dominguez, O., Juan, A. A., de la Nuez, I., & Ouelhadj, D. (2016). An ILS-biased randomization algorithm for the two-dimensional loading HFVRP with sequential loading and items rotation. Journal of the Operational Research Society, 67, 37–53. http://doi.org/10.1057/jors.2015.48.

[14] Dominguez, O., Guimarans, D., Juan, A. A., & de la Nuez, I. (2016). A biased-randomised large neighbourhood search for the two-dimensional vehicle routing problem with backhauls. European Journal of Operational Research, 255(2), 442–462. http://doi.org/10.1016/j.ejor.2016.05.002.

Cuckoo Search Algorithms for the Cardinality Portfolio Optimization Problem

Ibrahim H. Osman[1], Soha Maad[2], Karl Sawaya[1]

1-American University of Beirut
Olayan School of Business
Beirut, Lebanon
ibrahim.osman@aub.edu.lb
2- IGROW EEIG, Integrated Technologies and Services For Sustainable Growth European Economic Interest Grouping, Dublin, Ireland, sm@sohamaad.net, manager@igrow-eeig.com

Abstract

This paper presents Cuckoo Search (CS) Algorithms for solving approximately the cardinality portfolio optimization (CPO) problem. The paper presents a review on existing literature on both CPO and CS to highlight area of concern. A hybrid combination of CS method and an exact method is proposed where CS explores the cardinality search space using Levy flights to select the desired assets while exact method determines the optimal allocation of investments for the selected assets. The main contributions include the introduction of a new mapping between the CS continuous search space and the integer sequencing search space to guide the selection of assets. The CS algorithm is implemented and the obtained results compare favorably to previously published ones on datasets of instances of varying sizes from the literature. The paper concludes with remarks and research directions.

Keywords
optimization, portfolio, Cuckoo Search, selection, assets

Algorithms the min-max regret 0-1 Integer Linear Programming Problem with Interval Data

Iago A. Carvalho[1], Thiago F. Noronha[1], Christophe Duhamel[2]

[1] Universidade Federal de Minas Gerais
Department of Computer Science, Belo Horizonte, Brazil
{iagoac,tfn}@dcc.ufmg.br

[2] Université Clermont Auvergne
Laboratoire d'Informatique (LIMOS), Clermont-Ferrand, France
christophe.duhamel@isima.fr

Abstract

We address the Interval Data Min-Max Regret 0-1 Integer Linear Programming problem (MMR-ILP), a variant of the 0-1 Integer Linear Programming problem where the objective function coefficients are uncertain. We solve MMR-ILP using a Benders-like Decomposition Algorithm and two metaheuristics for min-max regret problems with interval data. Computational experiments developed on variations of MIPLIB instances show that the heuristics obtain good results in a reasonable computational time when compared to the Benders-like Decomposition algorithm.

1 Introduction

The 0-1 Integer Linear Programming problem (ILP, for short) is a well-known NP-Hard mathematical optimization problem, with linear objective function and constraints, in which the domain of all variables is $\{0, 1\}$. An ILP can be formulated by the objective function (1) and the constraints (2) and (3), where b and c are n-dimensional vectors of coefficients, A is a $m \times n$-dimensional matrix of coefficients, and x is a n-dimensional vector of binary variables.

$$\text{(ILP)} \quad \min c^T x \tag{1}$$

$$Ax \leq b \tag{2}$$

$$x \in \{0, 1\}^n \tag{3}$$

This abstract deals with problems where the coefficients in c are uncertain. We deal with the Interval Data Min-Max Regret 0-1 Integer Linear Programming problem (MMR-ILP, for short). In this problem, the value of the coefficient c_i, for all $i \in \{1, \ldots, n\}$, is unknown. However, it is assumed that the value of c_i is in the range $[l_i, u_i]$. A *scenario* is defined as a vector $S = (c_1^S, \ldots, c_n^S)$, where c_i^S is any real value in the interval $[l_i, u_i]$, i.e. a scenario corresponds to a valid assignment of values to the coefficients of variables x. There are infinitely many scenarios and MMR-ILP aims at finding a solution that is robust to all of them. MMR-ILP is also NP-Hard optimization problem, since the min-max regret version any problem has, at least, the same complexity of its deterministic counterpart [1].

MMR-ILP can be formally defined as follows. Let Γ be the set of all scenarios, and Φ be the set of all feasible solutions to the constraints in (2) and (3). The *regret* of a solution $x \in \Phi$ in a scenario $S \in \Gamma$ is the difference between the cost of x in the scenario S and the cost of the optimal solution y^S for the scenario S, i.e. it is the loss of using x instead of y^S if the scenario S occurs. The cost of x in S is denoted by $F(x, S) = \sum_{i=1}^{n} c_i^S x_i$, while the cost of y^S is denoted by

$$F(S) = \min_{y \in \Phi} F(y, S) = \min_{y \in \Phi} \sum_{i=1}^{n} c_i^S y_i.$$

The *robustness cost* $Z(x)$ of a solution $x \in \Phi$ is defined as the maximum possible regret of x among all scenarios in Γ, i.e.

$$Z(x) = \max_{S \in \Gamma} \{F(x, S) - F(S)\}.$$

Despite the fact that $|\Gamma| = \infty$, the scenario where the regret of x is the maximum is the scenario S^x, such that $c_i^{S^x} = l_i + (u_i - l_i)x_i$, i.e. $c_i^{S^x} = u_i$ if $x_i = 1$, and $c_i^{S^x} = l_i$ otherwise [1]. From this result, we have that

$$F(x, S^x) = \sum_{i=1}^{n} u_i x_i,$$

$$F(y^{S^x}, S^x) = \min_{y \in \Phi} \sum_{i=1}^{n} \big(l_i + (u_i - l_i)x_i\big) y_i,$$

and

$$Z(x) = F(x, S^x) - F(y^{S^x}, S^x).$$

It is worth noticing that $F(y^{S^x}, S^x)$ is still an ILP as in this case x_i is constant. Therefore, the robust cost of a solution x can be computed by solving a single ILP problem in the scenario S^x. MMR-ILP aims at finding the solution with minimum robustness cost, i.e.

$$\min_{x \in \Phi} Z(x) = \min_{x \in \Phi} \left\{ F(x, S^x) - F(y^{S^x}, S^x) \right\}.$$

For the general case, an mathematical formulation for MMR-ILP can obtained by replacing $F(y^{S^x}, S^x)$ with a free variable θ and adding a new set of linear constrains that bounds the value of θ to the value of $F(y^{S^x}, S^x)$. The resulting formulation (4)-(6) has an exponentially large number of constraints.

$$\min_{x \in \Phi} \sum_{i=1}^{n} u_i x_i - \theta \tag{4}$$

$$\theta \leqslant \sum_{i=1}^{n} \left(l_i + (u_i - l_i)x_i\right) y_i, \quad \forall\, y \in \Phi \tag{5}$$

$$\theta \in \mathbb{R} \tag{6}$$

2 Heuristics for the MMR-ILP

We solved the MMR-ILP using two metaheuristics for interval data min-max regret optimization problems: (i) the Algorithm Mean Upper (AMU) [4]; and (ii) the Scenario-Based Algorithm (SBA) [2, 3]. They are described as follows.

2.1 Algorithm Mean Upper

AMU is a 2-approximative heuristic for interval data min-max regret optimization problems. It solves the MMR-ILP into two specific scenarios: the *mean scenario* s^m, where the cost of each uncertain coefficient is set to its mean value, *i.e.* $c_{ij}^{s^m} = \frac{l_{ij}+u_{ij}}{2}$, and the *upper scenario* s^u, where the cost of each uncertain coefficient is set to its upper value, *i.e.* $c_{ij}^{s^u} = u_{ij}$. AMU computes the robustness cost of the computed solution in each scenario and returns the one which wields the smallest value.

2.2 Scenario-Based Algorithm

SBA is an extension of AMU which inspects a larger number of scenarios. Target scenarios between the lower scenario $s^l \in \Gamma$ (a scenario where the cost of the arcs are set to their respective lower, *i.e.* $c_{ij}^{s^l} = l_{ij}$) and the upper scenario (s^u) are investigated. SBA relies on three parameters: the initial scenario α; the final scenario β; and the step size γ. All parameters are real-valued in the interval $[0, 1]$. Target scenarios are computed as $\alpha + \delta\gamma$, for all $\delta \in \{0, \ldots, i\}$ such that $\alpha + \delta\gamma \leq \beta$. Thus, SBA investigates $\frac{\beta-\alpha}{\gamma}$ different scenarios. One can see that both the mean scenario s^m and the upper scenario s^u are considered by SBA. Thus, SBA produces solutions at least as good as AMU and also holds an approximation ratio of at most 2 for MMR-ILP. The SBA for MMR-ILP uses the parameter settings recommended in Coco et al. [3], being $\alpha = 0.5, \beta = 1.0$, and $\gamma = 0.05$. Therefore, it inspects a total of 11 scenarios.

3 Computational experiments

Computational experiments were carried out on a single core of an Intel Xeon CPU E5645 with 2.4 GHz clock and 32 GB of RAM, running under the operating system Linux Ubuntu. ILOG CPLEX solver version 12.6 is used with default parameters. We employed variations of the classic MIPLIB instances, being the coefficients interval generated as by Carvalho et al [2].We assess the quality of AMU and SBA by comparing their results with the primal bound given by the Benders-like Decomposition Algorithm (BDA) [5], one of the most successfully exact algorithms for interval data min-max regret optimization problems. We limited the running time of all algorithms to 7200 seconds.

Table 1 show the results of this experiment. The first column reports the BLD average running time and it's standard deviation. The second column shows the AMU average relative deviation regarding the BLD upper bound, computed as $\frac{AMU-BLD}{BLD}$. It also shows the standard deviation deviation of this same metric. The third column presents the AMU average running time and its standard deviation. We show the same information for SBA on the remaining columns.

BLD	AMU		SBA	
time (s)	dev (%)	time (s)	dev (%)	time (s)
4709 ± 4066	10.39 ± 22.16	201 ± 290	9.00 ± 0.20	1983 ± 2342

Table 1: Results for BLD, AMU, and SBA when solving the proposed MMR-ILP instances

One can see from Table 1 that BLD takes an average running time of almost 5000 seconds to run. AMU and SBA relative deviations are very close to each other. However, SBA running time is nearly ten times greater than of AMU.

We performed a Wilcoxon Sign-Rank Test to verify if there is a significant difference between AMU and SBA relative deviations. The Wilcoxon Test showed that both results do not significantly differ from each other ($p > 0.05$). Therefore, we can conclude that AMU performs better than SBA when solving the proposed MMR-ILP instances since it has a smaller average running time and their relative deviation do not significantly differ.

References

[1] H. Aissi, C. Bazgan, and D. Vanderpooten. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438, 2009.

[2] Iago A Carvalho, Thiago F Noronha, Christophe Duhamel, and Luiz FM Vieira. A scenario based heuristic for the robust shortest path tree problem. *IFAC-PapersOnLine*, 49(12):443–448, 2016.

[3] Amadeu A Coco, Andréa Cynthia Santos, and Thiago F Noronha. Senario-based heuristics with path-relinking for the robust set covering problem. In *Proceedings of the XI Metaheuristics International Conference (MIC)*, 2015.

[4] Adam Kasperski and Paweł Zieliński. An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Information Processing Letters*, 97(5):177–180, 2006.

[5] Roberto Montemanni and Luca Maria Gambardella. The robust shortest path problem with interval data via benders decomposition. *4OR: A Quarterly Journal of Operations Research*, 3(4):315–328, 2005.

Iterated Local Search for the Periodic Location Routing Problem with Fixed Customer-Depot Assignment

Hadir A. G. Castro[1], André B. Mendes[1], Yue Wu[2]

[1] Department of Naval Architecture and Ocean Engineering, University of Sao Paulo
Av Prof. Mello Moraes, 2231, São Paulo, SP, 13026-001, Brazil
hadir.garcia@usp.br, andbergs@usp.br

[2] Southampton Business School, University of Southampton
University Road, SO171 BJ, Southampton, UK
y.wu@soton.ac.uk

Abstract

This research proposes an iterated local search (ILS) algorithm for solving the periodic location routing problem, with the additional consideration that each customer must remain assigned to the same depot throughout the planning horizon. This is a practical consideration found in many real-world applications, that has not been properly considered in existing methods. The ILS algorithm will be solved for different network configurations, generated by a MIP assignment model. Instances from the literature will be solved and indicates that the proposed solution method is effective in dealing with this problem.

1 Introduction

When solving facility location problems, a well-known approach is to simultaneously combine location and routing decisions in the so-called location routing problem [8]. If the customers must be served one or more times within a planning horizon (e.g. a week) to have their demands fulfilled, then the problem is called a periodic routing problem [1]. The combination of the periodic routing with location decisions results in the periodic location routing problem (PLRP), as described in [3, 4, 7].

In the PLRP, several decisions must be taken such as the depots to open, the fleet to assign to each depot, the visiting periods (visiting days) for each customer, and the routes the vehicles will perform on each period. After having decided which depots to open and the customers visiting periods, it must be solved a multi-depot vehicle routing problem (MDVRP) [10] for each period. However, solving the MDVRP will not guarantee that a customer will remain assigned to the same depot throughout the planning horizon. In the offshore oil industry, for example, each offshore unit is served by a fixed onshore base [2], which is responsible for managing the supply operations. In the beverage industry, customers belonging to a determined district are covered by a fixed depot due to administrative and commercial reasons [9]. Therefore, it is important to devise solution methods that take this aspect into consideration.

The periodic location routing problem with fixed customer-depot assignment (PLRP-FA), is a challenging variation of the PLRP. Instead of solving a MDVRP for each period, the classical vehicle routing problem (VRP) is solved for each open depot and for each period; moreover, it must be decided, for each customer, the depot it will remain assigned during the whole planning horizon.

One challenging feature in any heuristic method focused on solving location problems is related to the subset of depots that must be opened, given its impact on the solution and on the remaining decisions. In this research, a MIP assignment model will be used to generate promising network configurations (depots to open), and an iterated local search (ILS) method [5] will be called for the identified network configurations, working on the remaining decisions such as fleet size, customer-depot assignment, visiting periods for each customer and the daily routes.

The existing literature on PLRP in qualified journals, although relevant, is very scarce. In [7], an iterated local search was proposed with the following modification: in the diversification phase, more than one solution is generated, and the best solution is chosen for the following iteration. The author also used an extended version of the Clarke & Wright algorithm to allow working with multiple depots, and tested different rules related to the customer-depot assignment. In [3], a large neighborhood search

was proposed with several destruction and repair operators and the annealing acceptance scheme. The method was implemented with parallelization to allow using multiple processors. In [4], a more general adaptive large neighborhood search was implemented to solve other variants of the PLRP.

2 Solution Method for the PLRP-FA

The proposed solution method has two main phases: 1) generation of network configurations; 2) optimization of each configuration through an iterated local search algorithm.

To generate network configurations a compound assignment model (CAM), derived from [1], is solved. Let I be the set of candidate depots; let J be the set of customers; let L be the set of periods (delivery days); let O_i be the fixed cost for opening depot i; let W_i be the capacity of depot i; each vehicle has a fixed cost F and a capacity Q; let $Comb_j$ be the set of visiting patterns r (i.e. a set of visiting days) drawing up from a given frequency of visits $freq_j$ for customer j (e.g. for $freq_j = 3$, visiting pattern $r = \{\text{Mon, Wed, Fri}\}$); let d_{jlr} be the demand of customer j on period l under pattern r; let a_{rl} be a binary parameter that equals one if period l belongs to pattern r; the traveling cost c_{ij} is twice the radial distance from depot i to customer j multiplied by $freq_j$. The binary decision variables $y_i = 1$ if depot i is opened, $f_{ijr} = 1$ if customer j is assigned to depot i with pattern r; the integer decision variables T_i are the fleet size assigned to depot i.

An α parameter adjusts overestimation of the actual routing costs generated by the influence of the radial distance costs in the objective function. As the resulting α value in an optimal solution of the PLRP-FA is unknown, varying $0.05 \leq \alpha \leq 0.95$ generates different network configurations that serve as input to the second phase of the proposed solution method. The CAM formulation is as follows:

$$\min Z = \sum_{i \in I} O_i y_i + \sum_{i \in I} F T_i + \alpha \sum_{i \in I} \sum_{j \in J} \sum_{r \in Comb_j} c_{ij} f_{ijr} \quad (1)$$

Subject to

$$\sum_{i \in I} \sum_{r \in Comb_j} f_{ijr} = 1 \quad \forall j \quad (2)$$

$$f_{ijr} \leq y_i \quad \forall i, \forall j, \forall r \in Comb_j \quad (3)$$

$$\sum_{j \in J} \sum_{r \in Comb_j} f_{ijr} a_{rl} d_{ilr} \leq W_i y_i \quad \forall i, \forall l \quad (4)$$

$$\sum_{j \in J} \sum_{r \in Comb_j} f_{ijr} a_{rl} d_{ilr} \leq Q T_i \quad \forall i, \forall l \quad (5)$$

$$y_i \in \{0,1\}, f_{ijr} \in \{0,1\}, T_i \subset \mathbb{N} \quad \forall i \in I, \forall j \in J, \forall r \in Comb_j \quad (6)$$

The objective function (1) minimizes the sum of all location costs, fleet size costs and routing costs. Constraints (2) assign a depot and a visiting pattern for each customer. Constraints (3) ensure that customers are assigned to open depots. The set of constraints (4) and (5) imposes capacity restrictions. Finally, constraints (6) define the domain for the binary and integer decision variables.

Each network conguration generated by the CAM is submitted to an ILS comprising the following steps: 2.1) *feasible initial solution generation* – based on f_{ijr} decision variables, a TSP is solved for each depot and for each period, producing a giant tour that is split into routes; 2.2) *local search* – the following operators are tested: a) intra-route and inter-route improvements considering all the movements described in [6]; b) customer-depot assignment modification; c) visiting pattern modication. In cases b and c, one to three customers are randomly excluded from the solution and are re-inserted in their best positions; 2.3) *perturbation* – consists of: a) maintain the current solution and add one vehicle to a depot; b) maintain the current fleet but modify the customer-depot assignment and/or the visiting pattern to a number of customers. Infeasible solutions are admitted regarding the vehicle capacity, through a penalization in the objective function. The number of perturbation phases is the stopping criterion.

3 Computational Results

Instances from [7] were solved. Table 1 indicates the instance identifier together with the instance size (customers-depots), the objective function value obtained with the proposed method, the CPU time in seconds and the percentual difference to the results found in [7]. Our results can be compared with [7], as the author did not vary the customer-depot assignment throughout the planning horizon.

Instance	ILS	CPU	Diff %	Instance	ILS	CPU	Diff %	Instance	ILS	CPU	Diff %
1 [20-5]	78,726	18.2	−0.02	11 [50-5]	155,250	98.7	−1.70	21 [100-10]	257,615	440.3	−8.14
2 [20-5]	75,690	6.3	+0.18	12 [50-5]	110,060	46.5	−3.98	22 [100-10]	166,526	296.3	−5.60
3 [20-5]	78,796	11.0	−1.98	13 [100-5]	343,031	384.4	−5.36	23 [100-10]	255,238	654.7	−6.73
4 [20-5]	62,491	8.2	−1.09	14 [100-5]	223,946	363.1	−4.61	24 [100-10]	190,323	350.0	−7.11
5 [50-5]	148,907	26.0	−4.90	15 [100-5]	264,746	222.7	−6.02	25 [200-10]	439,674	1,079.4	−7.63
6 [50-5]	137,691	77.3	−3.47	16 [100-5]	164,071	217.3	−2.78	26 [200-10]	370,951	944.5	−6.94
7 [50-5]	138,486	77.5	−3.79	17 [100-5]	219,194	107.4	−5.97	27 [200-10]	384,629	439.9	−4.79
8 [50-5]	111,235	80.0	−5.26	18 [100-5]	164,505	93.9	−6.19	28 [200-10]	312,061	428.6	−3.92
9 [50-5]	169,391	39.6	−4.58	19 [100-10]	262,372	264.6	−5.79	29 [200-10]	539,201	1,184.2	−4.96
10 [50-5]	98,044	33.1	−4.60	20 [100-10]	208,834	738.2	−5.13	30 [200-10]	343,500	620.1	−4.86

Table 1: Objective function values of the ILS for the Prodhon (2011) instances

4 Concluding Remarks

It was studied the periodic location routing problem with the additional consideration of a fixed customer-depot assignment throughout the planning horizon. The problem was solved in two stages, with a MIP assignment model followed by an ILS heuristic. The MIP model was responsible for generating promising network configurations that were further explored by the ILS. An average reduction of -4.59% was achieved in comparison with [7]. Future work involves the parallelization of the whole solution method.

Acknowledgements: This research was partially funded by FAPESP – The Sao Paulo Research Foundation, under grant #2014/09342-4. This support is gratefully acknowledged.

References

[1] Nicos Christofides and John Beasley. The period routing problem. *Networks*, 14(2):237–256, 1984.

[2] Roberto Cruz, André B. Mendes, Laura Bahiense, and Yue Wu. Integrating berth allocation decisions in a fleet composition and periodic routing problem of platform supply vessels. *European Journal of Operational Research*, 275(1):334–346, in press, 2019.

[3] Vera C. Hemmelmayr. Sequential and parallel large neighborhood search algorithms for the periodic location routing problem. *European Journal of Operational Research*, 243(1):52–60, 2015.

[4] Çar Koç. A unified-adaptive large neighborhood search metaheuristic for periodic location-routing problems. *Transportation Research Part C: Emerging Technologies*, 68:265–284, 2016.

[5] Helena R. Lourenço, Olivier C. Martin, and Thomas Stützle. Iterated Local Search: Framework and Applications. pages 363–397. Springer, Boston, MA, 2010.

[6] Christian Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002, 2004.

[7] Caroline Prodhon. A hybrid evolutionary algorithm for the periodic location-routing problem. *European Journal of Operational Research*, 210(2):204–212, 2011.

[8] Said Salhi and Graham K. Rand. The effect of ignoring routes when locating depots. *European Journal of Operational Research*, 39(2):150–156, 1989.

[9] Michel P. Seixas and André B. Mendes. Column Generation for a Multitrip Vehicle Routing Problem with Time Windows, Driver Work Hours, and Heterogeneous Fleet. *Mathematical Problems in Engineering*, 2013:1–13, 2013.

[10] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems. *Operations Research*, 60(3):611–624, 2012.

A heuristic approach for the combined inventory routing and crew scheduling problems

Carlos Franco [1], Eduyn López-Santana[2],

[1] Universidad del Rosario, Escuela de Administración
Calle 200 entre carrera 7 y autopista norte, Bogotá D.C., Colombia
carlosa.franco@urosario.edu.co

[2] Universidad Distrital Francisco José de Caldas
Carrera 7 No. 40b - 53, Bogotá D.C., Colombia
erlopezs@udistrital.edu.co

Abstract

We study the problem of combined inventory routing and crew scheduling problems. To solve, we propose a heuristic that consists in solving three mathematical models: inventory model, vehicle routing model and personnel allocation model. We state the general algorithm and main assumptions for our method and present preliminary results of a set of instances.

1 Introduction

Inventory and routing are two main interrelated decisions in logistics networks, in [1] an application of combined decisions of transportation and inventory management in a gas company is presented. From the development of this Inventory Routing application, several computational and real life applications have been developed [2]–[6]. One of the most important related works is presented in [7] where authors develop a first exact model based on branch and cut strategies, also evaluating two inventory strategies: order-up to level and maximum level. Besides Inventory and Routing decisions, shift scheduling and crew scheduling are operational problems that affects the planning process. [8] can be considered as one of the first works that combines inventory and routing decisions with crew scheduling.

In this research we propose an extension of the classical Inventory Routing Problem by including the crew scheduling problem considering breaks in the planning process. The aim of this problem is to design a distribution network comprising a central depot in which in every period decisions of replenishment must be taken according to inventory levels in each customer, also a set of capacitated vehicles are available at the depot in order to distribute the product to the customers. Finally, when planning the routing process, also we involve the crew scheduling problem by considering the drivers. The objective is to minimize the total costs divided into the inventory holding costs, the routing and the crew scheduling problems. The problem is difficult to solve involving different NP-Hard problems.

2 Problem description

The basic inventory routing problem is defined over a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V} = \{v_1, \dots, v_{n+1}\}$ is the set of $n + 1$ nodes (node 1 represents the supplier in a distribution network with n customers) and $\mathcal{A} = \{(i,j) \mid i,j \in \mathcal{V},\ i \neq j\}$ is the set of arcs that connects the nodes in the set $\mathcal{V}$. The supplier from a central depot must satisfy the demand of a subset of customers $\mathcal{V}_p = \mathcal{V} \setminus \{1\}$, the deterministic demand is defined by a parameter d_i^t and can change in every period $t \in \mathcal{T} = \{1, \dots, T\}$. Drivers, defined by a set $\mathcal{D} = \{1, \dots, \mathrm{D}\}$, must transport a single good throw the use of a set of vehicles $\mathcal{K} = \{1, .., K\}$. The following are the constraints considered in the model:

- Inventory level constraints at the depot and the customers
- Upper and lower inventory level constraints at the customers
- Capacity of vehicles constraints

- Vehicle routing constraints: flow, capacities, time windows and sub-tour elimination
- Shift scheduling constraints: beginning and end of shifts and breaks
- Allocation of drivers in shifts
- Allocation of drivers in vehicles
- Logical constraints of vehicles with shifts and drivers

3 Solution approach and preliminary results

Our method consists in algorithm with three mathematical models stated in Algorithm 1. We solve three models sequentially for each period in the planning horizon. The *inventory_model()* solves a mixed integer model to minimize the total inventory costs subject to constraints related with inventory level constraints at the depot and the customers, the upper and lower inventory level constraints at the customers, and the capacity of vehicles constraints. The *routing_model()* solves a vehicle routing problem with time windows in order to minimize the travel costs and the penalizations of time windows subject to vehicle routing constraints such as flow, capacities, time windows and sub-tour elimination and shift scheduling constraints such as beginning and end of shifts and breaks. The *personnel_model()* solves a mixed integer model to allocate drivers to shifts to minimize the personnel allocation costs subject to allocation of drivers in vehicles constrains and logical constraints of vehicles with shifts and drivers.

Algorithm 1. ILS approach

Define planning horizon $|\mathcal{T}'|$
for t in $\mathcal{T}'$, **do**
  Solve *inventory_model()*→ s_1
  $Isol_{it} \leftarrow I_{it} - SO_{it}\ \forall i \in \mathcal{V};\ t \in \mathcal{T}'$
  $qsol_{ikft} \leftarrow q_{ikft}\ \forall i \in \mathcal{V};\ k \in \mathcal{K};\ f \in \mathcal{F};\ t \in \mathcal{T}'$
  $ysol_{ikft} \leftarrow y_{ikft}\ \forall i \in \mathcal{V};\ k \in \mathcal{K};\ f \in \mathcal{F};\ t \in \mathcal{T}'$
  If s_1 is optimal **or** feasible **then**
    Solve *routing_model()*→ s_2
    $xsol_{ijkft} \leftarrow x_{ijkft}\ \forall\ i \in \mathcal{V}; j \in \mathcal{V};\ k \in \mathcal{K};\ f \in \mathcal{F};\ t \in \mathcal{T}'$
    $esol_{ikft} \leftarrow e_{ikft}\ \forall\ i \in \mathcal{V};\ k \in \mathcal{K};\ f \in \mathcal{F};\ t \in \mathcal{T}'$
    $wsol_{kft} \leftarrow w_{kft}\ \forall\ k \in \mathcal{K};\ f \in \mathcal{F};\ t \in \mathcal{T}'$
  Else: Break
  If s_2 is optimal **or** feasible **then**
    Solve *personnel_model()*→ s_3
    $gsol_{dhorizonte} \leftarrow g_{dhorizonte}\ \forall\ d \in \mathcal{D};\ t \in \mathcal{T}'$
    $Bi \leftarrow Isolit - Xi + dit$
  Else: Break
End for

3.1 Results

We adapt and generate a set of instances based in [2], [7]. We consider a set of customers between 5 and 10, a set of periods of 14, 28, and 56, a set of vehicles of 2,3, 5, and 10, and a set of drivers between 2 to 10.

In Table 1 we present the results of routing and personnel costs with the occupation per vehicles and driver of instances with 5 and 10 customers and 2 to 10 vehicles. It can be seen a relationship between all the resources, and this allows to evaluate the level of management costs, the available occupation, also the time that the driver and the vehicle must occupy of the total of their daily capacity is measured.

Table 1. Results of costs and occupation of instances of 5 and 10 customers

Resources	5 customers						10 customers					
	Routing cost	Avg of Vehicles per period	Avg Occupation of vehicles	Personnel cost	Avg Cost per driver	Avg Occupation of drivers	Routing cost	Avg of Vehicles per period	Avg Occupation of vehicles	Personnel cost	Avg Cost per driver	Avg Occupation of drivers
2 Vehicles												
2 Drivers	23591,384	1,816	79,432	14,478	0,134	4,938	93220,86	2	88,818	0	0	0
3 Drivers	23591,384	1,816	79,432	104,274	1,168	33,434	92957,26	2	89,694	33	0,332	8,224
4 Drivers	23591,384	1,816	79,432	256,836	3,3	81,014	92957,26	2	89,694	281,25	3,33	79,606
5 Drivers	23591,384	1,816	79,432	279,904	3,634	90,772	92957,26	2	89,694	282,542	3,866	84,984
6 Drivers	23591,384	1,816	79,432	288,236	3,834	90,228	93220,86	2	89,622	314,142	4,298	91,146
3 Vehicles												
2 Drivers	65815,46	2,032	56,598	15,642	0,2	4,552	188193,44	2,248	62,972	0	0	0
3 Drivers	65815,46	2,032	56,598	82,334	1,202	30,12	188193,24	2,248	63,342	4,012	0,1	1,322
4 Drivers	65815,46	2,032	56,598	193,502	2,534	60,704	188193,24	2,248	63,342	90,168	1,634	37,648
5 Drivers	65815,46	2,032	56,598	266,43	3,966	81,022	183172,44	2,182	61,852	183,232	3,432	67,54
6 Drivers	65815,46	2,032	56,598	279,988	4,164	82,79	284012,12	1,734	47,274	197,84	4,634	78,55
5 Vehicles												
4 Drivers	120944,9	2,084	34,908	91,472	1,4	34,1	240668,2	1,948	31,576	15,608	0,266	4,404
5 Drivers	120944,9	2,084	34,908	204,75	3,966	68,592	240668,2	1,948	31,576	67,886	1,632	27,098
6 Drivers	120944,9	2,084	34,908	245,666	4,866	80,898	240764,2	1,948	31,8	121,176	3,432	53,622
10 Drivers	120944,9	2,084	34,908	252,52	5,402	83,43	255243,4	1,964	31,614	184,732	5,866	82,502
10 Vehicles												
5 Drivers	110680,656	0,882	7,16	24,884	0,234	4,968	87717,5	0,65	4,98	0	0	0
10 Drivers	110680,656	0,882	7,16	114,418	3,634	47,948	87717,5	0,65	4,98	29,084	1,6	13,134
20 Drivers	110680,656	0,882	7,16	111,402	4,768	40,662	87717,5	0,65	4,98	26,36	2	10,99

References

[1] W. J. Bell *et al.*, "Improving the Distribution of Industrial Gases with an On-Line Computerized Routing and Scheduling Optimizer," *Interfaces*, vol. 13, no. 6, pp. 4–23, Dec. 1983.

[2] L. C. Coelho and G. Laporte, "The exact solution of several classes of inventory-routing problems," *Computers & Operations Research*, vol. 40, no. 2, pp. 558–565, Feb. 2013.

[3] L. C. Coelho, J.-F. Cordeau, and G. Laporte, "Consistency in multi-vehicle inventory-routing," *Transportation Research Part C: Emerging Technologies*, vol. 24, pp. 270–287, Oct. 2012.

[4] L. C. Coelho and G. Laporte, "Optimal joint replenishment, delivery and inventory management policies for perishable products," *Computers & Operations Research*, vol. 47, pp. 42–52, Jul. 2014.

[5] M. Rahimi, A. Baboli, and Y. Rekik, "A bi-objective inventory routing problem by considering customer satisfaction level in context of perishable product," in *2014 IEEE Symposium on Computational Intelligence in Production and Logistics Systems (CIPLS)*, 2014, pp. 91–97.

[6] F. Niakan and M. Rahimi, "A multi-objective healthcare inventory routing problem; a fuzzy possibilistic approach," *Transportation Research Part E: Logistics and Transportation Review*, vol. 80, pp. 74–94, Aug. 2015.

[7] C. Archetti, L. Bertazzi, G. Laporte, and M. G. Speranza, "A Branch-and-Cut Algorithm for a Vendor-Managed Inventory-Routing Problem," *Transportation Science*, vol. 41, no. 3, pp. 382–391, Aug. 2007.

[8] T. Benoist, F. Gardi, A. Jeanjean, and B. Estellon, "Randomized Local Search for Real-Life Inventory Routing," *Transportation Science*, vol. 45, no. 3, pp. 381–398, Aug. 2011.

A Biased Random Key Genetic Algorithm for the Flexible Job Shop Problem with Transportation[1]

Dalila B.M.M. Fontes[1,2], S. Mahdi Homayouni[1], Fernando A.C.C. Fontes[3]

[1] LIAAD - INESC TEC, Porto, Portugal.

[2] Faculdade de Economia, Universidade do Porto, Portugal.

[3] SYSTEC and Faculdade de Engenharia, Universidade do Porto, Portugal.

dfontes@inesctec.pt; smh@inesctec.pt; faf@fe.up.pt

Abstract

This work addresses the Flexible Job Shop Scheduling Problem with Transportation resources (FJSPT), which can be seen as an extension of both the Flexible Job Shop Scheduling Problem and the Job Shop Scheduling Problem with Transportation resources (JSPT). Regarding the former case, the FJSPT additionally considers that the jobs need to be transported to the machines they are processed in; while regarding the latter, in the FJSPT the specific machine processing each operation also needs to be decided. The FJSPT is NP-hard since it extends NP-hard problems. In here, we propose a biased random key genetic algorithm to efficiently find good quality solutions.

1 Introduction

Manufacturing competitiveness is a key concern in many industries and, typically, involves either cost-saving initiatives or productivity increasing ones. Here, we address manufacturing productivity by optimizing the scheduling of the activities involved in production; thus, improving resources utilization.

Due to the complexity involved in job shop scheduling problems, researchers typically limit the scope of the problem by considering some simplifying assumptions. This, in turn gives rise to a plethora of scheduling problems. The most common assumptions include predefined machine-operation assignments and disregarding material handling/transport activities - the job shop scheduling problem (JSP).

However, most manufacturing systems include flexible machines capable of performing several operations and thus, operations can be processed in a subset/all available machines. In such cases, machine-operation assignments are no longer predetermined and need to be found. Incorporating this flexibility into the JSP leads to the Flexible JSP (FJSP), which has been extensively researched [1]. On the other hand, usually, the movement of materials in the shop floor is disregarded, which in addition to being unrealistic, may render the schedules found infeasible. Some authors consider material movements by introducing a time lag. This is still unrealistic as it assumes vehicles to be always available. More recently, some studies have been reported on the JSP with a material handling system, designated as JSPMH, JSPT, or scheduling in flexible manufacturing systems. This problem also requires to determine the sequence of job movements by each vehicle. Most approaches proposed to address the JSPT solve the two subproblems sequentially rather than simultaneously. A solution obtained this way may be sub-optimal or even infeasible. There are, however, some approaches simultaneously scheduling both resources [7]. The JSPT is NP-hard since it combines two NP-hard problems, the JSP and the vehicle scheduling problem. Thus, most of the proposed approaches are of heuristic nature; indeed, only very recently, the first solvable mixed integer linear programming (MILP) model has been reported [3].

This work addresses the flexible JSP with transportation FJSPT combining the above two extensions, which has been recently proposed by Deroussi [2]. Therefore, a solution to the FJSPT includes finding: machine-operation assignments, a schedule for each machine, vehicle-move assignments, and a schedule for each vehicle. As far as we are aware of, only eigth works have been reported on this problem; see [5] and the references therein.

[1]The authors acknowledge the financial support of FEDER/COMPETE2020/POCI/MCTES/FCT through grants PTDC/EEI-AUT/2933/ 2014, POCI-01-0145-FEDER-031821, and PTDC/EEI-AUT/31447/2017.

2 Problem Definition

In the FJSPT one wishes to process n independent jobs $J = \{J_1, J_2, \ldots, J_n\}$ on a set of $M = \{M_1, M_2, \ldots, M_m\}$ machines, moved by a set of vehicles $V = \{V_1, V_2, \ldots, V_r\}$. Each job J_i consists of an ordered set of n_i operations $\{O_{i1}, O_{i2}, \ldots, O_{in_i}\}$, each of which can be performed in a subset of machines $M_{ij} \subseteq M$ with a processing time p_{ij}^m. Each resource (machine and vehicle) can perform (without preemption) one operation at the time and each job can be processed by one resource at the time. All vehicles move at the same speed and can transport jobs between any two locations l_1 and l_2 with travel time $T_{l_1}^{l_2}$. Note that, vehicles may move empty to pick up a job before dropping it off elsewhere. As usual, we assume sufficient machine buffer, where jobs can wait for either the machine or a vehicle. The shop floor layout is known and, in addition to the machines also has a load/unload (LU) area, where initially all jobs are located and are to be dropped off once completed.

Each job is picked up by a vehicle, at the LU area, and moved to the machine assigned to perform its first operation. Once an operation is completed, a vehicle moves the job from its current location to the machine performing its next operation. This process is repeated until the last operation of the job is processed, which can then be moved back to the LU area. The objective is to minimize the makespan.

3 A Biased Random Key Genetic Algorithm

This work proposes a Biased Random Key Genetic Algorithm (BRKGA) that makes use of the framework proposed in [4]. Since, the solution space of the optimization problem is searched indirectly, the framework can be used to solve a wide range of problems [4, 6] and requires only the definition of the chromosome representation and of the decoder. A solution to the FJSPT is encoded as a vector of $N = \sum_{i=1}^{n} n_i$ elements, where elements 1 to n_1 refer to the operations of job 1, elements $n_1 + 1$ to $n_1 + n_2$ to the ones of job 2, and so on. Each vector element is a random key (RK), i.e., a real number in the interval $[0, 1]$. The decoding procedure sorts the vector of RKs in ascending order and converts the indices of the sorted vector into a sequence of repeated jobs, which is then converted into a sequence of operations by replacing (from left to right) the j^{th} appearance of job i by operation O_{ij}, as illustrated in Figure 1 for three jobs each with two operations. Note that, a sequence of operations imposes a sequence of job moves as, to have operation O_{ij} performed at machine, say M_1, job i needs to be moved from the machine processing operation O_{ij-1} (or LU if O_{ij} is the job's first operation) to machine M_1.

A solution also requires machine-operation and vehicle-move assignments, which are determined by a greedy heuristic that starts by choosing the vehicle to perform the move as the one that can pick up the job at its current location the earliest and then the machine to process the next operation of the job as the one that can finish processing it the earliest, for details see Algorithm 1.

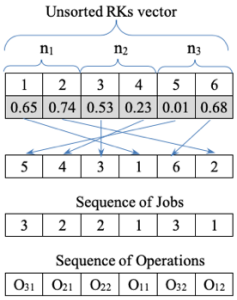


Figure 1: Decoding.

Get O_{ij} the next operation to be assigned to a machine and a vehicle
Get the completion time c_{ij-1} of operation O_{ij-1} and corresponding machine M_{j-1}
Get the current time $f_{(v)}$ and location $l_{(v)}$ of each vehicle
Get the current completion time c_{M_k} of each machine
Select the vehicle v^* that reaches M_{j-1} the earliest and corresponding arrival r_{j-1} and departure d_{ij-1} times

$$v^* = \underset{v \in V}{\operatorname{argmin}} \left\{ f_{(v)} + T_{l_{(v)}}^{M_{j-1}} \right\}$$

$$r_{ij-1} = f_{(v^*)} + T_{l_{(v^*)}}^{M_{j-1}} \text{ and } d_{ij-1} = max\{c_{ij-1}, r_{ij-1}\},$$

Select the machine M^* that can complete the operation the earliest

$$M^* = \underset{M_j \in M_{ij}}{\operatorname{argmin}} \left\{ max\left(d_{ij-1} + T_{M_{j-1}}^{M_j},\ c_{M_j}\right) + p_{ij}^{M_j} \right\}$$

Update the machine and the operation completion time,

$$c_{M^*} = c_{ij} = max\left(d_{ij-1} + T_{M_{j-1}}^{M^*},\ c_{M_j}\right) + p_{ij}^{M^*},$$

Update the time and location of the selected vehicle,

$$f_{(v^*)} = d_{ij-1} + T_{M_{j-1}}^{M^*}, l_{(v^*)} = M^*.$$

Algorithm 1: Assignment heuristic.

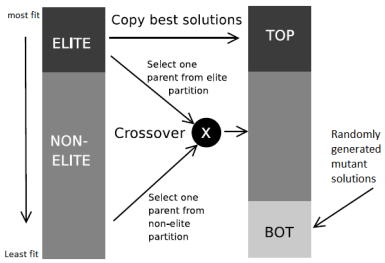


Figure 2: Evolutionary strategy.

A new population is obtained from the current one by copying elite solutions, introducing some random solutions, and generating through parametrized uniform crossover the remaining ones, see Figure 2. Bias is introduced by selecting one parent from the elite solutions and another from the remaining solutions and by giving a larger inheritance probability to the genes of the elite parent.

Table 1 reports the computational performance of the proposed BRKGA and compares it with that of state of art heuristics for the existing ten benchmark problem instances [2]. After providing information on the instances (M - machines, J - Jobs, O - operations), we report the optimal makespan (C^*_{max}) and time required (in seconds) to find it [5], the best makespan (Best) and optimality gap (Gap in percentage) for the three heuristics, and the average BRKGA time requirements (in seconds).

Instances	MILP [5]		GATS [8]		GTSB [9]		BRKGA		
M-J-O	C^*_{max}	Time	Best	Gap	Best	Gap	Best	Gap	Time
8 - 5 - 13	94	2.26	94	0.00	94	0.00	96	2.16	0.32
8 - 5 - 17	144	43.10	150	4.17	146	1.39	144	0.00	0.70
8 - 5 - 19	114	280.64	124	8.77	124	8.77	120	5.26	0.98
8 - 6 - 15	114	17.12	118	3.51	118	3.51	118	3.51	0.50
8 - 6 - 16	120	4.44	124	3.33	124	3.33	120	0.00	0.61
8 - 6 - 18	138	42.09	144	4.35	144	4.35	138	0.00	0.86
8 - 6 - 20	178	131.67	180	1.12	181	1.69	178	0.00	1.18
8 - 6 - 21	174	1396.50	178	2.30	178	2.30	174	0.00	1.34
8 - 7 - 19	134	554.25	144	7.46	146	8.96	138	2.99	0.97
8 - 8 - 19	110†	9852.20	124	12.73	122	10.91	114	3.64	0.93

In all instances, each operation is performed in one of two possible machines.

Table 1: Computational results (†no optimal solution was found within the 50 thousand seconds limit).

As it can be seen, the BRKGA can find very good quality solutions, actually optimal for half of the instances, very quickly. When compared with existing heuristics [8, 9] its best solution is better in eight of the 10 instances solved and only worse for one instance. In addition, the average optimality gap is under 2%, while for the other two heuristics it is almost 5%. Time requirements are always under two seconds; however, solving the MILP takes up to 23 minutes, disregarding the last instance.

References

[1] I.A. Chaudhry and A.A. Khan. A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, 23(3):551–591, 2016.

[2] L. Deroussi and S. Norre. Simultaneous scheduling of machines and vehicles for the flexible job shop problem. In *International Conference on Metaheuristics and Nature Inspired Computing*, pages 1–2. Djerba Island Tunisia, 2010.

[3] D.B.M.M. Fontes and S.M. Homayouni. Joint production and transportation scheduling in flexible manufacturing systems. *Journal of Global Optimization*, pages 1–30, 2017.

[4] J.F. Gonçalves and M.G.C. Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525, 2011.

[5] S.M. Homayouni and D.B.M.M. Fontes. Joint scheduling of production and transport with alternative job routing in flexible manufacturing systems. In *International workshop on Global Optimization*. AIP, 2018.

[6] L.A.C. Roque, D.B.M.M. Fontes, and F.A.C.C. Fontes. A biased random key genetic algorithm approach for unit commitment problem. In *Lecture Notes in Computer Science*, volume 6630, pages 327–339. Springer, 2011.

[7] C. Xie and T.T. Allen. Simulation and experimental design methods for job shop scheduling with material handling: a survey. *The International Journal of Advanced Manufacturing Technology*, 80(1-4):233–243, 2015.

[8] Q. Zhang, H. Manier, and M-A. Manier. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Computers & Operations Research*, 39(7):1713–1723, 2012.

[9] Q. Zhang, H. Manier, and M-A. Manier. Metaheuristics for job shop scheduling with transportation. *Metaheuristics for Production Scheduling*, pages 465–493, 2013.

Memory and feasibility indicators in GRASP for Multi-Skill Project Scheduling with Partial Preemption

Oliver Polo-Mejía[1,2], Christian Artigues[2], Pierre Lopez[2], Lars Mönch[3]

[1] CEA, DEN, DEC, SETC
St Paul lez Durance, France
oliver.polomejia@cea.fr

[2] LAAS-CNRS,Université de Toulouse, CNRS
Toulouse, France

[3] Department of Mathematics and Computer Science, University of Hagen
Hagen, Germany

Abstract

This paper describes a GRASP algorithm aiming to solve a new scheduling problem known as the Multi-Skill Project Scheduling Problem with Partial Preemption, in which not all resources are released during preemption periods. We use a self-adaptive strategy for fixing the cardinality of the restricted candidate list in the greedy phase of the GRASP. We also propose an adaptive evaluation function that includes memory-based intensification, exploiting the characteristics of the best solutions, and a feasibility element for increasing the number of feasible solutions visited. Numerical experiments show the interest of the proposed approach.

1 Introduction

We propose a variant of the Multi-Skill Project Scheduling Problem (MSPSP), aiming to represent the real scheduling process of activities within a nuclear research laboratory: the MSPSP with Partial Preemption (MSPSP-PP). The MSPSP is a generalisation of the well-known Resource-Constrained Project Scheduling Problem, where resources are characterised by the skills they master, and non-preemptive tasks require a certain amount of resources with a specific skill. Determining a solution consists in computing the periods in which each activity is executed and also which resources are assigned to the activity at each period, while satisfying activity and resource constraints: a resource can execute only those skills it masters and must cover only one skill per activity.

The main characteristic of the variant we consider, the MSPSP-PP, is that we allow activities preemption, but we handle the resource release during preemption periods in an innovative way: not all resources can be released, and the possibility of releasing them depends on the characteristics of the activity. We can classify activities within three types: *Non-preemptive*, for which no resource can be released; *partially-preemptive*, for which a subset of resources can be released; and *preemptive*, for which all resources can be released. These activities are scheduled over renewable resources with limited capacity; they can be cumulative mono-skilled resources (compound machines) or disjunctive multi-skilled resources (technicians) mastering Nb_j skills. Multi-skilled resources can respond to more than one skill requirement per activity and may partially execute it (except for non-preemptive activities where technicians must perform the whole activity). An activity is defined by its duration (d_i), its precedence relationships, its resource requirements k ($Br_{i,k}$), its skill requirements c ($b_{i,c}$) and the subset of preemptive resources. Also, activities might or not have either a deadline (dl_i) or a release date (r_i). The MSPSP-PP is NP-hard, and the use of exact methods can be prohibitive for large-sized industrial instances. That is why, in the remainder of this paper, we describe a Greedy Randomized Adaptive Search Procedure (GRASP) aiming to get near-optimal solutions in a reasonable amount of time.

2 GRASP for the MSPSP-PP

A GRASP is an iterative multi-start algorithm, in which each iteration consists of two phases: generation and local search. In the generation phase, a feasible solution is generated; then its neighbourhood is

explored by the local search algorithm. The best solution found over all GRASP iterations is kept as the results. In the following of this section, we describe a GRASP algorithm for the MSPSP-PP.

2.1 Generation and local search phase

In each greedy iteration, using a serial generation scheme, we select an activity randomly from the restricted candidate list (RCL), we then identify the earliest periods when this activity can be scheduled (according to its preemption type and resources and technicians availability), and we finally allocate the technicians to the activity using a Minimum-Cost Maximum-Flow (MCMF) problem.

Let $F(A_i)$ be an adaptive evaluation function, which indicates the degree of relevance of planning activity A_i in the current greedy iteration. Let also define M as the number of not yet scheduled activities after the current greedy iteration. The RCL is made up of the $1 + \alpha * M$, $\alpha \in [0, 1]$, activities having the best F values. Each element within the RCL has a probability of being chosen (π_i) defined as follows:

$$\pi_i = \frac{F(A_i)}{\sum_{j \in RCL} F(A_j)} \, . \tag{1}$$

For choosing the value of α, we propose to use the reactive strategy proposed by Praias and Ribeiro [2], where the value of α is randomly selected from a discrete set $\Psi = \{\alpha_1, ..., \alpha_n\}$ of possible α values. The probabilities associated with the choice of each value are all initially uniformly distributed. After a few iterations, they are periodically reevaluated taking into consideration the quality of the obtained solution for each $\alpha_k \in \Psi$.

The allocation problem is formulated as a MCMF problem over a bipartite graph $G_i = (X, F)$, $X = S \cup P$, where S represents the set of skills required by activity A_i and P is the subset of available technicians who master at least one of the skills within S. In this graph, there is an edge between the source vertex and each vertex $S_c \in S$ whose maximum capacity is equal to $b_{i,c}$ (need of the skill c for executing the activity A_i). There is also an edge between a vertex S_c and a vertex $P_j \in P$, iff the technician P_j masters the skill S_c. The maximum capacity of this arc is fixed to 1 since a technician can only respond to one unit of need per skill. Similarly, there is an edge between each vertex P_j and the sink of the graph, with a maximum capacity equal to the number of skills mastered by the technician P_j (Nb_j). We associate a cost ($CT_{i,j}$) that is related to the criticality of the technician P_j for not yet scheduled activities. To determine the technicians to allocate to the activity, we solve the MCMF problem, and we look at the vertices $P_j \in P$ through which the flow passes.

2.2 Local Search

To explore the neighbourhood, we use an incomplete binary search tree partially inspired on the "Limited Discrepancy Search". For each sequence used in the generation phase, there is a big amount of possible schedules that are defined by the technician allocations we made. The objective of the algorithm is to visit some of these possible schedules. For developing the search tree, we use the sequence obtained in the generation phase within a serial greedy algorithm (similar to the one used in the generation phase, but forced always to follow a given sequence). Every time we must effectuate a technician allocation, we generate a node having in the left-hand branch the best allocation we get solving the MCMF, while in the right-hand branch we have the second best allocation (if such a solution exists). Visiting the whole binary tree can be still prohibitive for industrial instances. From the way schedules are generated, we can expect that heuristics chance of making poor decisions decreases as we add more activities to the partial schedule (going deep in the search tree). We exploit this characteristic by giving to each node a probability, that decreases with its depth in the tree, to examine the right branch; reducing the number of explored branches. Moreover, we limit the maximal number of "discrepancies" (number of times we choose the second best solution for the MCMF) of the branch we visit.

2.3 Adaptive greedy evaluation function

The proposed adaptive greedy evaluation function has three components: priority rule ($L(A_i)$), intensification ($I(A_i)$) and feasibility ($G(A_i)$); and it is defined as follows:

$$F(A_i) = \beta * L(A_i) + \delta * I(A_i) + \gamma * G(A_i) \ . \tag{2}$$

Priority due to priority rule ($L(A_i)$): Computational experiments, presented in [1], suggest that using priority rules "Most Successors" and "Greatest Rank" provide smaller optimality gaps. Let Sc_i be the set of successors of activity A_i; we can define the priority function as follows:

$$L(A_i) = d_i + \sum_{j \in Sc_i} d_j \ . \tag{3}$$

Intensification component ($I(A_i)$): The idea is to use the characteristics of a set ε of q elite solutions to influence the construction phase. In our algorithm, the quality of the solution is highly dependent on the order (Seq_k) in which activities have been treated to obtain solution k. Let define $Bef(k, i, l)$ as a binary function taking the value of 1 is activity i was treated before activity l in the Seq_k, 0 otherwise. Let L be the set of not yet scheduled activities. The intensification component is defined as follows:

$$I(A_i) = \sum_{k \in \varepsilon} \sum_{l \in L} Bef(k, i, l) \ . \tag{4}$$

Feasibility factor ($G(A_i)$): Time windows makes it difficult to find feasible solutions with the greedy algorithm. We introduce a component giving priority to activities with a short slack time. Slack time ($Slack_i$) refers to the margin that an activity A_i has in its planning window. It is a function of the deadline (dl_i), the earliest start time ($r_{\underset{\sim}{i}}$), and the activity duration. The feasibility factor is defined as follows:

$$G(A_i) = \frac{1}{dl_i - r_{\underset{\sim}{i}} - d_i} \ . \tag{5}$$

Note that $L(A_i)$, $I(A_i)$ and $G(A_i)$ must be normalised before taking the weighted sum. Moreover, we have β, δ, $\gamma \in [0, 1]$ and $\beta + \delta + \gamma = 1$. Parameters δ and γ are self-adaptive, and their values are periodically updated. If after K iterations the number of infeasible solutions increases, we must increase the value of γ; on the contrary, if this number decreases, we decrease γ. On the other hand, the parameter δ decreases when the diversity of obtained solutions is too low and increases when the variability is high.

3 Results and conclusions

The GRASP algorithm was tested over a set of 200 instances. The instances have an average makespan of 80 time units, 30 activities with a duration between 5 to 15 time units, up to 5 skills per activity, 8 cumulative resources, 8 technicians (multi-skilled resources). 20% of the activities have release date and deadline (all other characteristics are random). The algorithm allows to obtaining an average gap to optimality of 2.12% within an average time of 35.5 seconds with the size of the elite solution set equal to 20, and the maximal number of GRASP iterations with feasible solutions equal to 550. To analyse the impact of the intensification component, we tested a version of the algorithm without this factor with the same stopping criteria. After a statistical test, we cannot prove that the intensification component significantly improved the average optimality gap (2.16% for the version without intensification). However, the time required by the algorithm is lower when the intensification component is used (44.87 sec if not used). This is because the intensification factors also help the algorithm to generate less unfeasible solutions.

References

[1] Oliver Polo-Mejía, Christian Artigues, and Pierre Lopez. A heuristic method for the multi-skill project scheduling problem with partial preemption. In *8th International Conference on Operations Research and Enterprise Systems*, pages 111–120, Prague, Czech Republic, February 2019.

[2] Marcelo Prais and Celso C. Ribeiro. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12(3):164–176, 2000.

The Flexible Job Shop Scheduling Problem under Non-Fixed Availability Constraints: a Late-Acceptance Hill Climbing Approach

Adrien Wartelle, Taha Arbaoui

Logistics and Industrial Systems Optimization team
Charles Delaunay Institute, FRE CNRS 2019
University of Technology of Troyes
12 Rue Marie Curie, 10004 Troyes, Cedex, France
adrien.wartelle@utt.fr, taha.arbaoui@utt.fr

Abstract

With the surge of smart manufacturing and automation, production systems require sophisticated methods and approaches to solve complex scheduling problems. Within this context, we address the flexible job shop scheduling problem with non-fixed unavailability constraints. This problem is an extension of the classical flexible job shop scheduling in which machines have a non-fixed unavailability period that should be scheduled within a time window. This may correspond to a preventive maintenance activity or a technological upgrade that should be scheduled within a determined period. To solve the problem, we propose a fast and efficient late-acceptance hill climbing approach based on an effective encoding and two neighborhood operators. The approach is then tested on a benchmark from the literature and we show that better results are attained.

1 Introduction

Flexible production systems are taking a bigger importance and gaining more attention due their practical benefits for industrial applications in the era of smart manufacturing. The Flexible Job Shop Scheduling Problem (FJSSP) is among the most studied scheduling problems for flexible production systems. FJSSP is an extension of the well-known Job Shop Scheduling Problem (JSSP), which is NP-Hard. It has been introduced by Brucker and Schlie [1] and was later addressed by numerous works. Different constraints and objectives have been considered for this strategic problem. Energy consumption [4], limited resources [6], among others, have all been considered in FJSSP with objectives such as the makespan, the total completion time, the maximum machine workload or the total machine workload.

In FJSSP, the aim is to schedule a set of n jobs on m machines. Each job i comprises a number of operations n_i, denoted $O_{i,1}, \ldots, O_{i,n_i}$. An operation k of a job i can be performed on a set of given machines, noted M_{ik}. To consider the job i as completed, all its operations must be executed on the set of allowed machines M_{ik}.

In this study, we consider the Flexible Job Shop Scheduling Problem with Non-Fixed Availability periods for the machines (FJSSP-nfa). This problem is an extension of the classical FJSSP whereby machines have some non-fixed unavailability periods, denoted PM_{jl} (period l of machine j). The machines are considered unavailable due to maintenance periods or technological upgrade and this unavailability period has to be scheduled during the scheduling process within a time window, i.e. the unavailability period is not fixed beforehand. Such a problem can be found today in production systems with preventive maintenance activities that are not fixed beforehand but have to be within a time window. We consider three objectives to be minimized: the makespan, the maximum machine workload and the total workload of machines.

Compared to FJSSP, fewer works have been interested to the FJSSP-nfa. This problem has been introduced by Gao et al. [3]. In their work, the authors considered the three objective functions to be minimized. They proposed a hybrid genetic algorithm and two heuristics based on the critical path for the problem. Since the FJSSP is a special case of FJSSP-nfa in which no unavailability period is considered, they modified existing FJSSP instances by considering one unavailability period for each machine. Rajkumar et al. [7] solved the same problem using a GRASP metaheuristic and showed

that their approach outperforms that of Gao et al. [3] for some instances. Li et al. [5] proposed a Discrete Artificial Bee Colony (DABC) approach and showed that their approach is able to find other non-dominated solutions for the problem.

2 Solution approach

In our work, we solve the FJSSP-nfa using the Late-Acceptance Hill Climbing approach. This metaheuristic was introduced by Burke and Bykov [2]. They showed that this metaheuristic overcomes hinges of well-known metaheuristics such as parameter tuning. With a unique parameter and adapted neighborhood operators, this approach can be effective in solving hard optimization problems. To solve the problem at hand, we use a modified scheme of the one used in [2]. To the best of our knowledge, this approach has never been applied to any variant of FJSSP.

A solution is encoded using a list of machine-operation assignments. To obtain a solution from such a list, the machine-operation assignments are taken in row and scheduled. The unavailability periods are then inserted into the schedule by shifting operations which are scheduled during unavailability periods. Such an encoding allows us to obtain a feasible solution without a need of reparation procedures.

The neighborhood operators consist of a swipe operator and a shift operator. The swipe operator swipes two machine-operation from the list in an attempt to shuffle the schedule. The shift operator changes the order of machine-operation assignments in the list by shifting one machine-operation assignment to another random place in the list. The choice of machine-operation assignments in both operators is randomly performed and the new solution is accepted according to the late-acceptance hill climbing criterion (see Algorithm 1). These operators showed good performance in mutating solutions and moving across neighborhoods.

Algorithm 1 displays the scheme of the approach. After the initialization of the parameter LFA and the creation of a random solution, the LAHC list of size LFA is initialized with the cost of the random solution. At each iteration i, the approach generates a candidate solution s' using the neighborhood operators. If the candidate solution has a better fitness than $L[p]$, then $L[p] = C(s')$. The approach runs for a number of iterations $ITER\_MAX$.

Algorithm 1: LAHC algorithm.

```
Initialize parameter LFA
Produce randomly an initial solution s_0 with a cost C_0
Create and initialize a current solution s = s_0 and a current cost C = C_0
Create a list L of size LFA, with L[z] = C_0, ∀z ∈ {0, ..., LFA − 1}
i = 0, p = 0
while i < ITER_MAX do
    Generate a candidate solution s' using neighbordhood operators on s
    Compute the cost of solution s' : C(s')
    p = i mod LFA
    if (C(s') ≤ L[p] or C(s') ≤ C) then
        Accept new solution and store its cost: s = s', C = C(s')
        L[p] = C(s')
    end
    i = i + 1
end
Return the best solution in L
```

Table 1 presents a comparison between existing approaches and the proposed LAHC. Due to space limitation, the largest instance 10×15 (10 machines, 15 jobs and 56 operations) is chosen to compare the four approaches: hGA [3], GRASP [7], DABC [5] and the proposed LAHC. As in [7], two linear

	hGA[3]	GRASP [7]	DABC [5]	LAHC
W_t	109	107	107	98
W_{max}	12	13	12	12
C_{max}	12	16	12	13
$F(0.5-0.3-0.2)$	60.5	60.3	57.7	55.2
$F(0.5-0.2-0.3)$	60.5	60.9	57.7	55.3

Table 1: A comparison between existing approaches and the proposed LAHC on a 15×10 problem.

combinations of the three objectives are chosen as objective functions. As it can been seen from Table 1, LAHC reaches a solution that dominates the solution of the GRASP approach. Moreover, when a linear combination of the three objectives is used, our approach clearly reaches the best solution.

3 Conclusion

This study tackles the flexible job shop scheduling problem with machine unavailability. The considered problem is an extension of the classical flexible job shop scheduling problem where machines have a non-fixed unavailability period that has to be scheduled within a specified time window. Three objective functions were to be minimized: the makespan, the maximum machine workload and the total machine workload.

An effective late-acceptance hill climbing metaheuristic was proposed. This approach has never been applied to flexible job shop scheduling problems and proved to be efficient in solving the problem. Indeed, a new better solution for the 10×15 problem was reached in a short time.

In future works, we aim to apply the approach on the classical flexible job shop scheduling problem. To that end, the proposed approach will be adapted, in particular the neighborhood operators.

References

[1] Peter Brucker and Rainer Schlie. Job-shop scheduling with multi-purpose machines. *Computing*, 45(4):369–375, 1990.

[2] Edmund K Burke and Yuri Bykov. The late acceptance hill-climbing heuristic. *European Journal of Operational Research*, 258(1):70–78, 2017.

[3] Jie Gao, Mitsuo Gen, and Linyan Sun. Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. *Journal of Intelligent Manufacturing*, 17(4):493–507, 2006.

[4] Zengqiang Jiang and Zuo Le. Study on multi-objective flexible job-shop scheduling problem considering energy consumption. *Journal of Industrial Engineering and Management*, 7(3):589–604, 2014.

[5] Jun-Qing Li, Quan-Ke Pan, and M Fatih Tasgetiren. A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. *Applied Mathematical Modelling*, 38(3):1111–1132, 2014.

[6] M Rajkumar, P Asokan, N Anilkumar, and T Page. A GRASP algorithm for flexible job-shop scheduling problem with limited resource constraints. *International Journal of Production Research*, 49(8):2409–2423, 2011.

[7] M Rajkumar, P Asokan, and V Vamsikrishna. A GRASP algorithm for flexible job-shop scheduling with maintenance constraints. *International Journal of Production Research*, 48(22):6821–6836, 2010.

A Multi-Objective Variable Neighbourhood Search for the Beam Angle Selection Problem in Radiation Therapy

Guillermo Cabrera-Guerrero[1], Maicholl Gutierrez[1], Gustavo Gatica[2], José Miguel Rubio[3]

[1] Pontificia Universidad Catlica de Valparaiso, Valparaiso, Chile
guillermo.cabrera@pucv.cl;maicholl.g@gmail.com

[2] Universidad Andres Bello, Santiago, Chile ggatica@unab.cl

[3] Universidad INACAP, Santiago, Chile jrubiol@inacap.cl

Abstract

In this paper, a novel multi-objective variable neighbourhood search algorithm (MO-VNS) is applied to a problem arising in radiation therapy called beam angle optimization. The MO-VNS implements two different neighbourhood moves: the first one is focused on the exploration of the search space while the second one is more focused on the exploitation of the search space. Further, the algorithm implements some problem-specific rules to guide its search. The algorithm is applied to a prostate case and compared to a Pareto local search algorithm previously proposed in the literature. Results show that our MO-VNS algorithm is quite competitive in terms of the quality of the obtained treatment plans and it is able to converge to a set of locally efficient plans within clinically acceptable times.

1 Introduction

Radiation therapy (RT) is, along with chemotherapy, the most common technique to treat cancer. The main goal in RT is to eliminate cancerous cells from the tumour without damaging surrounding normal tissues and organs at risk (OAR). Since OARs are usually quite close to the tumour region, delivering no radiation to the OARs is, in general, simply not possible and, thus, we usually have to be content with minimising the amount of radiation that is delivered to the OARs. This occurs because there is a trade-off between tumour radiation and OARs sparing. One technique that has been shown to be effective in this purpose is the intensity modulated radiation therapy (IMRT). One distinctive feature of IMRT is that it allows treatment planners to modulate the radiation that is delivered to the patient, leading to more uniform tretament plans.

The IMRT problem is a challenging optimisation problem where we need, first, to select the beam angles radiation will be delivered from. After the beam angle configuration (BAC) has been selected, the best possible treatment plan that can be delivered using such a BAC must be computed. This problem is called the fluence map optimisation problem (FMO). Finally, the optimal treatment plan computed during the FMO problem must be delivered using a sequence of apertures that needs to be computed. This problem is called the multi-leaf collimator (MLC) sequencing. In this study we focus on the beam angle selection problem (BAO) which, in turn, needs to solve the FMO problem for each BAC that is evaluated. We do not consider the MLC sequencing problem in this study.

The BAO problem has received an increasing attention during the last 15 years. The vast majority of the approaches proposed so far use a heuristic algorithm to explore the BAC space while using linear or nonlinear solvers to solve the associated FMO problem. Then, the BAC that produces the best treatment plan is selected as the best BAC found by the heuristic. Although appealing, all these single-objective approaches need a set of importance weights to be defined by treatment planners before the optimization step. This is, we need to define *a priori*, how important is to reach the prescribed dose to the tumour w.r.t. protecting a surrounding OAR. Obviously, to define the weights is a very difficult task, especially because each treatment planner (and medical institution) has their own protocols and reference values. More recently, some strategies have been proposed to solve the BAO problem from a multi-objective point of view. In [1], authors propose an adaptive Pareto Local Search to solve the MO-BAO problem. The proposed PLS is able to modify its importance weights at each iteration in order to produce a more

diverse and well-distributed set of efficient treatment plans. However, their approach takes too long before converging to a set of locally efficient treatment plans. This provokes that their approach cannot be applied in clinical practice.

In this paper, we propose a multi-objective variable neighbourhood search algorithm (MO-VNS) to solve the MO-BAO problem. We aim to find solutions that are as good as the ones obtained by the PLS algorithm within clinically acceptable times. For this reason, a set of problem-specific rules have been implemented in order to reduce the search space of the VNS algorithm. More details on the MO-VNS algorithm are given in Section 3

2 MO-BAO Problem: Mathematical Formulation

The MO-BAO problem we are investigating in this paper is

$$\min_{\mathscr{A}\in\mathcal{P}^N(K)} \text{MO-FMO}(\mathscr{A}) \tag{1}$$

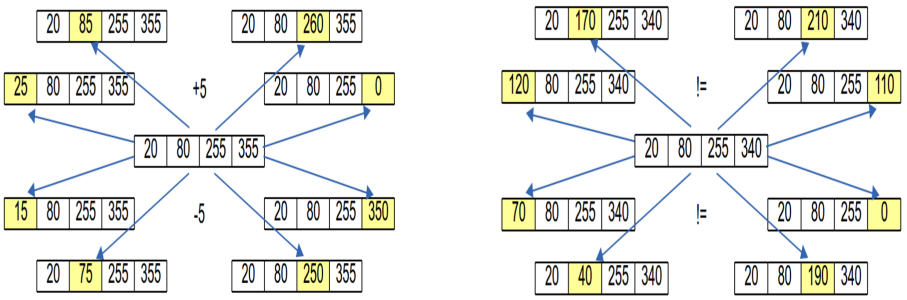
where $\mathcal{P}^N(K)$ is the set of all N-element subsets of $K = \{k\pi/36 : k = 0, 1, 2, \ldots, 72\}$, with $N > 0$ the number of angles in a BAC. Here the associated MO-FMO problem is

$$\text{MO-FMO}(\mathscr{A}) : \min_{x\in\mathcal{X}(\mathscr{A})} z(x) = \begin{pmatrix} z_1(x) \\ z_2(x) \\ \vdots \\ z_p(x) \end{pmatrix} \tag{2}$$

with $z : \mathbb{R}^n \to \mathbb{R}^p$, where $z(x)$ is a vector of p conflicting objective functions and the fluence map x. The set $\mathcal{X}(\mathscr{A})$ is the set of all feasible solutions of the MO-FMO problem when BAC $\mathscr{A}$ is considered. Only beamlets x_i that belong to a beam angle in $\mathscr{A}$ are allowed to have a value greater than zero. The solution of the MO-BAO problem is the set $\mathcal{A}_E$ containing all efficient BACs which use exactly N angles.

3 MO-VNS

In this study, a MO matheuristic algorithm based on VNS is proposed to solve the MO-BAO problem. While the VNS algorithm seeks for high quality BACs, a nonlinear solver, called IpOpt [2], is used to solve the corresponding MO-FMO problem. The VNS prposed here uses two different neighbourhood definitions. The first one consists on moving ± 5 each beam angle within the configuration. Thus, each BAC has $N \times 2$ neighbours (see Figure 1a). The second movement consists on replacing one of the beam angles in a BAC by a randomly chosen one (see Figure 1b). While the first movement is quite effective in exploiting the search space, the second one helps us to effectively explore the search space.



(a) (b)

We apply our MO-VNS algorithm to a prostate case where two OARs are considered, namely the bladder and the rectum.

4 Preliminary Results and Future Work

In this study we have presented a novel MO-VNS problem applied to the MO-BAO problem in radiation therapy. Preliminary results show that our approach can produce a large set of treatment plans within clinically acceptable times. Further, implemented ad-hoc rules allows the MO-VNS algorithm to better explore the search space, avoiding BACs that does not contribute to the quality of the set of efficient treatment plans. As future work, we aim to include the multi-leaf collimator sequencing problem within the proposed framework so we can solve the entire MO-IMRT problem using a one-step approach. Moreover, other problem-specific rules might also be implemented to make the search more efficient in the BAC space.

References

[1] G. Cabrera-Guerrero, A. Mason, A. Raith, and M. Ehrgott. Pareto local search algorithms for the multi-objective beam angle optimisation problem. *Journal of Heuristics*, 24(2):205–238, 2018.

[2] A. Wächter and L.T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

A Metaheuristic Approach for Correlated Random Vector Generation

Mauricio Hurtado[1], Oscar Guaje[1], Andrés L. Medaglia[1], Jorge A. Sefair[2]

[1] Universidad de los Andes
Centro para la Optimización y Probabilidad Aplicada,
Departamento de Ingeniería Industrial
Carrera 1 Este # 19 A – 40, Bogotá D.C., Colombia
{ed-hurta, oo.guaje10, amedagli}@uniandes.edu.co

[2] Arizona State University
School of Computing, Informatics, and Decision Systems Engineering
jorge.sefair@asu.edu

Abstract

The generation of correlated random variables is relevant in the stochastic simulation of financial and manufacturing systems, among many other applications. Generally accepted techniques to generate correlated multivariate observations rely on the mathematical attributes of the probability density functions of the random variables. In this paper, we propose a new approach based on Iterated Local Search (ILS) that induces a desired correlation structure to multivariate random data independently of the probability density function of the input variables. The proposed methodology is able to improve the quality of the results found by the Iman & Conover method [2]– currently used in commercial simulators such as Crystal Ball – at a low computational cost.

1 Introduction

The generation of correlated random variables is pertinent in the stochastic (Monte Carlo) simulation of financial (e.g., portfolios of actions) and manufacturing (e.g., tandem queues) systems, among other applications. The methods commonly used to generate correlated multivariate observations rely on the mathematical properties of the probability density functions, implying that they are hard to generalize. To generate correlated multivariate observations, commercial simulation software such as Crystal Ball and @Risk use the Iman-Conover (IC) [2][7] method to approximate a desired correlation structure by inducing on a (Spearman) rank correlation. Moreover, IC method independently generates random variables, instead of generating them directly from the multivariate distribution [1][8].

2 Problem Statement

Consider that n samples for k random ~~independent~~ vectors $\mathbf{x}_j, j = 1, \dots, k$, are available, which we represent using matrix

$$\mathbf{X} = \begin{bmatrix} x_1(1) & x_2(1) & & x_k(1) \\ x_1(2) & x_2(2) & \cdots & x_k(2) \\ \vdots & \vdots & & \vdots \\ x_1(n) & x_2(n) & & x_k(n) \end{bmatrix}, \quad (1)$$

where $x_l(j)$ corresponds to the j-th observation of random vector l. Let $\bar{\rho}_{ij}$ be the target correlation to be induced between the samples $\mathbf{x}_i$ and $\mathbf{x}_j$ with $i < j$. Because correlations are symmetric, these target correlations are organized in a triangular matrix $\bar{R} \equiv [\bar{\rho}_{ij}]$, with $i < j$. Given that each one of the n samples is drawn from a known (not necessarily identical) marginal distribution $f_{X_j}(x)$, the problem is to reorder the samples for each variable so that we obtain a good approximation of the desired correlation structure $\bar{R}$. Note that reordering the sample $\mathbf{x}_j$ does not affect the form of the marginal distribution of the variable X_j.

3 Inducing correlations with Iterated Local Search

The method of Iman-Conover [2] induces Spearman rank correlations to define the dependencies among the random variables and targeting as a proxy, the (linear) Pearson correlations. However, we can directly induce directly (linear) Pearson correlations (and other correlation structures) by means of mixed-integer programming. Existing exact methods and decomposition strategies are able to correlate multiple random vectors in small- to moderate-size instances (both in number of variables and samples), while large instances still remain a computational challenge [3][4][9]. To overcome this challenge, we propose an Iterative Local Search approach.

We define the search space S as any permutation of the components of the input vectors $\mathbf{x}_j$. Because for each vector there are $n!$ possible sample permutations, the size of the search space is $(n!)^k$. To see this, observe that a possible solution consists of one permutation from each vector, which leads to $n! \times n! \times \ldots n! = (n!)^k$ possible solutions (this is the same as the total number of X-matrices from (1)). The initial solution $X_0 \in S$ is a matrix with the original order obtained by generating every variable independently (see (1)). Let $\rho(X)$ be the triangular matrix with the sample correlations given by solution $X \in S$. The quality of such solution is given by the absolute difference between $\bar{R}$ and $\rho(X)$, only for entries (i, j) such that $i < j$ as the correlation is symmetric. We define the evaluation function of the local search, $\Delta(X)$, as:

$$\Delta(X) = \sum_{i=1}^{k} \sum_{j=i+1}^{k} |\rho(X)_{ij} - \bar{\rho}_{ij}| \tag{2}$$

A solution matrix X' is a neighbor of matrix X if and only if X differs from X' in at most two components. This means that the neighborhood of X, denoted by $N(X)$, is given by all possible swaps of two samples, which only occur within a vector (note that swaps of samples between vectors are not allowed as this will change the marginal distributions). Because the number of swaps within a vector is given by $\frac{n(n-1)}{2}$, then neighborhood of X contains $k\left(\frac{n(n-1)}{2}\right)$ elements.

Our proposed ILS method improves an initial solution X_0 until it obtains a near optimal solution X^*. At an iteration t and given a solution X_t, our method explores $N(X_t)$ aiming to improve (2) to get a new solutionX_{t+1}. If X_{t+1} improves on the current solution (i.e., $\Delta(X_{t+1}) < \Delta(X_t)$), ILS proceeds to iteration $t + 2$. Otherwise, ILS rejects the solution, keeps X_t, and performs a perturbation operation. The process is repeated until the algorithm explores a given number of solutions T_{max}.

4 Preliminary Results

We conducted two sets of preliminary experiments. The first experiment, starts from an independently generated solution and tries to find a solution close to the one obtained by the method of Iman-Conover [2]. The second experiment starts from the output of Iman-Conover [2] and tries to improve the induced correlations. Our preliminary results show that our method is able to achieve solutions with comparable quality to that of Iman-Conover; and then it improves the solution by at least 42% with a low computational burden.

Figure 1 shows the performance of our ILS in an instance with 5 variables and 500 samples, when the initial solution is a set of randomly generated vectors with no correlation induced. In 1257 iterations (5.2 seconds) the ILS achieves a solution as good as Iman-Conover. Moreover, in 3000 iterations (74.2 seconds), we obtain a value of $\Delta(X) = 0.062$, which reflects an improvement of 88% compared with the Iman-Conover solution.

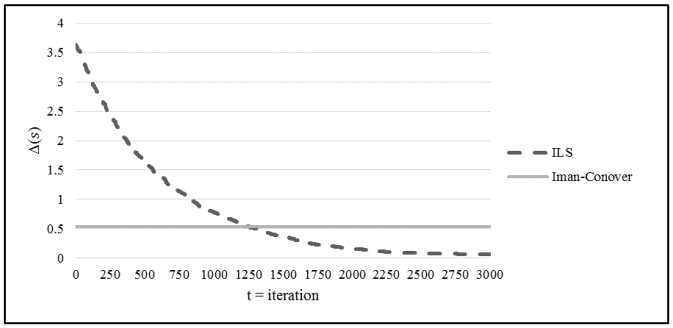


Figure 1: Gap of the algorithms

Figure 2 shows the performance of our ILS method when the result of the Iman-Conover is used as an initial solution. After 2000 iterations (231.3 seconds) we achieve a value of $\Delta(X) = 0.030$, which is an improvement of 93% compared with the original result of Iman-Conover.

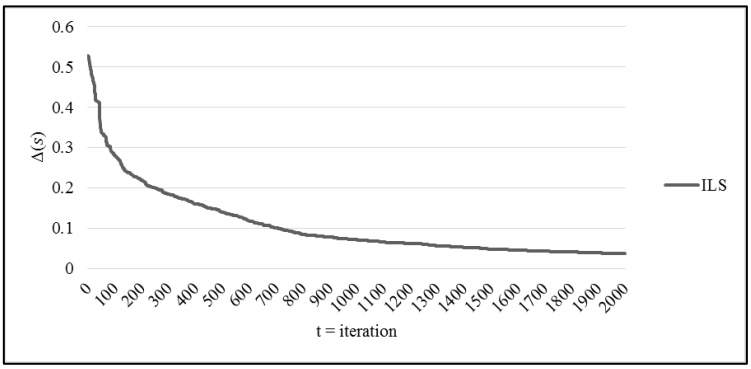


Figure 2: Convergence of ILS starting at Iman-Conover instance

References

[1] C.N. Haas. On modeling correlated random variables in risk assessment. *Risk Analysis*, 19(6):1205–1214, 1999.

[2] R.L. Iman and W.J. Conover. A distribution-free approach to inducing rank correlation among input variables. *Communications in Statistics - Simulation and Computation*, 11(3):311–334, 1982.

[3] O. Guaje, A.L. Medaglia, and J.A. Sefair. A decomposition approach for correlated random vector generation. *Column Generation 2016*. Búzios, Brazil, May 22-25, 2016.

[4] O. Guaje, J.A. Sefair, and A.L. Medaglia. Mixed-integer programming formulations for correlated random vector generation. Centro para la Optimización y Probabilidad Aplicada. Working paper, 2019.

[5] H.R. Lourenco, O.C. Martin, and T. Stützle. Iterated local search: framework and applications. In Michel Gendreau and Jean-Yves Potvin, editors, Handbook of Metaheuristics, pages 320–353. Springer, 2010.

[6] T. Schettini, F. Malluceli, and H. Ramalhinho. Scheduling of a line manager using Iterated Local Search. In Proceedings of the MIC/MAEB 2107 (MIC2017). Barcelona, Spain. July 4-7, pp: 76-78, 2017.

[7] S. Mildenhall. Correlation and aggregate loss distributions with an emphasis on the Iman-Conover method. Technical Report, Casualty Actuarial Society (CAS) Correlation Working Party - CAS Winter Forum, 2005.

[8] P. A. G. Van Der Geest. An algorithm to generate samples of multi-variate distributions with correlated marginals. *Computational Statistics and Data Analysis*, 27(3):271-289, 1998.

[9] A.L. Medaglia and J.A. Sefair. Generating correlated random vectors using mixed-integer programming. In Proceedings of the 2009 Industrial Engineering Research Conference (IERC). Miami, Florida, United States, 2009.

Extended Solution for the Team Selection Problem Using ACO

Lázaro Lugo[1], Marilyn Bello[1,2], Ann Nowe[3], Rafael Bello[1]

[1] Department of Computer Science, Universidad Central "Marta Abreu" de Las Villas, Santa Clara, Cuba
Santa Clara, Cuba
{ljplugo,mbgarcia}@uclv.cu
rbellop@uclv.edu.cu

[2] Faculty of Business Economics, Hasselt University
Hasselt, Belgium
marilyn.bellogarcia@uhasselt.be

[3] Artificial Intelligence Lab, Vrije Universiteit Brussels
Brussels, Belgium
ann.nowe@vub.ac.be

Abstract

Team's selection is an important task in human resources management in which the purpose is to carry out a process of personnel selection to form teams. This process is usually carried out from the *ranking* of candidates who express the preferences of decision-makers. This paper extends the problem of team selection problem in the competitive environment. The extensions include considerations in the team selection process where it is necessary to share candidates and establish selection priorities among decision-makers. Both extensions are problems in the conformation of software teams. To provide a solution to this problem we use Ant Colony Optimization.

1 Introduction

Personnel selection is the process of selecting the most prepared and qualified people to do a job [4]. This problem can also be seen as the problem of forming a team. In this case, the problem is not to select the most suitable employee for a job, but to select a group of people who should work as a team [1].

Most of the research and publications that deal with forming team, analyze the factors to be taken into account for the selection of its members, and how to use decision-making methods to shape them taking into account these factors, so that teamwork is achieved. In many cases the result is to generate a *ranking* of candidates that is used as a base for selection [2].

This problem becomes more complex when more than two decision-makers carry out the selection process in a competitive environment, i.e. when they must form their teams by selecting personnel from the same set of candidates. In this framework, a conflict of interest arises that affects the resulting selection [1, 5].

The research presented in [1] propose a method to form teams in a competitive environment based on Ant Colony Optimization (ACO) metaheuristics [3], and in [5] we propose an extension for when there are several decision-makers involved in the selection process. Although the results obtained in these proposals have been satisfactory, the solutions obtained still have some limitations that hinder their application in specific problems, such as the forming of software teams.

In this paper proposes two extensions to the contribution presented in [5] to form teams in a competitive environment. These extensions include the possibility for decision-makers to share candidates and establish selection priorities among decision-makers.

2 Extensions to the Method for the Team Selection in a Competitive Environment Using ACO

This research is based on the $ASMMTS$ algorithm proposed in [5] which uses ACO metaheuristics. Several models of the ACO metaheuristic have been proposed in the literature. In the context of this

work, a hybrid between the Max-Min Ants System (MMAS) [7] and the Multi-type Ants [6, 8] is used.

In the $ASMMTS$ algorithm the selection made by the decision-makers is made randomly. This causes ants to select candidates at random, without allowing an order of priority to be established in the decision-makers at the time of selection. In the process of formation of teams it is common that it is necessary to establish a certain priority for some decision-makers, either for very particular reasons or because there are decision-makers with a higher degree of importance. In order to adapt this new restriction to the $ASMMTS$ algorithm, point 7.1 is added,

7.1 The order of priority is established from the selection of each ant in the set V_{ipk}, this set consists of all the nodes that have not been selected yet by the ant k in the iteration i or by some of the ants of its group according to its type p. If no specific order is established, priority is given at random.

Example 1. For 5 decision-makers, the following priority can be set $\{D_2, D_4, D_5, D_3, D_1\}$. In this way the order in which the decision-makers are selected is established where D_2 has a higher priority than D_4, D_5, D_3 and D_1.

In the $ASMMTS$ algorithm when a candidate is selected by a decision-maker, it can not be used by any other decision-maker. In this way, the algorithm restricts the possibility that a candidate can be selected by two decision-makers at the same time. It happens that, sometimes, when making a selection of teams, it is necessary to be able to share human resources, due to the fact that the skills that a candidate possesses are not possessed by the other candidates that were present in the selection. Specifically in the formation of software development teams is common that people such as designers or specialists in a certain field are shared by more than one team. To adapt this new restriction to the $ASMMTS$ algorithm, point 12.1 is added,

12.1 After having obtained $R*$, where $R*$ is the set of all the solutions found by each type of ant or decision-maker, the candidates to be shared by the decision-makers are assigned taking into account the initial set of R of all preferences on the set of candidates for each decision-maker.

Example 2. For 10 candidates and 5 decision-makers, the candidate C_6 will be shared by decision-makers D_1 and D_2, and candidate C_5 will be shared by decision-makers D_2 and D_3, the algorithm would receive this information: $\{C_6, D_1, D_2, C_5, D_2, D_3\}$.

3 Experimental Study

The aim of this experimental study is to verify the efficacy of the $ASMMTS$ algorithm after the two new extensions (i.e., $ASMMTSvE$ algorithm). This problem has not been approached by any other research, so there is no precedent to establish a comparison with the method we are extending, so our experimentation is based on comparing the results obtained with the extended method and the strategy following the strictly order of the *rankings* of preference of each decision-maker.

A simulation of the team formation process was carried out similar to the one carried out in [**?**]. Tables 1 and 2 show the values of the heuristic evaluation function after executing the algorithm with the new modifications. The first column in Table 1 shows the priority of selection of decision-makers, while Table 2 shows the candidates that can be shared by more than one decision-maker. The second column shows the values of the heuristic evaluation function strictly following the order established in the *rankings*. The third column shows the values of the heuristic evaluation function after applying the $ASMMTSvE$ algorithm.

Priority of Selection	**Order in the Rankings**	**ASMMTSvE**
$\{D_1, D_5, D_3, D_2, D_4\}$	Eval(R*,R)=26.8	**Eval(R*,R)=7.0**
$\{D_4, D_2, D_1, D_3, D_5\}$	Eval(R*,R)=20.4	**Eval(R*,R)=6.2**
$\{D_1, D_5, D_3, D_4, D_2\}$	Eval(R*,R)=24.0	**Eval(R*,R)=5.4**

Table 1: Results of the experimentation establishing an order of priority in the selection.

Shared Candidates	Order in the Rankings	ASMMTSvE
$\{C_3, D_1, D_2\}$	Eval(R*,R)=15.6	**Eval(R*,R)=7.8**
$\{C_0, D_3, D_5; C_4, D_1, D_2\}$	Eval(R*,R)=13.6	**Eval(R*,R)=8.0**
$\{C_2, D_1, D_3; C_1, D_4, D_5, D_2\}$	Eval(R*,R)=31.2	**Eval(R*,R)=12.6**

Table 2: Results of experimentation with shared candidates.

The experimental results show that the $ASMMTSvE$ algorithm has a similar performance to the $ASMMTS$ algorithm proposed in [5]. The heuristic value obtained is lower than heuristic value obtained strictly following the order established in the *rankings*. In other words, the modifications made do not affect the satisfactory results obtained with the original algorithm.

4 Conclusions

The $ASMMTS$ algorithm is extended to allow it to be more applicable to problems of team forming in a competitive environment.

The two new adaptations make it possible to establish selection priorities among decision-makers and to have shared candidates.

From the experimental study, it can be concluded that extensions of the $ASMMTS$ algorithm do not affect its effectiveness.

References

[1] Marilyn Bello, Rafael Bello, Ann Nowé, and María M García-Lorenzo. A method for the team selection problem between two decision-makers using the ant colony optimization. In *Soft Computing Applications for Group Decision-making and Consensus Modeling*, pages 391–410. Springer, 2018.

[2] Stanislav Dadelo, Zenonas Turskis, Edmundas Kazimieras Zavadskas, and Ruta Dadeliene. Multi-criteria assessment and ranking system of sport team formation based on objective-measured values of criteria set. *Expert Systems with Applications*, 41(14):6106–6113, 2014.

[3] Marco Dorigo and Thomas Stützle. *The ant colony optimization metaheuristic: Algorithms, applications, and advances*, pages 250–285. Springer, 2003.

[4] Yaima Filiberto Cabrera, Rafael Bello Pérez, and Ann Nowe. A new method for personnel selection based on ranking aggregation using a reinforcement learning approach. *Computación y Sistemas*, 22(2), 2018.

[5] Lázaro Lugo, Marilyn Bello, Ann Nowe, and Rafael Bello. A solution for the team selection problem using aco. In *International Conference on Swarm Intelligence*, pages 325–332. Springer, 2018.

[6] Ann Nowé, Katja Verbeeck, and Peter Vrancx. Multi-type ant colony: The edge disjoint paths problem. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 202–213. Springer, 2004.

[7] Thomas Stützle and Holger H Hoos. Max–min ant system. *Future generation computer systems*, 16(8):889–914, 2000.

[8] Peter Vrancx, Ann Nowé, and Kris Steenhaut. *Multi-type ACO for light path protection*, pages 207–215. Springer, 2006.

The Sustainable Store Selection and VRP: a Math-Heuristic Approach and User Application

Paulo Bautista, Brian Rozo, Camilo Gomez,

Universidad de los Andes
Carrera 1 Este # 19 A – 40, Bogotá D.C., Colombia
pc.bautista304, ba.rozo10, gomez.ch {@uniandes.edu.co}

Abstract

We introduce the Sustainable Store Selection and Vehicle Routing Problem (S^3-VRP), in which a user specifies a set of products to be purchased, and needs to decide in which store to buy each product, aiming to minimize economic, social, and environmental costs. We provide two solution strategies for the S^3-VRP: a network flows based exact approach, and a metaheuristic approach based on the implementation of a GRASP and a Genetic Algorithm. We test our strategies with instances in Bogotá (Colombia) and deliver a prototype of an interactive user application. The obtained solutions are satisfactory at the prototype level but further specialization is required to achieve a product that satisfies real-time needs.

1 Introduction

Citizens in large urban centers routinely face a tradeoff when purchasing goods, namely: whether to pursue the cheapest prices or the most efficient tour. Although savings in purchases are desirable, choosing the cheapest store for each product may be sub-optimal, as longer trips imply more fuel consumption (along with its costs and associated carbon emissions), as well as more time spent in traffic, which takes a toll on work, leisure, and personal activities.

We introduce the Sustainable Store Selection and Vehicle Routing Problem (S^3-VRP), in which a user specifies a set of products to be purchased, and needs to decide in which store to buy each product in order to satisfy an objective that balances the direct cost of purchases and the costs associated to visiting the stores where the desired prices are available. The purpose of the S^3-VRP is to obtain a *sustainable* purchase plan, in the sense that it must incorporate economic, social, and environmental aspects. The *economic* dimension considers the minimization of the cost of purchases and fuel consumed in the tour. The *social* dimension pursues the minimization of the time spent by users in traffic, which represents a loss for the individual user and for the city (as it implies adding to a city's congestion). Finally, the *environmental* dimension accounts for the carbon emissions associated to the route.

The solution for the S^3-VRP is not trivial, as any purchase choice leads to a different set of visited stores, for which a TSP must be solved to determine costs associated to the route, which in turn may discourage a solution previously deemed as attractive in terms of purchase cost. We address the S^3-VRP through two solution strategies: a network flows approach (Section 2), and a metaheuristic approach involving a GRASP and a Genetic Algorithm (Section 3). We contrast the pros and cons of the proposed approaches and introduce a prototype user application (Section 3).

2 Network flows approach

We propose two strategies inspired by network flows models. First, we construct a network that exhaustively covers the permutations of store sequences and purchase choices, and solve a minimum-cost flow problem on such network, guaranteeing that all products are purchased. This strategy is evidently not competitive in terms of memory or computation time, but provides an exact global optimum that serves as a benchmark for other strategies, such as the metaheuristic approach presented in Section 3. Second, we introduce a twofold network, in which a set of variables determine the route to adopt, while a second set of variables (on the same network) determine the flow of products. This strategy solves the problem to optimality at a relatively competitive computation time (at least at the prototype level).

2.1 Exhaustive network

The exhaustive network consists of: (i) a source node and a sink node with unit supply and demand, respectively; and (ii) a set of transshipment nodes with as many layers as products must be purchased, accounting for possible permutations of stores that may be visited when purchasing products. For illustration, consider the following situation: one intermediate node represents "*buying product A at store X, coming from store Y*", and is connected to an intermediate node at the next layer which represents "*buying product B at store Z, coming from store X*". The arc connecting these nodes includes the cost of buying product B and the cost of traveling from store X to store Z; hence, arcs that connect nodes representing purchases at the same store only include the purchase cost, as no traveling is needed. The problem is solved as a shortest-path problem with a side constraint enforcing that each product is purchased exactly once. Because all permutations are considered, this network can be prohibitively large for realistic instances, but it is useful as a benchmark for middle sized problems.

2.2 Twofold network

The twofold network consists of: (i) a unique source node with a supply equal to the number of products to purchase; (ii) a sink node for each product (with a unit demand), which receives arcs coming from all stores (the cost of "*buying a product at a store*" depends on the arc that delivers the demanded unit); and (iii) a set of transshipment nodes with as many layers as products must be purchased (the costs of visiting different stores depend on the flows on arcs between layers). The "twofold" characteristic is derived from the use of two sets of variables: (i) variables that determine the tour of visited stores; (ii) variables that determine the purchases of products. The problem is solved as a minimum cost flow problem with balance constraints for each set of variables and a side constraint specifying that products can be only purchased at "visited stores". In spite of being exact, this approach is competitive at the prototype level, arguably due to the almost pure network structure of the problem. Promising results were obtained using Gurobi's solver, but these were not achieved when switching to open source solvers.

3 Metaheuristic model

The purpose of this research is to produce a prototype user application to assist urban consumers. Therefore, we need an approach that, unlike exact methods, does not rely on the power of commercial solvers and can provide reasonable computation times for a nearly real-time application. We propose a straightforward metaheuristic approach to solve the S^3-VRP, which uses a GRASP to initialize a population and a Genetic Algorithm to improve the solutions.

3.1 GRASP

The GRASP algorithm was used to create the initial population for the Genetic algorithm. The iterations of the algorithm are the creation of a new individual for the initial population. Each one is divided in two parts: construction and local search. The construction part consists of assigning the stores and the products to the individual. This phase begins with a list of candidates E, which are random stores and the products to be purchased. From each candidate, a restricted candidate list is constructed and one of these is randomly selected. At the end of this phase, the selected candidate has the best fitness. Then, the local search will find the best tour for the individual. The local search consists in making swap movements of the tour of all the candidate stores. Then, if the fitness of the individual has been improved, another swap takes place until the fitness of the individual does not improve any more. The algorithm stops when it has created the n individuals desired in the initial population.

3.2 Genetic Algorithm

With the initial population, the genetic algorithm uses crossover, mutation, elitism, and tournament selection to evolve the individuals. In each iteration, the best individuals are selected based on their fitness. Then, the crossovers that will take place are defined with k-tournament selection. Individuals will mutate with a rate λ. In crossovers, a random vector is created, which will define the chromosomes that the offspring will take from each parent. On the other hand, mutations only affect one chromosome of an individual if a generated random number is less than the mutation rate. Finally, the elitism strategy selects the m individuals that will survive in the next generation, and a new population is obtained via

crossovers and mutation. The algorithm stops when the best individual is repeated significantly in the generations.

4 Conclusions

We introduce the Sustainable Store Selection and Vehicle Routing Problem to assist urban consumers in balancing potential savings in purchases and the indirect costs of reaching the stores that offer attractive prices. We propose variations of network flows problems that provide bounds based on exact solutions of the proposed problem. Then, we develop a pilot metaheuristic approach that combines a GRASP and a Genetic Algorithm. For small and medium instances, the solutions obtained with the proposed metaheuristic achieve optimality but do not significantly improve the computational times of the exact approaches. Several analyses were done for determining the parameters of the GRASP and the Genetic algorithm: the size of the initial population, the rate of mutation, and the number of individuals selected by elitism. However, our implementation seems to have significant room for improvement: our model is coded in Python, whereas a C implementation can be much more competitive; the encoding of our individuals can be more compact and take advantage of more efficient data structures. The choice of Python was motivated by the need to develop a user application (using Dash), but a hybrid Python/C approach seems convenient for future work. The computational times depended on the number of stores and products.

Bi-Objective CVRP solved using a novel metaheuristic ILS Based on Decomposition (Extended Abstract S1)

Luis F. Galindres[1], Ramón A. Gallego[2], Frederico Gadelha Guimarães[3]

[1] Universidad Tecnológica de Pereira
Carrera 27 No. 10-02 Barrio Alamos, Pereira, Colombia
lugal@utp.edu.co

[2] Universidad Tecnológica de Pereira
Carrera 27 No. 10-02 Barrio Alamos, Pereira, Colombia
ragr@utp.edu.co

[3] Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627, Belo Horizonte, Brasil
fredericoguimaraes@ufmg.br

1 Introduction

The problems of vehicle routing (VRP), usually have been studied with a single objective function, which is defined by the distances associated with the routing of vehicles, the problem is to design a set of routes to meet the demands of the customers at minimum cost. However, in real life, it is necessary to take into account other objective functions, such as social functions, which take into account equity considerations such as drivers' workload balance. This has allowed a growth in both the formulation of multi-objective models and in the exact and approximate solution techniques [9].

The CVRP consists in the distribution of goods from a deposit to a set of clients. Each of the clients needs a certain amount of goods, for this, we have a fleet of vehicles and a set of routes associated with a single vehicle must be designed in such a way that the demand of each of the customers is met taking into account the following operational restrictions: i) Do not exceed the capacity of the vehicle, ii) in each proposed solution a customer must be served by a single vehicle, iii) each of the vehicles in the fleet starts at the depot, delivers the respective demands and returns to the depot [3].

In particular, in this paper, the bi-objective Capacitated Vehicle Routing Problem (bi-objective CVRP) is addressed taking into account economic and social objectives. The economic objective is to minimize the cost associated with the CVRP route design, the social objective is to balance the workload of each of the drivers [9]. The formulation of the social objective must satisfy several restrictions established by union contracts and company regulations (for example, work periods during the day, number and duration of breaks during service, maximum duration of driving periods, extra hours) [3], in addition, the allocation must be equitable among the drivers so it must take into account factors such as traffic conditions and the number of customers assigned to each vehicle [5]. Furthermore, in some cases depending on the characteristics of goods and services, the concept of equity is essential [14].

For the study of pareto fronts obtained using the proposed bi-objective model, different metrics will be used that determine both the quality of the solutions and the validation of the metaheuristic.

2 Solution Method

2.1 Exact Method

To solve the bi-objective CVRP it is proposed initially to implement an exact method, which will serve as a reference to carry out a comparison with those obtained using the metaheuristic.

To solve the problem in an exact way, a mixed integer linear model (MILP) was defined in which the economic objective function consists of minimizing the costs associated with the design of the routes. For its part to define the social objective function, we took into account what was defined in [7] where 7 objective functions related to the route balance were studied, finding that the most used are $\max - \min$ (also

called range), which minimizes the difference between the maximum and minimum length of routes; $\min\max$ which minimizes the maximum length path; and var that minimizes the variance of the lengths of the routes. In a recent study [9] analyzed the objective functions MAD (Mean Absolute Deviation), defined as the average absolute difference between the length of each route and the average of the route lengths; and the $Gini$ coefficient is one of the most commonly used measures for inequality studies. Finally, in [2] the quadratic objective function var was used, showing satisfactory results with a high computational cost.

To solve the bi-objective CVRP in a exact way, we used the traditional cost function and the linear objective function $\min\max$, which seeks to minimize the maximum path length of the solution as shown in (1).

$$minimize(F_1, F_2) \tag{1}$$

where

$$F_1 = \sum_{i,j \in V} c_{ij} x_{ij} \tag{2}$$

$$F_2 = \max_{i \in \Omega} \{\Omega_i\} \tag{3}$$

c_{ij} and x_{ij} correspond to the cost and existence of the arc between the nodes i and j. Ω_i corresponds to the length of the i route.

In the optimization process, the decomposition concept will be used, whose objective is the generation of the pareto front. Decomposition is a widely used strategy in multi-objective optimization problems [18] which consists of dividing the multi-objective problem into N scalar optimization subproblems.

This method transforms multiple objectives into an aggregated objective function by multiplying each objective function by a weighting factor and summing up all weighted objective functions. Besides, to find a well-distributed Pareto front, for example, in [13] the weighted Tchebycheff aggregation method is used. In this paper the weighted Tchebycheff method is used too and vectors λ are generated equally spaced using the method implemented in [10]. According to [8] this formulation provides a necessary condition for Pareto optimality.

The formulation that includes the weighted Tchebycheff aggregation is shown in 4:

$$minimize \qquad \Psi \tag{4}$$

subject to:

$$\Psi \geq \lambda_i (F_i - Z_i) \qquad \forall 1 \leq i \leq m \tag{5}$$

The parameters $\lambda_i > 0$ are the weights ($\sum_{i=1} \lambda_i = 1$); Z_i is the utopia point defined as $Z_i = \min_{x \in X} F_i(x) - \delta_i$ for $1 \leq i \leq m$ ($\delta_i > 0$)

To determine Z_i, the vectors $\lambda^1 = [1, 0]$ and $\lambda^2 = [0, 1]$ are generated and the equation (5) is modified by $\Psi \geq \lambda_i F_i \quad \forall 1 \leq i \leq m$ with which problem in (4) is solved, obtaining the minimum values of each objective function.

2.2 Approximated Method

On the other hand, to solve the problem in an approximate way, we propose a novel method based on metaheuristic Iterated Local Search (ILS) and again the concept of decomposition (ILS-D) to generate the front. In this paper, an adaptation of the ILS-RVND technique shown in [15] is made, which consists in that from an initial solution a local search is performed to improve the solution, from this a perturbation mechanism is applied to obtain a new solution. This process is performed in a certain number of iterations. The local search is based on the construction of neighborhoods whose solutions are accepted

according to a criterion. This method has shown to be efficient in this type of problems [16]. Decomposition is used as in the previous case, each subproblem corresponds to the initial solution used in the ILS-RVND.

The N subproblems are optimized using the MOEA/D framework as shown in [18]. In addition, 2 modifications to the previous framework were taken into account to increase its efficiency: i) **MOEA/D-GR** (Global Replacement) is used to make a more adequate update in the neighboring subproblems $(B(i))$. In this scheme, we first find a more appropriate neighborhood to be replaced by the candidate solution x_i. The details are shown in [17]. ii) **MOEA/D-GRA** (Generalized Resource Allocation) is used to dedicate more resources to the subproblems that are more likely to improve. In each iteration computational resources are assigned to some subproblems according to the vector *PoI*. Specifically, the resource allocation strategy *Online* is used to define a utility function based on the relative improvement in the last ΔT iterations. The details are shown in [19].

The details of the proposed **ILS/D** are shown in Algorithm 1.

1: **N** Initial Solutions x_i
2: $g_i \leftarrow 0,\ u_i \leftarrow 0,\ p_i \leftarrow 0.5 \quad \forall\, i = 1, 2, ..., N$
3: $iter \leftarrow 1$
4: **while** $iter \leq maxIter$ **do**
5: **for** $i = 1, 2, ..., N$ **do**
6: **if** $rand() \leq p_i$ **then**
7: $x_i' \leftarrow$ **ILS**$(x_i,\ \lambda_i,\ z_i)$
8: $update\ z_i$
9: Find subproblem of x_i' **MOEA/D-GR**
10: $update\ B(j)$
11: **end if**
12: **end for**
13: updating u_i and p_i **MOEA/D-GRA**
14: $iter \leftarrow iter + 1$
15: **end while**

Algorithm 1: ILS/D Algorithm

Lines (1-3) of the algorithm 1 show the initialization of the algorithm highlighting the initial value of the probability vector $p_i \leftarrow 0.5$. In lines (6-11), the algorithm ILS is executed for each subproblem i, in line (8) the reference point z_i is updated for the current subproblem, in lines (9-10), find the most appropriate neighborhood to make the replacement of x_i'. Finally on line (13) the probability vector p_i is updated taking into account the performance of the subproblem i in the last ΔT iterations.

The list of neighborhoods used by the algorithm ILS, are classified in *inter-route* (table 1) and *intra-route* (table 2).

i	N_i
1	$swap(1,1)$
2	$swap(2,1)$
3	$swap(2,2)$
4	$insertion()$
5	$shift(2,0)$
6	$shift(3,0)$
7	$2\_opt\_parallel()$
8	$2\_opt\_crossed()$

Table 1: List of neighborhoods $Inter\_Route$.

i	N_i
1	$swap(1,1)$
2	$insertion()$
3	$shift(2,0)$
4	$shift(3,0)$
5	$2\_opt()$

Table 2: List of neighborhoods $Intra\_Route$.

3 Results

The instances used in this work correspond to those proposed in [1].

Initially, the instance *A-n37-k5* was run by the exact method using the solver **CPLEX**, the script was implemented using **AMPL**. Due to the high computational effort in larger instances, the results obtained in this instance were used to be compared with those obtained with the approximate method. This instance is also analyzed in [11] where they solve the bi-objective CVRP problem using a hybrid technique based on the ϵ-constraint method and the Multi-objective NSGA-II algorithm. The obtained optimal front is shown in the figure 3 marked with red color.

The proposed algorithm **ILS/D** was coded in C++ and instances of low, medium and high mathematical complexity were run in the range of 37-79 clients, taken from the benchmark presented by [1].

To validate the proposed algorithm, we obtained 20 fronts in independent runs for the instance *A-n37-k5*. In order to measure the quality of the front, we used the most common metrics in multi-objective optimization that address the indicators of both *convergence* and *diversity* [6], [12]:

1. **Inverted Generational Distance (IGD)** shows the deviation of the smallest distances between the points of the optimal front with respect to the approximate front [4], defined as shown in 6

$$IGD(P,S) = \frac{\left(\sum_{i=1}^{|P|} d_i^q\right)^{1/q}}{|P|} \tag{6}$$

   where P is the approximate front and S is the optimal front, d_i is the minimum distance between each point of the optimal front with each point of the approximate front and $q = 2$.

2. **Hyper Volume Indicator (HV)** classified in [4] as a metric that measures convergence and diversity simultaneously, was proposed by [20]. In particular, the closer the solutions of S are to the true PFs, the larger is the value of HV. At the same time, a higher HV could also indicate solutions of S are scattered more evenly in the objective space. However, The downside of HV lies in the high computational complexity.

$$HV(S) = volume\left(\bigcup_{i=1}^{|S|} v_i\right) \tag{7}$$

   where v_i is the volume of the *hyper-cube* formed by the point i of the front S and a reference point (nadir point).

The statistical information of the fronts obtained for the instance $A-n37-k5$ are shown in the figures 1 and 2.

As shown in figure 1, the obtained IGD values show little variation, without presence of outliers. In figure 2 greater variation is observed, although without the presence of outliers. When comparing the variation coefficients as shown in the table 3 for both samples, it can be affirmed that both are homogeneous and representative.

Additionally, the fronts with the best indicators of IGD and HV were compared with the front obtained with the exact method. The fronts are shown in the figure 3. The front presented in the figure 3

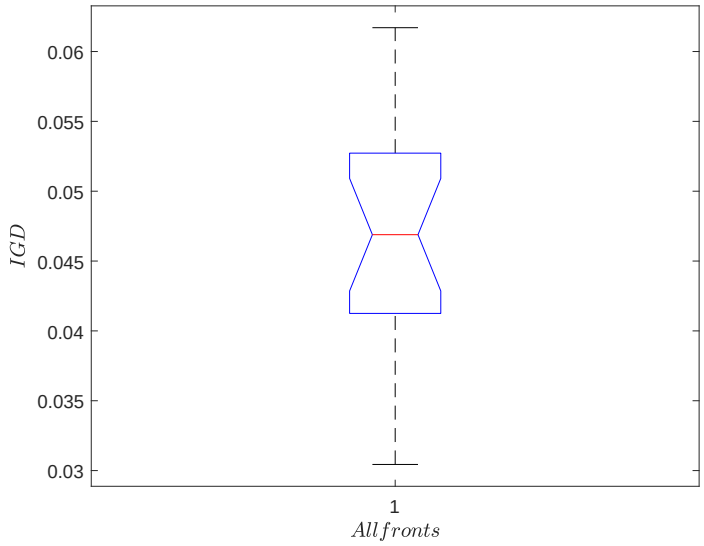


Figure 1: IGD obtained from A-n37-k5 instance [1]

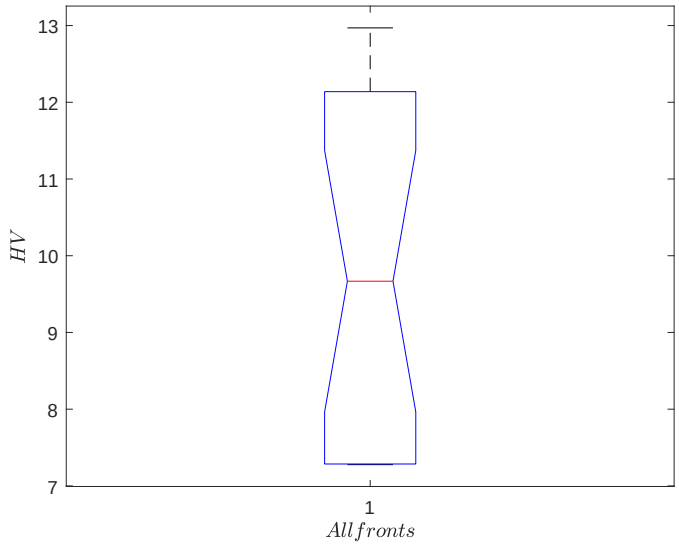


Figure 2: HV obtained from A-n37-k5 instance [1]

Descriptive	IGD	HV
Mean	0.05	9.88
Standard Deviation	0.01	2.29
Var Coeff	0.18	0.23

Table 3: Description of the data obtained for IGD and HV metrics.

marked with green corresponds to the front with the best value (0.0304) for the metric IGD. On the other hand, the front in the figure 3 marked with blue corresponds to the front with the best value (12.9694) for the metric HV.

Finally, different fronts obtained for some instances of the group presented in [1] are shown. For each instance 20 independent runs were made, the front with the highest value for HV is highlighted with red color. The fronts obtained are shown in the figures 4, 5, 6 and 7.

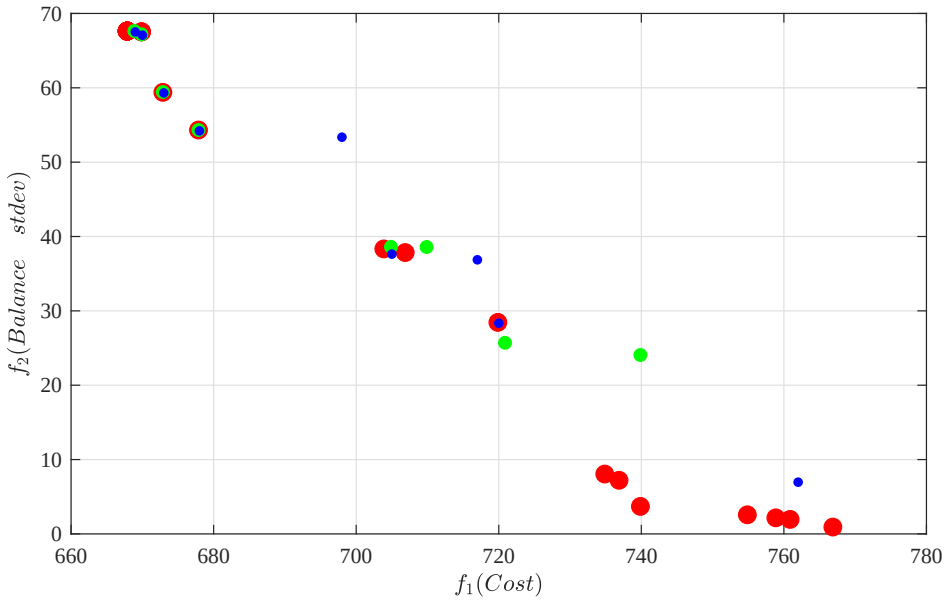


Figure 3: Obtained fronts from A-n37-k5 instance [1]

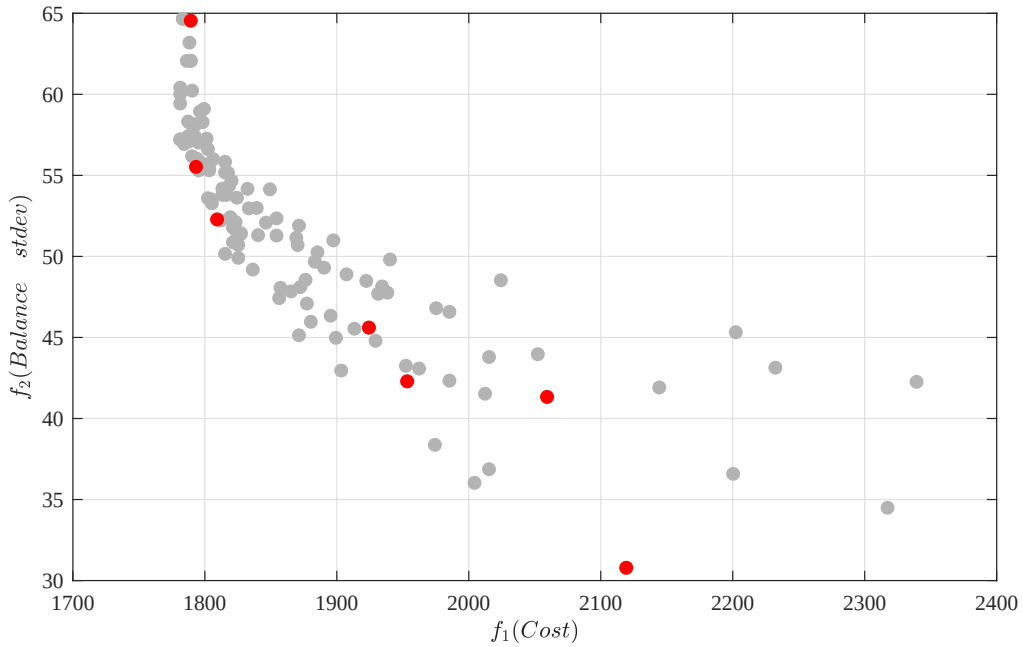


Figure 4: Obtained fronts from A-n80-k10 instance [1]

4 Conclusions and Future Work

A Multi-objective algorithm was implemented that resolves the bi-CVRP taking into account the cost of the routes and their balance. In this algorithm the decomposition concept was used, for the solution of the subproblems the ILS was used. The results are initially confronted using an exact method, with instances of low mathematical complexity. Finally, the algorithm is validated with instances of medium and high mathematical complexity.

An exact model was proposed to solve the bi-CVRP problem that seeks to optimize, in addition

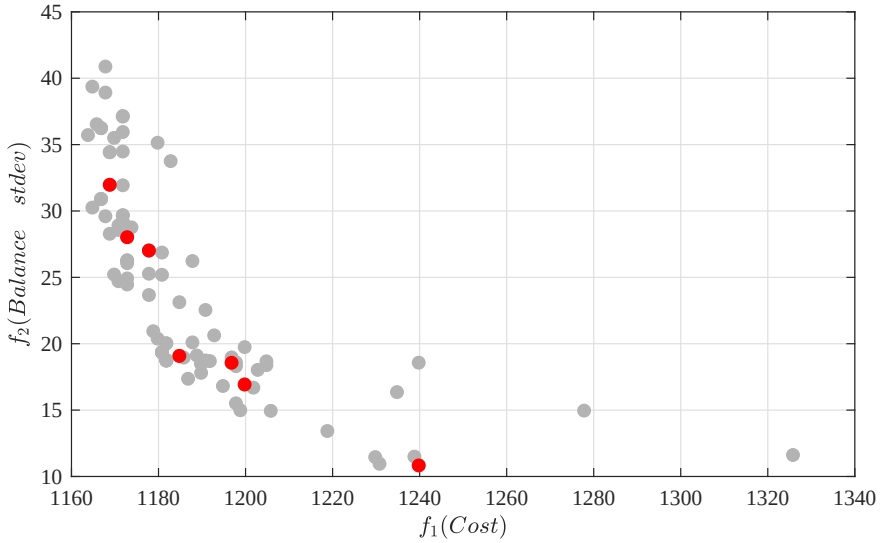


Figure 5: Obtained fronts from A-n69-k9 instance [1]

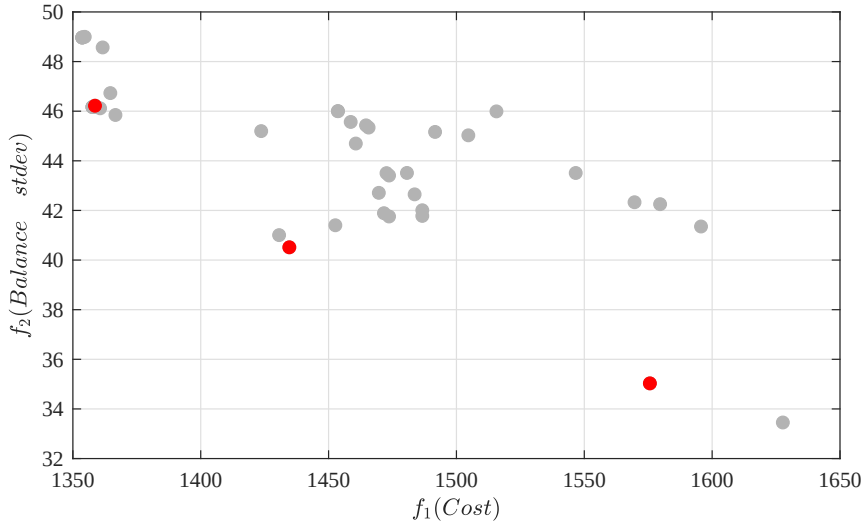


Figure 6: Obtained fronts from A-n60-k9 instance [1]

to the economic costs, the workload of the drivers through the balancing of the length of the routes. The exact model is used to validate the proposed methodology, using for this purpose instances of low mathematical complexity.

An algorithm based on decomposition taken from $MOEA/D$ and ILS was implemented, the first one decomposes the Multi-objective problem into N Mono-objective subproblems and the second solves each one of these subproblems. The algorithm implemented showed great efficiency in generating fronts, whose fundamental characteristics are diversity between solutions and convergence.

When analyzing the results using the IGD and HV metrics, an adequate distribution and convergence of the fronts is observed, with which an adequate behavior of the proposed algorithm in the solution of this problem is verified.

Since the problem is decomposed into N independent subproblems, it is possible to apply parallel processing, in such circumstances N threads are generated that are processed simultaneously.

It is proposed to carry out a comparison between the MOEA/D, traditionally used in the solution of this type of problems and the decomposition algorithm proposed in this work.

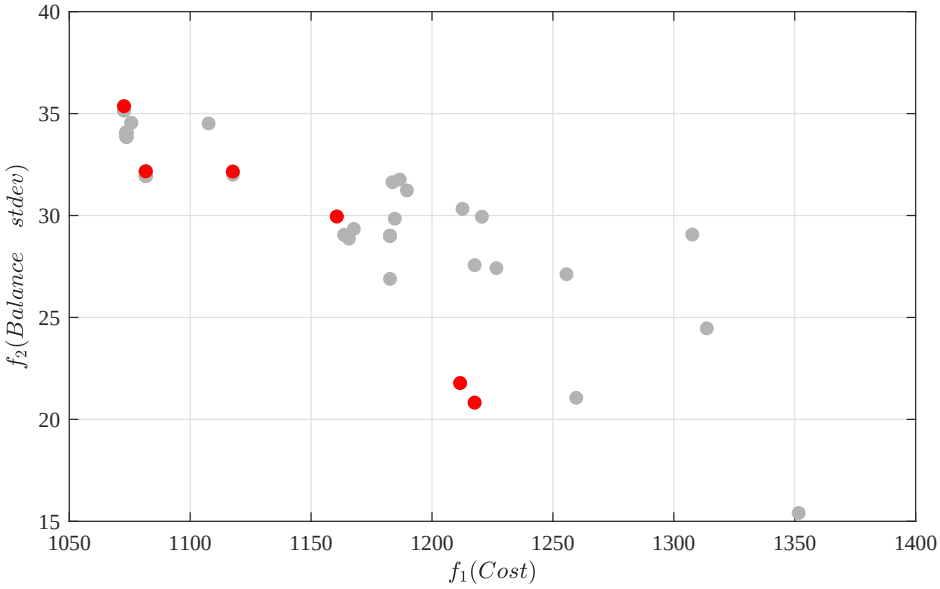


Figure 7: Obtained fronts from A-n55-k9 instance [1]

References

[1] Ph Augerat, Jose Manuel Belenguer, Enrique Benavent, A Corberán, D Naddef, and G Rinaldi. Computational results with a branch-and-cut code for the capacitated vehicle routing problem. 1998.

[2] Luis Fernando Galindres-Guancha, Eliana Mirledy Toro-Ocampo, and Ramón Alfonso Gallego-Rendón. Multi-objective MDVRP solution considering route balance and cost using the ILS meta-heuristic. *International Journal of Industrial Engineering Computations*, 9(1):33–46, 2018.

[3] Stefan Irnich, Paolo Toth, and Daniele Vigo. Chapter 1: The Family of Vehicle Routing Problems. In *Vehicle Routing*, pages 1–33. Society for Industrial and Applied Mathematics, Philadelphia, PA, nov 2014.

[4] Siwei Jiang, Yew-Soon Ong, Jie Zhang, and Liang Feng. Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE transactions on cybernetics*, 44(12):2391–2404, 2014.

[5] T.-R. Lee and J.-H. Ueng. A study of vehicle routing problems with load-balancing. *International Journal of Physical Distribution & Logistics Management*, 29(10):646–657, dec 1999.

[6] Kaiwen Li, Rui Wang, Tao Zhang, and Hisao Ishibuchi. Evolutionary many-objective optimization: A comparative study of the state-of-the-art. *IEEE Access*, 6:26194–26214, 2018.

[7] Jairo Lozano, Luis-Carlos González-Gurrola, Eduardo Rodriguez-Tello, and Philippe Lacomme. A statistical comparison of objective functions for the vehicle routing problem with route balancing. In *2016 Fifteenth Mexican International Conference on Artificial Intelligence (MICAI)*, pages 130–135. IEEE, 2016.

[8] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[9] P. Matl, R. F. Hartl, and T. Vidal. Workload Equity in Vehicle Routing Problems: A Survey and Analysis. *Transportation Science*, page trsc.2017.0744, 2017.

[10] Nader Ghaffari Nasab, M. Saeed Jabalameli, Ali Saboury, Juan J. Durillo, Qingfu Zhang, Antonio J. Nebro, Enrique Alba, Multi-objective Vrp, A Survey, Yi Mei, Ke Tang, Xin Yao, Qingfu Zhang, and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6683 LNCS(6):140, 2011.

[11] Peter Reiter and Walter J Gutjahr. Exact hybrid algorithms for solving a bi-objective vehicle routing problem. *Central European Journal of Operations Research*, 20(1):19–43, 2012.

[12] Nery Riquelme, Christian Von Lücken, and Benjamin Baran. Performance metrics in multi-objective optimization. In *2015 Latin American Computing Conference (CLEI)*, pages 1–11. IEEE, 2015.

[13] Funda Samanlioglu. A multi-objective mathematical model for the industrial hazardous waste location-routing problem. *European Journal of Operational Research*, 226(2):332–340, apr 2013.

[14] Silvia Schwarze and Stefan Voß. Improved load balancing and resource utilization for the Skill Vehicle Routing Problem. *Optimization Letters*, 7(8):1805–1823, 2013.

[15] Anand Subramanian. Heuristic, exact and hybrid approaches for vehicle routing problems. *Universidad Federal Fluminense. Tesis Doctoral. Niteroi*, page 13, 2012.

[16] Anand Subramanian, LAF Cabral, and LS Ochi. An efficient ils heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Relatório Técnico, Universidade Federal Fluminense, disponível em http://www. ic. uff. br/˜ satoru/index. php*, 2008.

[17] Zhenkun Wang, Qingfu Zhang, Maoguo Gong, and Aimin Zhou. A replacement strategy for balancing convergence and diversity in moea/d. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2132–2139. IEEE, 2014.

[18] Q. Zhang and H. Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[19] Aimin Zhou and Qingfu Zhang. Are all the subproblems equally important? resource allocation in decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary computation*, 20(1):52–64, 2016.

[20] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer, 1998.

Applying Speed-Up Techniques to Local Search and Solving the Traveling Salesman Problem

Maša Avakumović, Martin Josef Geiger

Helmut Schmidt University/University of the Federal Armed Forces
Holstenhofweg 85, 22043 Hamburg, Germany
{masa.avakumovic,m.j.geiger}@hsu-hh.de

Abstract

The main goal of our work is to quickly find a high-quality approximation for a Traveling Salesman Problem (TSP) solution. In our approach we used the logic of the Iterated Local Search (ILS) with adequate initialization methods and efficient implementation techniques, which contribute to the faster convergence towards the optimal solution. An independent assessment and comparison with other implementation methods was carried out on the optil.io platform [6], where we currently hold the top 2 positions.

1 Introduction

In our work, we deal with the classical symmetric traveling salesman problem for finding a Hamiltonian cycle in a graph. The resulting cyce has the minimum cost and therefore it is the optimal tour for the traveling salesman. Under the assumption that graph has n nodes, a tour $t = (x_1, x_2, \ldots, x_n)$ is a permutation of the vector $(1, 2, \ldots, n)$ with the corresponding cost $C(t) = \sum_{i=1}^{n-1} d(x_i, x_{i+1}) + d(x_n, x_1)$ where d_{x_i,x_j} represents the Euclidean distance between two nodes in the tour. Over the years, there have been many attempts with several high-quality implementations, including the one of today's most precise TSP solver: Concorde [1]. However, Concorde focuses on exact solving large datasets, rather than producing fast solutions for small examples. Motivated by finding the best possible solution in a very short time period, we focus our analysis on the local search heuristic method and present some implementation techniques for gaining the time advantage.
Among other numerous application in which the speed of the solving a TSP plays a crucial role, we underline the Vehicle Routing Problem (VRP), where a need for a high-quality TSP solution arises for each vehicle individually, numerous times. Whereas the most of the VRPs pose their restrictions to the vehicle's capacity, there are also some problems where the length of a single route is restricted to the maximum length. However, the check if the length of the newly created route exceeds the a priori given limit is again a TSP for itself because the length of a route naturally depends on the order in which the customers are visited. Therefore, a fast and efficient method is needed.

2 Solution approach

When solving TSPs with the speed of finding a solution as the most important criterion, exact algorithms are often infeasible. Consequently, different heuristics are used for approximating the solution. Our solution method applies the Iterated Local Search (ILS) [5], a high-quality and relatively easy to implement local search heuristic, that solves the problem of getting stuck in the local minimum in a considerably efficient fashion. This escape is achieved with *perturbation* operator that explores the neighboring solutions as a starting solutions for the repeated local search. The size of the neighborhood is growing proportionally with the number of iterations that did not find any better solution. Local search that we iterated in our experiments is a well-known *2-opt* local search method.
Another factor that plays a crucial role by evaluating the efficiency and the speed of a solution approach is the implementation itself. In that spirit, we reveal some of the techniques that allowed us to reach the top of the optil.io ranking in the ongoing TSP challenge. Those implementation techniques consist of

simple checks that by omiting some unnecessary computations accelerate the local search towards the local optimum.
An initial start advantage for our ILS is obtained by initialization method, whose type depends on the data distribution itself. Three different initialization types are implemented; nearest neighborhood, random, and the grid like construction method [4]. However, we gain the most time by applying simple checks that we embedded into our *2-opt* local search. Here we suggest the following checks in order to save the computing time.

1. (Check 1) The first check is to compare the lower bound (the sum of the new edge and the minimum edge in the current tour) and the upper bound (the sum of the old edge and the maximum edge in the current tour).

$$d_{x_i,x_j} + d_{min} > d_{x_j,x_{j+1}} + d_{max}$$
$$d_{x_{i+1},x_{j+1}} + d_{min} > d_{x_i,x_{i+1}} + d_{max}$$

   In those cases in which the lower bound exceeds the upper bound, the iteration immediately proceeds to the next node in the permutation.

2. (Check 2) An additional check that significantly accelerated finding a high-quality solution is inspired by [3]. If the length of one of the new edge exceeds the sum of the two old edges, we again proceed to the next node in the permutation.

$$d_{x_i,x_j} > d_{x_i,x_{i+1}} + d_{x_j,x_{j+1}}$$
$$d_{x_{i+1},x_{j+1}} > d_{x_i,x_{i+1}} + d_{x_j,x_{j+1}}$$

In addition to the aforementioned checks, a significant speed contribution is gained by only summing up the two new edges $d_{x_i,x_j} + d_{x_{i+1},x_{j+1}}$ and comparing the length with the sum of the length of the two old edges $d_{x_i,x_{i+1}} + d_{x_j,x_{j+1}}$, instead of calculating the entire tour length in each iteration of our local search. In the literature, this is known as a delta evaluation [2]. Finally, the node x_i will be connected with x_j only when the first sum is actually smaller.

3 Results and discussion

The performance of the previously defined checks was measured by *saving coefficient 1 (SC1)* and *saving coefficient 2 (SC2)* which are defined as a percentage of iterations where the calculation of delta evaluation was avoided due to the positive response of the Check 1 or Check 2. The total number of distinct edges to be compared in a tour with n nodes can be calculated using the formula $\frac{(n-1)*(n-2)}{2}$. Naturally, to find the solution closest to the optimum, the main goal of the *2-opt* iterations is to choose those two edges that result with the highest delta evaluation (the sum of the two new edges is substracted from the sum of the two old edges). Our implementation uses the *best improvement* technique, where the search method stops only if all of the possible new edges for the current node are checked ($n - 2$ of them). Only then is the current tour modified by adding the two new edges and removing two old edges. In our experiments we used mid-sized datasets ($1000 < n < 2500$) from the TSLIB library and each experiment was granted only one CPU second. For each dataset, a total number of distinct edges in a tour, saving coefficient 1 and 2 and the corresponding gap, are calculated and reported in Table 1. The gap represents the difference between our solution and the known optimum (in percentage). Since the two saving coefficients are proportional to the number of skipped delta evaluations, they represent a suitable measure for the amount of time that is saved.
Our checks are complementary, meaning that the increase of the SC1 will have a decreasing effect on the SC2. This was tested by carrying out multiple experiments on the same datasets but different starting

Dataset	Total number of checks	SC1	SC2	Gap with checks	Gap without checks
pr1002	500500	0.133	99.40	9.09	9.56
u1060	560211	81.57	18.02	8.14	8.22
vm1084	585903	0.04	99.59	10.01	11.08
rl1304	848253	0.08	99.69	12.98	13.95
rl1323	873181	0.01	99.72	12.22	12.64
d1219	831405	0.01	99.34	13.67	14.53
d1655	1367031	0.05	99.66	12.16	12.51
nrw1379	948753	0.23	96.42	8.94	9.24
vm1748	1525131	0.07	95.71	12.04	12.66
u1432	1023165	97.43	3.56	13.42	14.16
u1817	1648020	95.32	4.30	13.80	13.93
rl1889	1781328	0.04	94.47	15.21	15.88
u2152	2312325	95.89	3.88	13.55	14.35
u2319	2685403	91.52	7.85	11.94	12.23

Table 1: The effect of savings coefficients to the final solution after exactly one CPU second.

solutions. With a better approximation of the start solution, the number of times when the Check 1 has a positive response is decreased and consequently the SC1 value. This decrease is a consequence of the shorter distances between nodes, therefore the minimum (maximum) edge needed for calculating the lower (upper) bound in the Check 1 are also itself. smaller.
Implementing the two previously defined checks creates a time saving effect for every tested dataset. In consequence, this time advantage allows us to reduce the gap to the optimum, as can be seen by comparison of the two last columns in Table 1.
Our experiments demonstrate that cleverly pruning the iterations of the local search can lead to better results within a short period of time. We are convinced that further experiments and additional checks have the potential to further improve the known local search methods.

References

[1] David Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. Concorde. 2004.

[2] Mauro Birattari, Prasanna Balaprakash, Thomas Stützle, and Marco Dorigo. Estimation-based local search for stochastic combinatorial optimization using delta evaluations: a case study on the probabilistic traveling salesman problem. *INFORMS Journal on Computing*, 20(4):644–658, 2008.

[3] Martin Josef Geiger. Algorithms, pseudo-codes, implementations–some reflections on interdependencies and potential implications. *Electronic Notes in Discrete Mathematics*, 69:37–44, 2018.

[4] David S. Johnson. Local optimization and the traveling salesman problem. In Michael S. Paterson, editor, *Automata, Languages and Programming*, pages 446–461, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[5] Helena Lourenço, Olivier Martin, and Thomas Stützle. Iterated local search. In Fred Glover and Gary A. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57, pages 320–353. Springer, 2003.

[6] Szymon Wasik, Maciej Antczak, Jan Badura, Artur Laskowski, Maciej Olszowy, Tomasz Sternal, and Krzysztof Wedrowicz. Optil.io platform: Evaluation as a service for metaheuristics. In *Proceeding of the MIC and MAEB 2017 Conferences*, pages 225–227. University Pompeu Fabra, 2017.

A vehicle routing problem with periodic replanning

Guido Ignacio Novoa–Flores[1], Luisa Carpente[1], Silvia Lorenzo–Freire[1]

MODES Research Group, Department of Mathematics, Faculty of Computer Science, University of A Coruña
Campus de Elviña s/n, 15071, A Coruña, Spain
guido.novoa@udc.es, luisacar@udc.es, slorenzo@udc.es

Abstract

In this work we focus on the problem of truck fleet management of the company GESUGA. This company is responsible of the collection and proper treatment of animals not intended for human consumption in Galicia (Spain). On a daily basis, the company must design routes for the following day with the uncollected orders until that moment. However, these routes may be replanned during their execution as new orders appear from customers and the company is interested in collecting it. Thus, the aim of this work is to provide a tool that computes good quality solutions in short time. The problem treated belongs to the family MDCVRPTW with the particularity of the route redesign. For its resolution we have adapted linear programming models, simulation techniques and metaheuristic algorithms.

1 Introduction

After the spread of Bovine Spongiform Encephalopathy, typically known as mad cow disease, the European Union took a number of measures (e.g. Regulation (EC) No 999/2001) to prevent its spread and transmission. This regulation forbids the burial of carcasses of dead animals at livestock farms. In Galicia, one of the main companies responsible for this task is GESUGA. This company focuses its business area on the integrated management of meat by-products not intended for human consumption. Its main activity consists of the collection and transport of the different meat by-products, generally animal carcasses, from livestock farms to treatment plants, for proper processing.

To serve customers, the company has 32 trucks distributed in its three plants (12 in the main plant, 10 in the remaining plants). From Monday to Friday, customers are served and products are transported to the different plants. Taking into account the characteristics of this problem, it could be classified as a Multi Depot Capacitated Vehicle Routing Problem with Time Windows (MDCVRPTW).

2 Description of the problem

As mentioned in the introduction, the company has to visit its customers all over Galicia on a daily basis. However, the company must take into account the following restrictions:

- Trucks leave and return from the plant to which they are assigned only once a day.
- Drivers have a maximum 8 hours working day which includes a rest and disinfection of the vehicle at the end of the day.
- Trucks have a maximum loading capacity.
- Orders must be picked up within 48 hours from receipt.

Currently, route planning is manually made by the logistics department and it is organized in two phases:

- Offline Phase: It is made the day before according to the following steps:
  - At 19:00 there are some pre-routes with the uncollected orders until that moment.

  - At 20:00 these pre-routes are reviewed with the logistics manager adding new orders and making the necessary changes.
  - At 21:00 drivers receive the set of places that they must visit, but they are free to organize it.

- Online Phase: It is manually made during the collection of the orders planned in the offline phase considering the incoming orders.

Note that the route design is manually made by the logistics department and drivers are free to organize their routes. Hence, the efficiency of the company depends deeply on human knowledge. Therefore, the company is interested in a tool to calculate routes automatically, satisfy the needs of customers and achieve the following objectives:

- Minimize the ratio between the distance travelled and orders collected.
- Minimize the ratio between trucks used and orders collected.

3 Implementation of the algorithm

The implementation of the algorithm was made in JAVA language using open source libraries. For each of the phases mentioned the algorithm works as follows:

- **Offline Phase:**
  - All the uncollected orders[1] are assigned to its nearest plant by solving the generalized allocation problem (GAP).
- **Online Phase:**
  - Routes of each plant are computed according to the Ruin and Recreate principle described in [2] combined with the strategies proposed in [1]. In case of considering simulated orders, they are eliminated after computing each route and they are reorganized by solving the travelling salesman problem (TSP).
  - The incoming orders are assigned to a plant by solving the GAP and are assigned to one of the existing routes considering the position of trucks. Note that for this step, there is a replanning of the routes and the user must decide the frequency of the replanning.

Currently, the following strategies are being considered:

- Lazy: No orders are collected during the online phase, i.e., the routes computed in an offline phase are not modified during its execution.
- Minimum$-k$: A truck leaves the plant when, at least, k orders are assigned to it. Thus, it could happen that a truck leaves the plant during the offline phase or the online phase in a replanning.

  Many implementations have been considered for the two strategies varying some parameters: minimum number of orders needed to take out a truck, time at which the replanning in the online phase is performed, hours delaying the departure of trucks or number of simulated orders. Table 1 details the 6 implementations.

Name	Description
Lazy	Lazy strategy is used everyday.
Lazy & Delay	Lazy strategy is used everyday and a delay is applied on Fridays.
Lazy & Minimum$-k$	Lazy strategy is used everyday but the Minimum$-k$ strategy is used on Fridays.
Minimum$-k$	Minimum$-k$ strategy is used everyday.
Minimum$-k$ & Lazy	Minimum$-k$ strategy is used but the Lazy strategy is used on Fridays.
Minimum$-k$ & Delay	Minimum$-k$ strategy is used and a delay is applied on Fridays.

Table 1: Description of the different strategies implemented.

Strategy	Orders collected	Distance	Ratio Distance/Orders
Lazy	5872	60362.3	10.279
Lazy & Delay	5872	61807.89	10.526
Lazy & Minimum$-k$	5872	62561.71	10.654
Minimum$-k$	6104	71457.68	11.707
Minimum$-k$ & Lazy	6098	66764.08	10.95
Minimum$-k$ & Delay	6081	61407.3	10.09
Real Case	6040	66885.6	11.073

Table 2: Results obtained with the different strategies.

4 Results

The best results achieved for each implementation and the results obtained for the company are summarized in Table 2.

Note that, for any of the Lazy strategies the number of orders collected is 5872. Thus, following objective 1 the best option would be to use the Lazy strategy. Comparing with the real case all the Lazy implementations improve the ratio distance by order although the number of orders collected (5872) is far from the real case (6040).

In order to collect more orders than the real case, any of the Minimum$-k$ strategies should be selected. On the one hand, the Minimum$-k$ implementation can collect 6104 orders but the ratio distance by order (11.707) worsens in comparison with the real case (11.073). The others implementations of the Minimum$-k$ strategy collect more orders and improve the ratio distance by order than the real case. It is clear that the Minimum$-k$ & Delay implementation performs the best results.

References

[1] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.

[2] Gerhard Schrimpf, Johannes Schneider, Hermann Stamm-Wilbrandt, and Gunter Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171, 2000.

[1]Optionally, it is possible to include simulated orders. These orders can be obtained considering historical data and can lead to more realistic routes.

An Iterated Greedy Heuristic for the Minimum-Cardinality Balanced Edge Addition Problem

Ruben Interian, Celso C. Ribeiro

Universidade Federal Fluminense, Institute of Computing, Niterói, RJ 24210-346, Brazil.
rinterian@ic.uff.br, celso@ic.uff.br

Abstract

The Minimum-Cardinality Balanced Edge Addition Problem (MinCBEAP) appears in the context of polarized networks as a strategy to reduce polarization by external interventions using the minimum number of edges. We show that every instance of MinCBEAP can be reduced to an instance of the Minimum-Cardinality-Bounded-Eccentricity Edge Addition Problem (MCBE). A restarted iterated greedy heuristic is developed for solving MinCBEAP via the transformed MCBE. Preliminary computational results are reported.

1 Problem statement

According to the Oxford Dictionaries, polarization is the division into sharply contrasting groups or sets of opinions or beliefs [1]. Academic articles, newspapers, and the media in general constantly report the growth of fake news, misinformation spreading, and polarization. These phenomena are closely interrelated with each other. Fake news spread faster in polarized networks or groups [6]. At the same time, fake and tendentious news can accentuate polarization within already existing echo chambers in the social networks.

Interian and Ribeiro [3] showed that many case studies real-world networks are extremely polarized. A polarized network is divided into two or more strongly connected groups, with few edges between vertices belonging to different groups. Most of the time, only same-group vertices communicate to each other and most of the information that a vertex can receive comes from inside the same group to which it belongs.

In order to reduce polarization, networks can be treated by external interventions consisting of the addition or the removal of vertices and edges. In this work, we address a new optimization problem that treats the issue of polarization reduction by edge additions. Given a graph $G = (V, E)$ and a subset of (polarized) vertices $A \subset V$, we seek a minimum-cardinality set of edges $E' \subseteq V \times V \setminus E$ to be added to G such that all vertices that are not in A can be reached from any vertex in a A by paths with at most D edges, where D is a problem parameter. The decision version of our optimization problem can be cast as:

Minimum-Cardinality Balanced Edge Addition Problem (MinCBEAP):
Instance: Graph $G = (V, E)$, subset $A \subset V$, integer D, integer L.
Goal: Is there a set $E' \subseteq (V \times V) \setminus E$ with cardinality $|E'| \leq L$ such that $d_{G'=(V,E\cup E')}(v, V \setminus A) \leq D$, $\forall v \in A$?

In the above formulation, $d_{G'=(V,E\cup E')}$ denotes the number of edges in a shortest path from $v \in A$ to the closest vertex in $V \setminus A$ in $G' = (V, E \cup E')$.

Given a graph $G = (V, E)$, the eccentricity $\epsilon(v)$ of a vertex $v \in V$ is the longest of the shortest paths in G from v to all other vertices in $V \setminus \{v\}$, see [2]. Any instance of MinCBEAP can be reduced to an instance of the Minimum-Cardinality-Bounded-Eccentricity Edge Addition Problem (MCBE):

Minimum-cardinality-bounded-eccentricity edge addition problem (MCBE):
Instance: Graph $G = (V, E)$, source vertex $s \in V$, integer B, integer p.
Question: Is there a supergraph $G' = (V, E \cup E')$ of G with $E' \subseteq (V \times V) \setminus E$ such that $|E'| \leq p$ and $\epsilon_{G'}(s) \leq B$?

Interian and Ribeiro [4] showed that MCBE is NP-complete. They also showed that MinCBEAP is NP-complete for D greater than or equal to 2, using a polynomial transformation from MCBE.

We propose in the following an efficient iterated greedy heuristic for approximately solving MinCBEAP using a transformation to MCBE. This article is organized as follows. In Section 2, we describe the polynomial transformation from an instance of MinCBEAP to one of MCBE and the iterated greedy heuristic. In Section 3, some preliminary computational results are presented.

2 Iterated greedy heuristic

We propose the following transformation of an instance of MinCBEAP into an instance of MCBE. Consider an instance of the optimization version of MinCBEAP defined by a graph $G = (V, E)$, a subset $A \subset V$, and an integer D. To build the corresponding optimization instance of MCBE, consider a graph $\bar{G} = (\bar{V}, \bar{E})$ and an integer $B = D$. Let $\bar{V} = A \cup \{s\}$, where s is the source vertex representing the set $V \setminus A$, in such a way that if there is an edge $(u, v) \in E$ with $u \in A$ and $v \in V \setminus A$, then there will be an edge $(u, s) \in \bar{E}$.

The MCBE instance is solved after the problem transformation. Let $S_H \subseteq (\bar{V} \times \bar{V}) \setminus \bar{E}$ be the optimal solution of MCBE for the instance defined by $\bar{G} = (\bar{V}, \bar{E})$. This solution is transformed to a solution $S_G \subseteq (V \times V) \setminus E$ of the original MinCBEAP instance as follows. Consider any edge $(u, v) \in S_H$. If both $u, v \in A$, then the edge (u, v) also belongs to S_G. Otherwise, in case the source $s \notin A$ coincides e.g. with extremity u of edge (u, v), then edge (w, v) is placed in S_G, where w is a randomly generated vertex from $V \setminus A$. Following this strategy, we obtain a solution S_G that solves MinCBEAP satisfying $|S_H| = |S_G|$.

The heuristic proposed in this work is based on the following lemma (the proof is omitted here due to space limitations):

Lemma 2.1. *There is a solution $S^* \subseteq (V \times V) \setminus E$ for MCBE such that all edges in S^* are incident to the source vertex s.*

Therefore, the set of candidate edges to be inserted in the solution can be restricted to those incident to vertex s. The problem is then reduced to determining a subset $X^* \subseteq V \setminus \{s\}$ of vertices such that for every vertex $v \in X^*$ there is an edge $\{v, s\} \in S^*$. In the algorithm, the solution S^* will be represented by the corresponding subset X^* of vertices, where $|S^*| = |X^*|$.

The cost of a solution to MCBE is given by the cardinality $|X|$ of the set X. It is well known that local search procedures often do not perform satisfactorily in the context of optimization problems without weights, in which the objective function is related to the cardinality of the solution, due to the lack of gradient information and to the exponential size of neighborhoods based on cardinality improving. In order to overcome this barrier, we propose in the following an iterated greedy heuristic [7] for MCBE, which is solely based on the repeated, successive application of destruction and reconstruction procedures, without appealing to local search.

In summary, iterated greedy starts from a greedy or a semi-greedy candidate solution, and generates a sequence of solutions using two main phases: destruction and reconstruction. During the destruction phase, some edges are removed from the current solution. The reconstruction procedure then applies a greedy constructive heuristic to reconstruct a complete candidate solution. If the cardinality of the newly constructed solution is less than the cardinality of the incumbent solution, then the latter is updated. The heuristic iterates over these steps until some stopping criterion is met [7].

One important building block of the iterated greedy heuristic is the semi-greedy (or greedy randomized) algorithm. There is a number $p(v)$ of non-solved neighbors associated with each vertex $v \in V$, that is, the number of neighbors of v that are still at a distance greater than B from the source vertex s. At each step, this algorithm adds one random vertex from a restricted candidate list (RCL) to the initially empty solution set X. The RCL is formed by all vertices $w \in V \setminus \{s\}$ that satisfy the condition $(1 - \alpha)p_{max} \leq p(w) \leq p_{max}$, where $p_{max} = max\{p(v) : \ v \in V \setminus \{s\}\}$ and α is the greediness

parameter of the semi-greedy algorithm. We recall that $\alpha = 0$ corresponds to a purely greedy criterion, while $\alpha = 1$ corresponds to a completely random selection of the next vertex from $V \setminus \{s\}$.

Algorithm 1 shows the pseudo-code of the proposed iterated greedy heuristic. In line 1, the best solution X^* and its cardinality f^* are initialized. The while loop in lines 2–16 performs a new iteration while a stopping criterion is not met. The semi-greedy algorithm is applied in line 3 to build a solution X using the greediness parameter α. This solution is improved by performing a sequence of destruction-reconstruction phases in lines 4–12, until k_{max} iterations without any improvement are executed. The destruction phase is applied in line 7, by randomly removing a fraction β of the vertices in X. Next, the resulting solution is reconstructed by a greedy deterministic algorithm in line 8, which corresponds to the semi-greedy algorithm with $\alpha = 0$. If the cardinality of the newly reconstructed solution X' is less than the cardinality of the incumbent X, then the latter is updated in line 10. If the solution obtained after the sequence of destruction-reconstruction iterations is better than the best known solution X^*, then the best solution X^* and its cardinality f^* are updated in line 14. In line 17, the best solution found X^* is returned.

Algorithm 1 Restarted Iterated Greedy for MCBE

Parameters: α, β, k_{max}
Input: $G = (V, E)$
Output: X^*

1: $X^* \leftarrow V$; $f^* \leftarrow |V|$;
2: **while** stopping condition is not met **do**
3: $X \leftarrow SemiGreedy(G, \alpha)$;
4: $i \leftarrow 0$;
5: **while** $i \leq k_{max}$ **do**
6: $i \leftarrow i + 1$;
7: $X' \leftarrow PartialDestruction(G, X, \beta)$;
8: $X' \leftarrow GreedyReconstruction(G, X')$;
9: **if** $|X'| < |X|$ **then**;
10: $X \leftarrow X'$; $i \leftarrow 0$;
11: **end if**;
12: **end while**;
13: **if** $|X| < f^*$ **then**
14: $X^* \leftarrow X$; $f^* \leftarrow |X|$;
15: **end if**;
16: **end while**;
17: **return** X^*;

In the above algorithm, an external loop was added to the iterated greedy heuristic originally described by Ruiz and Stützle [7]. Instead of starting with a completely greedy solution, we start with a semi-greedy solution and the iterated greedy construction can be embedded in a multi-start procedure. This hybrid variant was named Restarted Iterated Greedy (RIG) by Pinto et al. [5].

3 Preliminary numerical results

We generated 13 random graph instances with different number of vertices. The number of edges in each instance was fixed as 8 times the number of vertices. The maximum length of the connecting paths was fixed as $D = 2$. The greediness parameter of the restarted iterated greedy was set at $\alpha = 0.1$, the fraction of the solution to be destructed at $\beta = 50\%$, the number of iterations without improvement at $k_{max} = 100$, and the time limit of 3600 seconds as the stopping criterion for the heuristic.

Table 1 shows preliminary results for the random instances, including the number of vertices $|X^*| = |S^*|$ in the best solution found and the running time in seconds. We are currently develo-

ping a mixed integer programming (MIP) approach for exactly solving small- and medium-size instance of MCBE. Preliminary results obtained by the MIP approach make it possible to identify the optimality of the solutions found by the restarted iterated greedy heuristic for the instances with up to 1000 vertices.

Instance	$\|A\|$	$\|X^*\|$	Time (s)	Solved?
inst_100v_8x	100	3	0.32	yes
inst_200v_8x	200	7	0.64	yes
inst_500v_8x	500	20	0.19	yes
inst_1000v_8x	1000	38	246.08	yes
inst_2000v_8x	2000	81	3610.24	(no)
inst_3000v_8x	3000	121	3609.47	(no)
inst_4000v_8x	4000	160	3609.60	(no)
inst_5000v_8x	5000	209	3606.40	(no)
inst_6000v_8x	6000	253	3614.21	(no)
inst_7000v_8x	7000	292	3621.25	(no)
inst_8000v_8x	8000	331	3617.79	(no)
inst_9000v_8x	9000	392	3638.78	(no)
inst_10000v_8x	10000	426	3644.16	(no)

Table 1: Results for the restarted iterated greedy heuristic.

References

[1] Oxford Dictionaries. Definition of polarization. Last access on 2017-09-06. https://en.oxforddictionaries.com/definition/polarization.

[2] F. Harary. *Graph Theory*. Addison Wesley, 1969.

[3] R. Interian and C. C. Ribeiro. An empirical investigation of network polarization. *Applied Mathematics and Computation*, 339:651–662, 2018.

[4] R. Interian and C. C. Ribeiro. Minimum-cardinality balanced edge addition in polarized networks: Formulations, complexity, and integer programming approaches. 2019. (Submitted for publication).

[5] B. Q. Pinto, C. C. Ribeiro, I. Rosseti, and A. Plastino. A biased random-key genetic algorithm for the maximum quasi-clique problem. *European Journal of Operational Research*, 271:849–865, 2018.

[6] M. H. Ribeiro, P. H. Calais, V. A. F. Almeida, and W. Meira Jr. Everything I disagree with is #FakeNews: Correlating Political Polarization and Spread of Misinformation. Workshop on Data Science + Journalism @KDD 2017. https://sites.google.com/view/dsandj2017/accepted-papers.

[7] R. Ruiz and T. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177:2033–2049, 2007.

A GRASP with path-relinking heuristic for the prize-collecting generalized minimum spanning tree problem

Ruslán G. Marzo[1], Celso C. Ribeiro[1]

Institute of Computing, Universidade Federal Fluminense, Niterói, RJ 24210-346, Brazil.
ruslangm@id.uff.br, celso@ic.uff.br

Abstract

The prize-collecting generalized minimum spanning tree problem is a generalization of the NP-hard generalized minimum spanning tree optimization problem. We propose a GRASP (Greedy Randomized Adaptive Search Procedure) heuristic for its approximate solution. The computational experiments showed that the heuristic developed in this work found very good optimal and suboptimal solutions for test problems with up to 439 vertices. Furthermore, we also showed that path-relinking and restart strategies consistently improved the more basic version of GRASP algorithm.

1 Introduction

Let $G = (V, E)$ be a graph defined by its vertex set $V = \{v_1, \ldots, v_n\}$ and its edge set $E \subseteq V \times V$, with a prize $p_i \geq 0$ associated to each vertex $v_i \in V$, $i = 1, \ldots, n$ and a cost $c_e \geq 0$ associated to each edge $e = (i, j) \in E$, with $i, j \in V$. Without loss of generality, we assume that graph G is complete. We also assume that the vertex set V can be partitioned in K disjoint sets or groups $V_1, \ldots, V_K$, i.e., $\bigcup_{k=1}^{K} V_k = V$ and $V_k \cap V_\ell = \emptyset$, for any $k, \ell \in \{1, \ldots, K\} : k \neq \ell$. Let $e = (i, j) \in E$ be an edge with extremities $i \in V_k$ and $j \in V_\ell$. If $k \neq \ell$, then e is said to be an intergroup edge, otherwise e is said to be an intragroup edge. The *prize-collecting generalized minimum spanning tree problem* (PC-GMSTP) consists in finding a subtree of G spanning exactly one vertex of each group $V_1, \ldots, V_K$ and minimizing the sum of the costs of the edges of the tree less the prizes of the vertices selected in each group [2, 5, 6].

According to [2, 6], PC-GMSTP was introduced by Stanojevic et al. [9] in the context of the design of regional communication networks connecting local area networks. Similar applications exist in marketing [6] and other areas.

There are several integer programming formulations for PC-GMSTP, see e.g. [2, 6]. Golden et al. [2] proposed heuristics for PC-GMSTP and adapted a local search and a genetic algorithm originally developed for the generalized minimum spanning tree problem. A genetic algorithm solution approach was proposed in [4]. In this work, we propose a new GRASP heuristic with path-relinking and restarts for approximately solving PC-GMSTP. Section 2 describes the construction and local search phases of a GRASP heuristic for PC-GMSTP. Section 3 describes the hybridization of the basic GRASP heuristic with path-relinking, which is a major enhancement that adds to the original algorithm a long-term memory mechanism using an elite set of diverse high-quality solutions. Section 4 completes the heuristic, integrating a restart strategy to diversify the search and to reduce the running times taken by GRASP with path-relinking. Computational experiments and numerical results are reported in Section 5.

2 GRASP heuristic for PC-GMSTP

GRASP, which stands for a Greedy Randomized Adaptive Search Procedure, is a multi-start metaheuristic, in which each iteration consists of two main phases: construction and local search. The first phase is the construction of a feasible solution, usually by a greedy randomized algorithm. Once a feasible solution is obtained, its neighborhood is investigated until a local minimum is found during the second phase of local search. The best overall solution is kept as the result. The reader is referred to Resende and Ribeiro [8] for a complete account of GRASP with applications.

2.1 Construction phase

The greedy randomized heuristic implemented in the construction phase is a combination of the algorithms of Kruskal and Prim for the minimum spanning tree problem Their adaptation in a heuristic for PC-GMSTP takes into account the fact that exactly one vertex should be selected from each group. It is also based on considering not only the edge costs, but also the vertex prizes. Only intergroup edges should be considered, because intragroup edges are not allowed to be part of a solution, as far as only one vertex of each group is selected as its representative.

2.2 Local search

Local search for PC-GMSTP is based on the 1-opt neighborhood defined as follows. Each solution is implicitly characterized by K representative vertices, each of them representing the vertices in one of the groups $V_1, \ldots, V_K$. Associated to each solution there are potentially $\sum_{k=1}^{K}(|V_k|-1)$ neighbors, obtained by replacing exactly one of the selected vertices in the current solution by another one in the same group. Once the subset of K representative vertices is selected, a complete solution is computed as a minimum spanning tree of the graph induced in G by the selected vertices.

We have developed two local search procedures for PC-GMSTP. The first local search procedure deterministically visits the groups of the graph in the increasing order of their labels. At any local search iteration, the procedure moves from the current solution to the best neighbor. The second local search procedure visits the groups of the graph in a random order, following a similar proposal in [2].

3 Intensification by path-relinking

Path-relinking is a major enhancement that adds a long-term memory mechanism to GRASP heuristics. GRASP with path-relinking implements this long-term memory using an elite set of diverse high-quality solutions previously identified during the search. Basically, at each iteration the path-relinking operator is applied to the solution found at the end of the local search phase and a randomly-selected solution from the elite set, returning new solutions that are similar to both of them, but possibly better. The solution resulting from path-relinking is a candidate for inclusion in the elite set [8].

Path-relinking is always carried out from an initial solution to a guiding solution. We implemented the backward strategy for path-relinking, where the guiding solution is not better than the initial solution. Normally, GRASP with backward path-relinking outperform the other path-relinking variants.

4 Diversification by restarts

Resende and Ribeiro [7, 8] have shown that restart strategies are able to reduce the running time taken by a GRASP with path-relinking heuristic to reach a target solution value for many problems.

A simple and effective restart strategy for GRASP with path-relinking is to keep track of the last iteration when the incumbent solution was improved and restart the GRASP with path-relinking heuristic if κ iterations have gone by without improvement. This strategy is called restart(κ). A restart consists in saving the incumbent and emptying out the elite set.

5 Computational experiments

The GRASP+PR heuristic was implemented in C++ with the GNU GCC compiler version 5.4.0. All experiments were performed on a computer with a 4-processor Intel Core i5-4460S CPU, with 2.90 GHz of clock frequency and 8 GB of RAM running the operating system Linux Ubuntu 16.04 LTS of 64 bits.

Since the test instances used in [2, 4] could not be obtained in the literature, we generated 30 new instances similar to those in [2]. The graphs and the distribution of their vertices in groups have been

Instance	Avg $\|\mathcal{E}\|$	Avg #PR	Avg time (s)	Max time(s)	Avg value	Best value
11EIL51	7.0	95.1	0.64	0.66	62.0	**62**
16PR76	6.9	141.5	2.30	2.35	47127.0	**47127**
20KROA100	2.9	52.2	6.75	7.03	7661.0	**7661**
20KROB100	10.0	137.6	5.99	6.18	8147.0	**8147**
20KROC100	9.0	165.0	6.07	6.20	7923.0	**7923**
20KROD100	2.4	52.5	6.49	6.72	7192.0	**7192**
20KROE100	9.7	162.3	5.99	6.17	8060.0	**8060**
21EIL101	9.9	138.0	5.46	5.65	78.0	**78**
22PR107	10.0	192.2	8.87	9.12	20360.0	**20360**
25PR124	9.6	151.3	13.71	14.08	29071.0	**29071**
28PR136	10.0	189.8	19.33	19.91	37376.0	**37376**
29PR144	3.9	115.7	24.15	24.87	39902.0	**39902**
39RAT195	10.0	197.7	73.79	76.96	511.0	511
40KROA200	9.2	195.3	82.90	86.00	11461.0	11461
40KROB200	9.8	195.0	84.73	88.33	11467.0	11467
46PR226	2.0	0.0	106.27	113.28	55254.0	**55254**
53PR264	10.0	194.0	264.52	276.78	21637.0	21637
60PR299	10.0	198.3	445.09	483.71	19999.4	19995
64LIN318	10.0	198.2	476.16	494.10	18142.5	18140
88PR439	10.0	198.8	1885.55	2048.42	51093.8	51093

Table 1: Results obtained with GRASP+PR heuristic on the 20 test instances.

provided by the authors of [1]. The prizes have been randomly generated in the interval $[0, 10]$, as for some instances in [2]. The 30 randomly generated instances are now available at Mendeley Data [3].

5.1 Parameter tuning

We selected ten training instances for tuning the three main parameters of heuristic GRASP+PR, corresponding to 36 different configuration settings: nine values for the quality parameter α of the restricted candidate list of the construction phase ($\alpha = 0.1, 0.2, \ldots, 0.9$); two local search strategies (deterministic or randomized); and two values for the size $n_{\mathcal{E}}$ of the elite set ($n_{\mathcal{E}} = 10, 20$).

We selected as the best version that with $\alpha = 0.8$, using the deterministic local search, and the size of the elite population $n_{\mathcal{E}} = 10$, for which the average gap from the best known value was the lowest.

5.2 Experiments on test instances

GRASP with path-relinking with the parameter settings determined in the previous settings was applied to the 20 remaining instances. Each instance was run 10 times with different seeds for 200 iterations each. Detailed results are presented in Table 1. For each instance in this table, we indicate, over the 10 runs, the average number of solutions in the elite set, the average number of iterations in which path-relinking was performed, the average and maximum running time of the heuristic in seconds, and the average and best solution value (the thirteen solution values marked in boldface are optimal values obtained by CPLEX). For all 20 test instances, the best known solution value was reached in at least one run. For each of the 17 smallest test instances, the best known solution value was obtained in all ten runs and, therefore, the average is equal to the best value.

We evaluated the performance of the restart strategy for PC-GMSTP for κ = 50, 100, 200. Table 2 illustrates typical computational results for instance 60PR299. Compared with the strategy without restarts, strategy restart(100) was the one that lead to the most significant reductions not only on the fourth quartile (long runs in the distribution tail), but also in the average running time.

Strategy	1st	2nd	3rd	4th	Average
No restarts	26.65	85.65	195.82	578.19	221.58
Restart(50)	26.65	**85.26**	**183.53**	498.57	198.50
Restart(100)	26.65	85.65	192.30	**476.72**	**195.33**
Restart(200)	26.65	85.65	195.82	512.57	205.17

Table 2: For instance 60PR299, 200 independent runs were executed for each strategy. Each run was made to stop when a solution as good as the target solution value 19998 was found. For each quartile, the table gives the average running times in seconds over all runs in that quartile. The average running times over the 200 runs are also given for each strategy.

	no restarts: 1000 iterations			restart(100): same time as no restarts		
Instance	Time (s)	Cost	Iteration	Time (s)	Cost	Iteration
60PR299	2488.25	19997	21	2490.43	**19995**	232
64LIN318	2617.43	**18140**	355	2618.39	**18140**	258

Table 3: Long runs: no restarts and restart(100), one run for each instance. Stopping criterion: 1000 iterations for no restarts, same running time as no restarts for restart(100).

Next, we performed long runs for two large instances, comparing in Table 3 the results for two variants of the heuristic. The strategy without restarts was made to stop after 1000 iterations, while restart(100) was made to stop after the same running time taken by the previous one. The best value for each instance is marked in boldface. Extensive numerical results will be presented in the full, forthcoming version of this paper.

References

[1] M. Fischetti, J.J. Salazar González, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, 45:378–394, 1997.

[2] B. Golden, S. Raghavan, and D. Stanojević. The prize-collecting generalized minimum spanning tree problem. *Journal of Heuristics*, 14:69–93, 2008.

[3] R.G. Marzo and C.C. Ribeiro. Test instances for the prize-collecting generalized MST problem, 2018. https://data.mendeley.com/datasets/vftc7kg6ty/1.

[4] P. C. Pop, O. Matei, C. P. Sitar, and D. Danciulescu. A genetic algorithm based solution approach to solve the prize-collecting generalized minimum spanning tree problem. In *Proceedings of International Conference on Computers and Industrial Engineering, CIE*, 2017.

[5] P.C. Pop. New Models of the Generalized Minimum Spanning Tree Problem. *Journal of Mathematical Modelling and Algorithms*, 3:153–166, 2004.

[6] P.C. Pop. On the prize-collecting generalized minimum spanning tree problem. *Annals of Operations Research*, 150:193–204, 2007.

[7] Mauricio G. C. Resende and Celso C. Ribeiro. Restart strategies for GRASP with path-relinking heuristics. *Optimization Letters*, 5:467–478, 2011.

[8] M.G.C. Resende and C.C. Ribeiro. *Optimization by GRASP: Greedy randomized adaptive search procedures*. Springer, New York, 2016.

[9] D. Stanojevic, B. Golden, and S. Raghavan. The Prize-Collecting Generalized Minimum Spanning Tree Problem. In *7th INFORMS Telecommunications Conference*, Boca Raton, 2004.

Efficient Algorithm for Packet Routing Problems Using Transmission History Information

Akinori Yoshida[1] , Masaya Kaneko[1] , Toshichika Aoki[1] , Takayuki Kimura[2], and Tohru Ikeguchi[3,4]

[1] Graduate School of Electronics, Information and Media Engineering, Faculty of Engineering, Nippon Institute of Technology
[2] Department of Electrical, Electronics and Communication Engineering, Faculty of Fundamental Engineering, Nippon Institute of Technology
4–1 Gakuendai, Miyashiro, Minami-Saitama, Saitama, 345-8501, Japan
[3] Department of Management Science, Graduate School of Engineering, Tokyo University of Science
[4] Department of Information and Computer Technology, Faculty of Engineering, Tokyo University of Science
6-3-1 Niijuku, Katsushika, Tokyo 125-8585, Japan
2198022@stu.nit.ac.jp, 2178007@estu.nit.ac.jp, 2188001@sstu.nit.ac.jp, tkimura@nit.ac.jp, tohru@rs.tus.ac.jp

Abstract

The simplest shortest path problem in a network whose weights of edges are static can be solved easily using the well-known Dijkstra algorithm. However, in practical cases such as packet communication networks, the weights of edges fluctuate by flowing packets or queuing packets at nodes (for example, routers). Obtaining the shortest paths of the packets from their sources to destinations in communication networks is called a packet routing problem. In the packet routing problems, both the optimum routes of packets at the current time are that at the next time cannot be guaranteed because the distributions of packets changes continually with time. We herein report the performance of routing algorithms using transmission history information for packet routing problems. Numerical experiments demonstrate that our method improves the average arrival rate of packets by approximately 50% over conventional routing methods for different topologies of communication networks.

1 Introduction

The method to establish the shortest path between two nodes according to any condition of networks is a well-known and important problem in science and engineering. The simplest case of the shortest path problems of a network whose weights of links are static can be solved easily using the Dijkstra algorithm [3]. However, in practical cases such as communication networks, the weights of edges are dynamically fluctuated because of the distributions of queuing packets at nodes (for example, routers) change continually with time. If few packets are flowing in a communication network, the weights of the links are almost static; subsequently, the shortest paths for the packets are found easily. However, if the flowing packets increase, the weights of the links change dynamically; thus, it is difficult to obtain the shortest paths. Transmitting the packets to their destinations as quickly as possible by avoiding congestion can be achieved by obtaining the effective paths of the packets according to the spatio–temporal distribution of the packets using sophisticated methods. We term obtaining the shortest paths from sources and destinations of packets in a communication network a packet routing problem.

To solve packet routing problems, a routing algorithm using transmission history information was proposed [5], and the method was evaluated using scale-free communication networks, i.e., networks in which most of the nodes have small numbers of connections, but some nodes are hub nodes with large numbers of connections. However, recent studies revealed that only 4% of communication networks are scale free [2]. In addition, evaluations of our method for different network structures are of interest. Hence, we investigate the performance of the routing method using the transmission history information for different topologies of communication networks.

The basic elements of a model of communication network are nodes, edges (links), and packets. All the packets contain randomly assigned sources and destinations, and are transmitted to destinations through any of the links according to the first-in first-out principle. The objective function of the packet routing problem is defined as follows:

$$Q_i^* = \min_{j \in m_k} q_{ij} \ (i = 1, \ldots, G), \tag{1}$$

where G is the total generating packets; m_k is a set of nodes in the kth transmitting route; q_{ij} is the total transmitting time of the generating packets i using route j.

Further, the shortest paths of the packets can be obtained by selecting the optimum adjacent transmitting node at every node according to the spatio–temporal distributions of packets in the communication networks.

2 Realization of the routing method using transmission history information

To realize a routing method using transmission history information (abbr. memory) [5], we define an evaluating function that determines the transmission of a packet from node i to its adjacent node j as follows:

$$\xi_{ij}(t) = \frac{d_{ij} + d_{jg(p_i(t))}}{\sum_{l \in N_i} (d_{il} + d_{lg(p_i(t))})}, \tag{2}$$

$$\zeta_{ij}(t) = \alpha \sum_{s=1}^{t} k_r^s x_{ij}(t-s), \tag{3}$$

$$x_{ij}(t) = \begin{cases} 1 & (\min\left(\xi_{ij}(t) + \zeta_{ij}(t)\right)), \ j \in N_i, \\ 0 & (\text{otherwise}), \end{cases} \tag{4}$$

where d_{ij} is the shortest distance between node i and its adjacent node j; N_i is a set of adjacent nodes of node i; $g(p_i(t))$ is the destination node of the transmitted packet of node i at the tth time; $\alpha > 0$ is a control parameter that determines the strength of the memory information; $0 < k_r < 1$ is a decay parameter of the memory information; $x_{ij}(t)$ is an indicator function that memorizes the packet transmission from node i to node j at the tth time. If $\xi_{ij}(t) + \zeta_{ij}(t)$ contains the smallest value of the other nodes, node i transmits a packet to the adjacent node j. If node i frequently transmits the packets to the adjacent node j, ζ_{ij} increases. Consequently, node i avoids transmitting the next packet to the adjacent node j. We expect this diversification of transmitting routes using the memory information to be applicable to obtain the shortest paths of packets while avoiding packet congestion.

3 Numerical experiments

In this study, we used four types of different communication network topologies with 300 nodes: scale free [1], scale-free with cluster relationships [4], and ring-topology regular and small-world [6] communication network models. The small-world networks [6] have short distances between nodes and strong cluster relationships. In our communication network models, the transmission capacity of node i is defined as $C_i = 1 + \lfloor \gamma K_i + 0.5 \rfloor$, where K_i is the degree of node i and $\gamma > 0$ is a tunable parameter. The transmission capacity corresponds to the maximum number of packet transmission in one iteration. Further, every node has an infinite size of buffer. We repeated packet transmission at every node selection of an adjacent transmitting node and the transmission of a packet, for 10^3. In this study, the packets were removed from the network if they arrived at the destinations.

The parameters of our method were set as follows: $\alpha = 0.01$, $k_r = 0.99$, and $\gamma = 0.4$. We generated R packets whose sources and destinations were randomly determined at each iteration. We conducted numerical experiments for 20 times and calculated the average of the results.

The results are shown in Fig. 1.

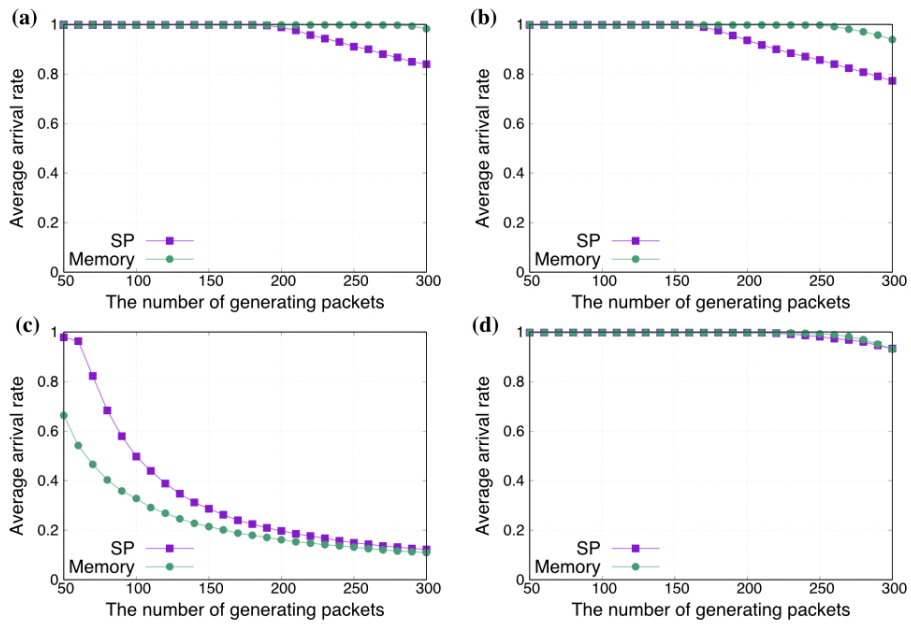


Figure 1: Relationship between the number of generated packets (R) and average arrival rate of packets for (a) scale free [1], (b) scale-free with cluster relationships [4], (c) ring-topology regular [6], and (d) small-world [6] communication network models.

As shown in Fig. 1, the memory method successfully obtained a higher average arrival rate of packets than the shortest path (SP) method for (a) scale-free, (b) scale-free with cluster relationships [4], (d) small-world [6] network models. Because ring-topology communication networks rarely exist in autonomous system level communication networks, the routing method using transmission history information demonstrates effective performance for various topologies of communication networks, and effectively solves the packet routing problem. Further, we hope our proposed method will be successfully applied to OpenFlow protocols.

The research of T. K. was supported by JSPS KAKENHI Grant Number 16K21327. The research of T. I. was supported by JSPS KAKENHI Grant Numbers 15TK0112 and 17K00348.

References

[1] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[2] A. D. Broido and A. Clauset. Scale-free networks are rare. arXiv:1801.03400, 2018.

[3] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerical Mathmatics*, 1:269–271, 1959.

[4] P. Holme and B. J. Kim. Growing scale-free networks with tunable clustering. *Physical Review E*, 65(2), 1998.

[5] T. Kimura. Congestion avoidance on networks using independent memory information. *Lecture notes in computer science, Springer*, 2019 (in press).

[6] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.