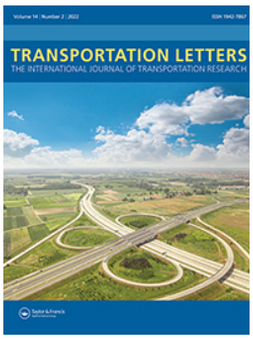

Taylor & Francis
Taylor & Francis Group

Transportation Letters

The International Journal of Transportation Research

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ytrl20

Variable fixing heuristics for the capacitated multicommodity network flow problem with multiple transport lines, a heterogeneous fleet and time windows

Lucas Rebouças Guimarães, Jorge Pinho de Sousa & Bruno de Athayde Prata

To cite this article: Lucas Rebouças Guimarães, Jorge Pinho de Sousa & Bruno de Athayde Prata (2022) Variable fixing heuristics for the capacitated multicommodity network flow problem with multiple transport lines, a heterogeneous fleet and time windows, Transportation Letters, 14:2, 84-93, DOI: 10.1080/19427867.2020.1815143

To link to this article: https://doi.org/10.1080/19427867.2020.1815143

TRANSPORTATION LETTERS
2022, VOL. 14, NO. 2, 84–93
https://doi.org/10.1080/19427867.2020.1815143

Taylor & Francis
Taylor & Francis Group

Variable fixing heuristics for the capacitated multicommodity network flow problem with multiple transport lines, a heterogeneous fleet and time windows

Lucas Rebouças Guimarães[a], Jorge Pinho de Sousa[b] and Bruno de Athayde Prata[c]

[a]Department of Industrial Chemistry, Federal Institute of Education, Science, and Technology of Espírito Santo, Vilha Velha, Brazil; [b]Faculty of Engineering, University of Porto/INESC TEC, Porto, Portugal; [c]Department of Industrial Chemistry, Federal University of Ceará, Vilha Velha, Brazil

ABSTRACT

In this paper, we investigate a new variant of the multi-commodity network flow problem, taking into consideration multiple transport lines and time windows. This variant arises in a city logistics environment, more specifically in a long-haul passenger transport system that is also used to transport urban freight. We propose two mixed integer programming models for two objective functions: minimization of network operational costs and minimization of travel times. Since the problems under study are NP-hard, we propose three size reduction heuristics. In order to assess the performance of the proposed algorithms, we carried out computational experiments on a set of synthetic problem instances. We use the relative percentage deviation as performance criterion. For the cost objective function, a LP-and-Fix algorithm outperforms other methods in most tested instances, but for the travel time, a hybrid method (size reduction with LP-and-Fix algorithm) is, in general, better than other approaches.

KEYWORDS

Combinatorial optimization; network design problems; mixed integer linear programming; city logistics; size reduction algorithms

Introduction

In general terms, network flow problems aim to enhance the operation of logistic systems, by moving some objects from a set of origins to a set of destinations, subject to constraints such as warehouse and vehicle capacities, time windows, etc.

In particular, multi-commodity flow problems consist in transporting goods, from suppliers to customers, aiming to optimize some network attributes. These problems have applications in various types of systems, such as in communication, urban traffic, railways, multiproduct production-distribution systems, or military logistics (see e.g. Wang 2018).

Many variants of these models can be found in the literature, such as linear and non-linear models, capacitated and uncapacitated networks, or fixed-charge network flows. The practical importance of these problems is clearly demonstrated by early surveys, such as Kennington (1978), Assad (1978) or Ouorou, Mahey, and Vial (2000). More recently, Wang (2018) presented a literature review of multi-commodity network flows problems covering applications and solution methods.

Some initial research has shown that the passenger vehicles baggage compartments generally have significant empty spaces. Therefore, the transport efficiency can increase by filling these spaces with loads that have the same destination of the vehicle. This feature will be the basic principle for the design of the solutions proposed in this work. Moreover, the passenger's services considered here are intercity passengers transport.

We expect that the system developed in this work will directly contribute to a decrease of the transport operations costs of some stakeholders (shippers and carriers) and, indirectly, to a traffic reduction in intercity highways and inside cities, with positive environmental and social impacts. This new transport system will be designed around the following elements:

- bus stations (that will provide small storage areas for goods);
- passenger vehicles (that will transport freight between origin and destination bus stations);
- suppliers (enterprises that will use the transport services and provide storage areas for their products);
- and customers (enterprises that will pick up goods from the destination bus stations).

The approach developed in this work is based on a new multi-commodity network flow model with time windows, multiple transport lines, and a heterogeneous fleet (MCFPTWHF), to transport freight in long-haul passenger vehicles (buses). This approach will contribute to increase transport efficiency and reduce costs, while decreasing the environmental and social impacts resulting from transport traffic. These results will be mainly achieved by the reduction of vehicles circulation around and between cities.

Another contribution for increasing transport efficiency is the maximization of the use of vehicles, since they are already going from an origin point to a destination point, carrying passengers. In this way, allocating freight to these vehicles can contribute to dilute the operating cost of travels. Therefore, this work contributes to the Operations Research field by developing a new problem idea and a methodology to support its resolution (mixed integer linear programming). Finally, the developed heuristic techniques are expected to find satisfactory solutions to the problem in low, acceptable computational times. According to our thorough literature review, the problem variant addressed in our work has not been reported yet in the literature, despite its theoretical and practical importance.

In order to optimize the system performance, the problem will be modeled in a way similar to one well known mixed integer model from the literature, the *multi-commodity network flow problem* (Wang, 2018).

The capacitated multicommodity network flow problem with multiple transport lines, a heterogeneous fleet and time windows is a NP-hard problem, since it is a generalization of the classical

CONTACT Bruno de Athayde Prata baprata@ufc.br Department of Industrial Engineering, Campus do Picí, Bloco 724, Fortaleza, Ceará, Brazil. CEP 60455-760

capacitated multicommodity network flow problem, where producing an integer flow satisfying all demands is NP-hard (Even et al. 1976). This obviously justifies the development of heuristic techniques as part of this work.

The main contributions of this paper are, first, two mixed integer programming (MIP) formulations for the problem under study and, second, three MIP-based heuristics for finding near-optimal solutions with an acceptable computational effort. Moreover, a comprehensive set of computational tests was performed to evaluate the robustness of the proposed approaches.

The remainder of this paper is organized as follows: in Section 2, a literature review is presented; in Section 3, the multicommodity network flow problem treated in this paper is described; in Section 4, the heuristic algorithms proposed to solve the problem are presented; in Section 5, computational results are discussed; and, finally, in Section 6, some conclusions are drawn and suggestions for future work presented.

Literature review

Several approaches have been developed to deal with the multicommodity network design problem and its variants. In this section, a brief literature review on this problem is presented, and the associated solution techniques are briefly described. The references are presented in chronological order.

Kirby, Hager, and Wong (1986) presented an approach that deals with natural resource and transportation network investments, represented by two models for multiple commodities and periods: a transshipment model with fixed-charge arcs and a land allocation model. To solve this problem, the authors use two different approaches. When the size of the mixed-integer program is relatively small, they use exact methods with state-of-the-art optimization software packages. For large instances, a heuristic procedure based on solving an LP relaxation of the problem with successive improvement attempts was developed.

Helme (1992) developed a computer-based decision tool to deal with the reduction of air traffic delay in a space-time network. The problem consists in evaluating the impact of airway capacities upon traffic, from multiple origins to multiple destinations. The problem was modeled as a multicommodity minimum cost flow problem, over a network in space-time.

Farvolden, Powell, and Lustig (1993) developed an approach based on both primal partitioning and decomposition techniques to solve the multicommodity network flow problem. This approach involves simplifying the computations normally required by the simplex method.

Barnhart and Sheffi (1993) developed a network-based primal-dual heuristic solution approach for large-scale multicommodity network flow problems. The authors found out that primal-dual and price directive algorithms (exact solution strategies) are unable to achieve even an initial solution for this problem, due to excessive memory requirements. However, network-based heuristics can find optimal solutions. Barnhart et al. (1994) present a partitioning solution procedure for large-scale multicommodity flow problems, using a cycle-based problem formulation, and column generation techniques to solve a series of reduced-size linear programs in which a large number of constraints are relaxed.

Farvolden and Powell (1994) present a local-improvement heuristic for a service network design problem. The scheduled set of vehicles departures are modeled as a multi-commodity network flow problem. In this way, the heuristics are structured in two steps: one for dropping a scheduled service, and another for introducing a new service. Both are based on subgradients derived from the optimal dual variables, by the shipment routing subproblem.

Cruz, Smith, and Mateus (1998) developed a branch-and-bound algorithm to solve optimally the uncapacitated fixed-charge network flow problem. Frangioni and Gallo (1999) describe a cost decomposition approach for the linear multicommodity min-cost flow problem, where the mutual capacity constraints are dualized, and the resulting Lagrangian dual is solved with a dual-ascent algorithm, belonging to the class of bundle methods. Gabrel, Knippel, and Minoux (1999) developed an exact solution procedure for the multicommodity network design problem with general discontinuous step-increasing cost functions, including single-facility and multiple facilities capacitated network loading problems. The procedure is a specialization of the general Benders decomposition procedure.

Crainic, Gendreau, and Farvolden (2000) present an efficient procedure to solve the fixed-charge capacitated multicommodity network design problem, using a tabu search framework that explores the space of the continuous flow variables, by combining pivot moves with column generation, while evaluating a mixed integer objective. In this way, they determine tight upper bounds on the optimal solution of realistic size problem instances.

Barnhart et al. (2002) developed an approach to solve a particular service network design, the express shipment delivery problem. They developed models and a solution technique designed specifically for large-scale problems with time windows. The solution approach consists of dividing the service network into two subproblems: route generation and shipment movements. The first problem is solved using a branch-and-price-and-cut algorithm. The shipment movement subproblem is a large-scale integer multicommodity network flow model, with side constraints, and is solved using a branch-and-price algorithm to find the network shortest paths.

Holmberg and Yuan (2003) present an efficient column generation approach for solving the multicommodity network design problem with side constraints on paths. In this approach, the solutions are built up successively by path generation, and the objective is to find the set of shortest paths. Alvarez, González-Velarde, and de-Alba (2005) developed a grasp embedded scatter search, to solve the multicommodity capacitated fixed charge network design problem. Topaloglu and Powell (2006) present a stochastic and time-dependent version of the minimal cost multicommodity flow problem. The authors propose an interactive, adaptive dynamic-programming-based methodology, making use of linear and nonlinear approximations of the value function.

Lim and Smith (2007) deal with a network interdiction problem as a multicommodity flow network. According to the authors, an attacker disables a set of network arcs to minimize the maximum profit that can be obtained from shipping commodities across the network. The interdiction can be discrete (concerning the choice of each arc) or continuous (related to the capacities of the arcs). To deal with the discrete problem, a linearized model for the optimized network was developed and compared to a penalty model that does not require linearization constraints. In the continuous case, a heuristic algorithm for optimal partitioning was developed, used to estimate the objective function value.

Oimoen (2009) presents three contributions for solving highly dynamic capacitated multicommodity network flow problems. First, he develops an ant colony algorithm to solve the static problem with weak constraints. Then, another algorithm is used to adjust the exploration parameter of the solution space dynamically. Finally, a distributed approach is proposed, replacing the previously centralized solver.

Aloise and Ribeiro (2011) present heuristics based on the shortest path and maximum flow algorithms, combined with adaptive

memory, to obtain an approximate solution to the multicommodity network design problem in the framework of a multi-start algorithm. Guardia and Lima (2010) designed a numerical implementation of a primal-dual interior-point method to solve the linear multicommodity network flow problem. In this approach, at each iteration, the corresponding linear problem, expressed as an augmented indefinite system, is solved by using the AINV algorithm, combined with an indefinite preconditioned conjugate gradient algorithm.

Kim, Jun, and Kim (2011) considered the fixed-charge capacitated multicommodity network design problem with turn penalties, and proposed a mixed integer programming model and a two-phase heuristic algorithm to solve the problem. The approach consisted in building an initial flow path network by a constructive method, that was then improved with an iterative approach.

Kleeman et al. (2012) developed a multiobjective evolutionary algorithm to solve the multicommodity capacitated network design problem. According to the authors, this variation represents a hybrid communication network with multiple objectives including costs, robustness, vulnerability, delays, and reliability. They used a modified and extended nondominated sorting genetic algorithm with a novel initialization procedure and mutation method, to deal with the highly constrained problem features. The obtained results were good, and the algorithm seems to perform very efficiently.

Abdulhakim (2013) presented heuristic algorithms for multicommodity flow problems in computer networks engineering, aiming to minimize shipping costs. Malairajan et al. (2013) worked with the multicommodity network flow problem and with a variant referred to as the bi-objective resource allocation problem, with bound and varying capacity. To solve both problems, the authors developed a recursive function inherent genetic algorithm, thus achieving good quality solutions. Tadayon and Smith (2014) developed two linearization techniques to solve the min-cost multicommodity network flow problem, proposing an integer programming approach and cutting plane techniques. Lagos et al. (2014) developed a hybrid metaheuristic approach combining tabu search and genetic algorithms, to deal with a capacitated multicommodity network flow problem. While the tabu search acts as the main algorithm, the genetic algorithm is used to select the best option among the neighbors of the current solution.

Masri, Krichen, and Guitouni (2015) presented a multi-start variable neighborhood search to solve a capacitated single path multicommodity flow problem. Wei et al. (2014) worked on the multi-source single-path multicommodity network flow problem and developed a simulated annealing metaheuristic, to minimize the network total transportation cost. Lin and Kwan (2017) presented a class of multicommodity flow problems with commodity compatibility relations among commodities used at each network node. To solve this problem, the authors use a node-family branching technique, that removes incompatible commodities at nodes, by a branch-and-bound algorithm.

Table 1 shows an overview of the multi-commodity network flow problem approaches briefly presented in this section, their variants, the solution methods (classified as exact or heuristic approaches), objective function type (minimization or maximization), arc type (capacitated or uncapacitated) and finally, the problem features (fixed charge costs and time windows).

The majority of the surveyed works addresses the multicommodity network flow problem and its variations by heuristics techniques. Probably this is due to the complexity of the problem when large, real-size instances are considered. We can also notice that the multicommodity network flow problem has many variations and it is used to help modeling many real problems. This is due to the fact that the model is very flexible and easily adaptable to different real situations.

In a large majority of these studies, the objective function is a cost minimization, instead of a maximization of the flows in the

Table 1. An overview of the multi-commodity network flow problem approaches.

	Solution Approach		Objective		Arc Type		Problem Features	
Reference	Exact Method	Heuristic Method	Min	Max	Capacitated	Uncapacitated	Fixed Charge	Time Windows
Kirby, Hager, and Wong (1986)		X	X		X		X	
Helme (1992)***	X		X		X			X
Farvolden, Powell, and Lustig (1993)	X		X		**	**		
Barnhart and Sheffi (1993)		X	X	X	**	**		
Barnhart et al. (1994)	X		X		**	**		
Farvolden and Powell (1994)		X	X		**	**		
Cruz, Smith, and Mateus (1998)	X		X			X	X	
Frangioni and Gallo (1999)	X		X			X		
Gabrel, Knippel, and Minoux (1999)	X		X		X			
Crainic, Gendreau, and Farvolden (2000)		X	X		X		X	
Barnhart et al. (2002)	X		X		**	**	X	X
Holmberg and Yuan (2003)	X		X		X			
Alvarez, González-Velarde, and de-Alba 2005		X	X		X			
Topaloglu and Powell (2006)****		X		X	**	**		
Lim and Smith (2007)	X	X	X	X	**	**		
Oimoen (2009)		X	X		X			
Aloise and Ribeiro (2011)		X	X		**	**		
Guardia and Lima (2010)	X		X		X			
Kim, Jun, and Kim (2011)		X	X		X		X	X
Kleeman et al. (2012)		X	X		X		X	X
Abdulhakim (2013)		X	X	X	**	**		
Malairajan et al. (2013)		X	X		X			
Tadayon and Smith (2014)	X		X			X	X	
Lagos et al. (2014)		X	X		X		X	
Masri, Krichen, and Guitouni (2015)		X	X		X			
Wei et al. (2014)		X	X		X		X	
Lin and Kwan (2017)	X		X		**	**		
TOTAL	**12**	**16**	**27**	**4**	**14**	**3**	**10**	**4**

** *This information is not available in the paper.*

arcs. We can also notice that in a majority of the cases, capacitated arcs were used instead of uncapacitated arcs, showing the need for approaches that are closer to reality.

Finally, it should be noted that from the 28 studies reported in this literature review, only 10 consider fixed charge costs, and only 4 deal with time windows. It seems therefore that these two features of multicommodity design flow problems have still been explored in in a quite limited way.

Problem statement

In this problem, more specifically, we will decide how freights are going to be transported from one point in the network (origin) to another point (destination). Hence, for each cargo, we must be able to choose one specific path from all possible paths, and determine the goods associated with each flow.

The main purpose of our work is to develop a framework for managing urban freight flows in a network, minimizing total costs and time, and satisfying all the demand. The goods considered for transportation in this context are products with no risk characteristics. Using an already existing (bus) network (with their associated transportation lines) is surely an interesting alternative, as transportation costs can be significantly decreased, since vehicles are already going to the specific required destinations, transporting passengers, independently of having or not having freight allocated.

As in practice transport companies operate individually, to make this idea a realistic approach, it is necessary to consider a 'broker' that will mediate the relationship between the bus companies and the suppliers/demanders of goods to be transported. The role of this broker can be operationalized using the algorithm approach proposed in this work.

Mathematical model

In general terms, the problem can be represented by the following mathematical models, that can be viewed as a mono-objective MCPTWHF (multicommodity network flow model with time windows, multiple transport lines, and a heterogeneous fleet). The first model (Equations (1), (3)–(14)) minimizes costs; while the second model (Equations (2), (3)–(14)) minimizes travel times – the two models have the same set of constraints.

In this approach, the extensions of the original model are based on considering a graph $G = (E, V)$, that accommodates the following problem features. Each vertex in V is a physical node (that can be viewed as an origin or a destination) and has an offer/demand freight value for each kind of product.

Each edge e in E is associated to a set of indices (*tovqkij)* – from origin i to destination j (the 'physical' arc), the flow starts on line t, vehicle v, and can continue in line o, vehicle q. Each edge in E has: a variable cost related to the specific transported product amount; a lower bound and an upper bound for volume per freight and per product; a departure time, a travel duration and an arrival time, for the vehicle from the transport line.

Most of the details of the problem are reflected in the developed data structures, and are briefly described here to facilitate the understanding of the model parameters. The notation adopted in this work is presented in Table 2.

Using this notation, the two variants of the MCPTWHF model can be formulated as follows:

Model 1:

$$min \sum_{(t,o,v,q,k,i,j)\in E} c_{tovqkij} y_{tovqkij} \tag{1}$$

Table 2. Parameters used in the models.

Notation	Description
c_e	unit variable cost in edge e E (i.e. from origin i to destination j, starting on line t, vehicle v, and continuing in line o, vehicle q)
l_e	lower bound of the flow in edge e
u_e	upper bound of the flow in edge e
p_e	departure time for edge e
h_e	duration time for edge e
d_e	arrival time for edge e
a_{oqij}	the capacity of arc (i,j) and transport line o, vehicle q. (these values are the total load capacities for each vehicle, from the different transport lines).
b_{ik}	demand from node i by product k.
y_e	flow in edge e E (i.e. from origin i to destination j, starting on line t, vehicle v, and continuing in line o, vehicle q)
x_e	binary decision variable that edge e is active/used (=1) or not (=0)
w_e	total displacement time on edge e

Model 2:

$$min \sum_{(t,o,v,q,k,i,j)\in E} w_{tovqkij} \tag{2}$$

subject to:

$$\sum_{j:(t,o,v,q,k,i,j)\in E} y_{tovqkij} - \sum_{j:(t,o,v,q,k,i,j)\in E} y_{tovqkji} = b_{ik}, \quad i,k \in V \tag{3}$$

$$\sum_{j:(t,o,v,q,k,i,j)\in E} y_{tovqkij} \le a_{oqij}, \quad (o,q,i,j) \in E \tag{4}$$

$$l_{tovqkij} \le y_{tovqkij} \le u_{tovqkij}, \quad (t,o,v,q,k,i,j) \in E \tag{5}$$

$$y_{tovqkij} \le \left(\sum_{(i,k)\in V, b>0} b_{ik} \right) x_{tovqkij}, \quad (t,o,v,q,k,i,j) \in E \tag{6}$$

$$x_{tovqkij} \le y_{tovqkij}, \quad (t,o,v,q,k,i,j) \in E \tag{7}$$

$$x_{tovqkij} \times d_{tovqkij} \le w_{tovqkij}, \quad (t,o,v,q,k,i,j) \in E \tag{8}$$

$$y_{tovqkij} \ge 0, \quad (t,o,v,q,k,i,j) \in E \tag{9}$$

$$w_{tovqkij} \ge 0, \quad (t,o,v,q,k,i,j) \in E \tag{10}$$

$$x_{tovqkij} \in \{0,1\}, \quad (t,o,v,q,k,i,j) \in E \tag{11}$$

Pre-processing conditions:

$$w_{tovqkij} = p_{tovqkij} + h_{tovqkij}, \quad (t,o,v,q,k,i,j) \in E \tag{12}$$

$$a_{oqij} < \sum_{(t,o,v,q,k,i,j)\in E} \left(u_{tovqkij} - l_{tovqkij} \right), \quad (o,q,i,j) \in E \tag{13}$$

$$\sum_{(i,k)\in V} b_{ik} = 0 \tag{14}$$

Objective function (1) aims at minimizing the total transport costs of the network, thus reducing the distance covered by the cargo, since the cost is proportional to the distance. Objective function (2) minimizes the total travel time.

Constraint set (3) guarantees the flow conservation on the vertices, preventing the moved items from standing in the vertices of

the transport network. Constraint set (4) guarantees that the flow in the arc does not exceed its total capacity. Constraint set (5) defines the upper and lower bounds on the arc flows. These limits represent the maximum (u_e) and minimum (l_e) quantities of cargo (k) that can be moved in the arc (i,j) by a specific vehicle on a transport line. Constraint set (6) guarantees the inexistence of flow in arcs where vehicle time windows are incompatible, since the binary variable x_e is associated with time compatibility, (see Equations (8) and (11)) while the constraint set (7) assures that the binary variable x_e will take the value '1' just when the arc is being used (i.e. non-zero y_e variables). Constraint sets (8) assures the time compatibility between vehicles from different transport lines.

The final three sets of equations are pre-processing conditions. Equations (12) define the time spent in the arcs. Equations (13) consider the vehicle's capacities for each arc, and Equations (14) guarantee that the total offer will be sufficient to satisfy all the demand.

Proposed algorithms

The next sections describe the heuristics developed in this work to solve the MCPTWHF, namely the size-reduction (SR) heuristic, the "LP-and-fix (LPF) heuristic, and a hybrid algorithm (HA) that combines the two previous heuristic approaches. The first step of both heuristics is to read the data structured in the three matrices *edges, capacities*, and *vertices*.

Size-reduction heuristic

According to Fanjul-Peyro and Ruiz (2011), a reduction heuristic explores the topology of the problem and reduces some problem attributes, without making its solution infeasible. These authors obtained good results applying a set of metaheuristics based on a size-reduction of the original assignment, in the unrelated parallel machines scheduling problem. The method produced solutions of very good quality, in a short computational time.

The reduction heuristic described here has a very simple operating mechanism. As presented in section 3, the number of possible *edges* significantly increases the complexity of the data structure. Therefore, the size reduction heuristic will reduce the number of *edges*, as a way to reduce the number of available options to move the goods along the transport network. Figure 1 presents a flowchart with the size-reduction heuristic procedure.

This procedure can worsen the value of the solution depending on the reduction size (given by a pre-defined *reduction percentage*), but it is expected that it also reduces computational times, this being the purpose of the procedure.

Therefore, the developed approach makes reductions analyzing the pairs of origin (i)/destination (j) nodes that are repeated throughout the data structure. In this way, the procedure does not impair the functioning of the network. The number of nodes analyzed by the algorithm will depend on the given reduction percentage.

The reduction criteria are based on first analyzing the cost and then the time. The algorithm traverses all the edges of the network and creates a list containing the edges with repeated pairs (i, j) of origin and destination nodes, so that the edges in the list are ordered decreasingly, in terms of costs and time. The reductions are made using the given reduction percentage, starting with the edges in the top of the list.

It should be noted that, to be efficient, the edge reductions in the heuristic procedure need to be made in a rather smart way, as the multi-transport line with transshipment possibilities increases a lot the complexity of the problem. This is due to the increase in the number of choices to move goods between the origin and destination nodes, as each edge is a new possibility to make the freight between the same nodes, but using different transport lines and vehicles, with or without transshipment.

For instance, we can have various ways to move the same cargo from a given node i to another node j, e.g., using different transport lines with different vehicles, costs, capacities, time windows, and possibly making a transshipment between two transport lines. I.e., each new edge represents a new possibility for this choice.

Therefore, the size reduction heuristic will list all the origin and destination nodes that are repeated over the network (creating a list of repetitions that stores the edges positions in the matrix *edges*), and eliminating from this matrix, those that have the largest costs and times. Thus, the chances of making the network operation unfeasible through reductions of its edges decreases. In this way, the procedure allows larger reduction percentages without making the problem unfeasible.

At this time, it should be noted that in the *repetitions* list, we are only considering the repetition of the origin and the destination nodes inside the *edges* matrix (which has 13 indices), all other elements of the various listed edges being different themselves. Figure 2 presents the pseudocode of the proposed size-reduction algorithm.

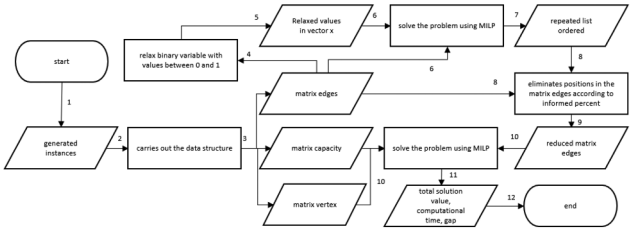

Figure 1. Size-reduction heuristic procedure.

```
begin
INPUT generated_instance;
        reading_data()        / /carries out the data structure
        INPUT generated_instance;
            repeat
                read each element of data structure edges and store in the matrix_edges[ ][ ];
                read each element of data structure capacity and store in the matrix_capacities[ ][ ];
                read each element of data structure vertex and store in the matrix_vertices[ ][ ];
            until all the data has been read;
        OUTPUT matrix_edges[ ][ ], matrix_capacities[ ][ ], matrix_vertices[ ][ ];
    repeated_edges()      //creates a list with repeated edges
        INPUT matrix_edges[ ][ ];
            repeat
                search repeated pair of origin (i) and destination (j) nodes in matrix_edges[ ];
                creates the vector repeated_list[ ];
                stores the repeated nodes position in repeated_list[ ];
            until all the repeated nodes position be included in repeated_list [];
         OUTPUT repeated list[ ];
    ordering_positions();      // ordering decreasingly repeated_list in terms of cost and time
        INPUT repeated_list[ ], matrix_edges[ ][ ];
            read each element of repeated_list[ ];
            repeat
                order decreasingly each element of the repeated_list[ ] according with correspondent cost (c) in
matrix_edges[ ][ ];
                order decreasingly each element of the repeated_list[ ] according with correspondent time (p) + (h) in
matrix_edges[ ][ ];
            until all the repeated_list[ ] values has been ordered in terms of cost and time;
        OUTPUT repeated_list[ ]; //ordered
    reduction_positions()      //eliminates positions in the matrix_edges[ ][ ] according to informed percent
        INPUT matrix_edges[ ][ ], repeated_list[ ], reduction_percent;
            repeat
                eliminates the position in the top of restricted_list[ ] in matrix_edges[ ][ ];
            until performs the reduction percent;
        OUTPUT matrix_edges[ ][ ];   //reduced
    concert_cplex_solve()
        INPUT matrix_edges[ ][ ], matrix_capacity[ ][ ], matrix_vertex[ ][ ];
            solve the problem using cplex optimization environment through concert technology;
        OUTPUT total_solution_value, computational_time, solution_gap;
end
```

Figure 2. Size-reduction heuristic.

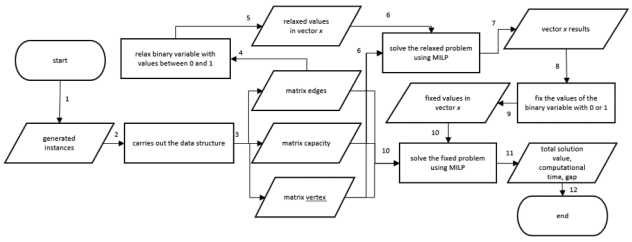

Figure 3. LP-and-fix heuristic procedure.

LP-and-fix heuristic

According to Maes, Mcclain, and Van Wassenhove (1991) and Toledo et al. (2015), heuristics based on mathematical programming are, in general, an efficient way to solve mixed integer problems. As the above procedure, the relaxation and fixation heuristic described in this section has a very simple operating mechanism. It consists in relaxing some integer or binary decision variables, (easily) solving the relaxed model, then fixing the values of those variables according to some rule or criterion and finally, solving the resulting problem. Figure 3 presents a flowchart of our LP-and-fix heuristic procedure.

In our context, the LP and fix heuristic was implemented relaxing the value of the binary decision variables x (that indicate the presence of a flow in a given edge), letting those variables have values between 0 and 1, and then solving the resulting relaxation using the CPLEX optimization environment. Then, the algorithm fixes the values of variables with a 0 value to 0, and those with a 1 value to 1, and then solves the resulting problem, using the same optimization tool.

This relaxed problem is easily solved, as it has more freedom in terms of allocation levels between the network edges. Then, by fixing the variables to 0 or 1, we significantly reduce the search procedure, making the optimization easier.

The two other sets of decision variables, y (the amount of goods moved in each edge) and w (the amount of time spent by each edge) are already continuous variables. Figure 4 presents the pseudocode of the proposed LP-and-fix algorithm.

```
begin
INPUT generated_instance;
        reading_data()        / /carries out the data structure
        INPUT generated_instance;
            repeat
                read each element of data structure edges and store in the matrix_edges[ ][ ];
                read each element of data structure capacity and store in the matrix_capacities[ ][ ];
                read each element of data structure vertex and store in the matrix_vertices[ ][ ];
            until all the data has been read;
        OUTPUT matrix_edges[ ][ ], matrix_capacities[ ][ ], matrix_vertices[ ][ ];
    relaxing _x[ ]()            //relax binary variable with values between 0 and 1
        INPUT matrix_edges[ ][ ];
            creates the vector x[ ] with the same number of matrix_edges[ ][ ] positions;
            repeat
                relax all the positions of the vector x[ ] with values from 0 until 1;
            until all the positions have been relaxed;
        OUTPUT x[ ];    //relaxed values
    concert_cplex_solve()     //solve relaxed problem
        INPUT x[ ],matrix_edges[ ][ ], matrix_capacities[ ][ ], matrix_vertices[ ][ ];
            solve the problem using cplex;
        OUTPUT total_solution_value, computational_time, solution_gap, x[ ];
    fixing _x[ ]()   //fix the values of the binary variable with 0 or 1
        INPUT x[ ];
            repeat
                read the values of each position of vector x[ ];
                fix the position of the vector x[ ] with values 0, as 0 and 1, as 1;
            until all the positions have been fixed;
        OUTPUT x[ ];    //fixed values
    concert_cplex_solve()
        INPUT x[ ],matrix_edges[ ][ ], matrix_capacity[ ][ ], matrix_vertex[ ][ ];
            solve the problem using cplex;
        OUTPUT total_solution_value, computational_time, solution_gap;
end
```

Figure 4. LP-and-fix heuristic.

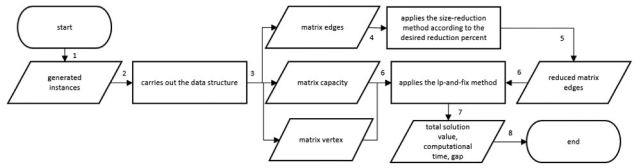

Figure 5. Hybrid algorithm heuristic procedure.

```
begin
INPUT generated_instance;
        reading_data()        / /carries out the data structure
        INPUT generated_instance;
            repeat
                read each element of data structure edges and store in the matrix_edges[ ][ ];
                read each element of data structure capacity and store in the matrix_capacities[ ][ ];
                read each element of data structure vertex and store in the matrix_vertices[ ][ ];
            until all the data has been read;
        OUTPUT matrix_edges[ ][ ], matrix_capacities[ ][ ], matrix_vertices[ ][ ];
    size_reduction()      //applies the size reduction method
        INPUT matrix_edges[ ][ ], reduction_percent;
            repeat
                reduce the nodes in matrix_edges[ ][ ];
            until all the nodes be reduced according to reduction_percent;
        OUTPUT matrix_edges[ ][ ]; //reduced
    lp_and_fix()      //applies the lp and fix method
        INPUT matrix_edges[ ][ ], matrix_capacities[ ][ ], matrix_vertices[ ][ ];
            relax all the positions of the vector x[ ] with values from 0 until 1;
            solve the problem using cplex;
            fix the position of the vector x[ ] with values 0, as 0 and 1, as 1;
            solve the problem using cplex;
        OUTPUT total_solution_value, computational_time, solution_gap;
end
```

Figure 6. Hybrid algorithm.

Hybrid algorithm: combining the SR and the LP-and-Fix heuristics

We have developed a third approach to solve the MCPTWHF problem, by hybridizing the SR heuristic and the LP-and-Fix heuristic. Figure 5 presents a flowchart with this hybrid heuristic procedure.

In a simple way, this approach starts by reducing the number of arcs by a pre-established percentage of the *edges* matrix. Then, the algorithm takes into consideration the obtained matrix, jointly with other data structures (*capacities* and *vertices*) and apply the LP-and-fix approach to solve the problem with the relaxed binary decision variables x. Afterward, the value of these variables are fixed to 1 or 0, according to the above criterion, and the problem is solved again (as described in section 5.2 above).

The algorithm is developed using the C++ language and its interface with CPLEX is made using the Concert technology (see Figure 6).

Computational results

In this section we present the computational results obtained by a comprehensive set of tests, performed using the four methods described below, and applied to 60 instances randomly created by a specially designed generator. Such preliminary assessment was performed with the IBM ILOG CPLEX software, with the main purpose of validating the model and checking whether there were bugs or inconsistencies. These tests were run in a CPU Intel Core i7-4710 HQ 3.5 GHz and 8 GB memory.

Instances generator

We have generated 3 classes of instances in order to test the performance of the developed solution methods, when applied to the MCPTWHF. The problem classes vary according to the number of suppliers and nodes, which will significantly impact the number of arcs in the network. For each class, 4 variants were established, with different number of nodes and suppliers. So, each class has 20

Table 3. Instances classes.

Problem class	class I: Small				class II: Medium				class III: Large				Total
Nodes	6	8	10	12	10	12	14	16	12	16	18	20	60
Suppliers	10	6	4	2	8	6	4	2	10	6	4	2	
Instances	5	5	5	5	5	5	5	5	5	5	5	5	
Total	20				20				20				

instances, and there are 3 classes (small, medium and large instances), totalizing 60 problem instances. In Table 3 we present the characteristics of the generated test instances.

We generated the instances using an algorithm developed in C++ (on the Microsoft Visual Studio IDE), using guidelines proposed by Alvarez, González-Velarde, and de-Alba (2005), in order to produce feasible problems, especially considering the arcs capacity relative to the network demand. These guidelines include some rules to determine the demand for the products by node (values between 1 and 100), for the arc costs (values between 1 and 100) and for the vehicle capacities. For these capacities, we have considered values between 24% and 28%, as the values proposed by Alvarez, González-Velarde, and de-Alba (2005) – between 12% and 16% of the total demand of each product – lead to a high number of infeasible instances). The pseudocode of the instance generator is presented in Figure 7.

In order to better analyze the efficiency of the heuristics, the CPLEX presolve configurations were turned off. Tests were made for the 60 instances, with each of the previously described methods, and considering both objective functions (cost and time). Therefore, the total number of tests was 480 (60 x 4 × 2).

Analysis of results

In this section, computational results are presented and analyzed. As referred, four 'methods' were used and compared, as follows:

- MILP: solving the problem with mixed-integer linear programming, without pre-solve heuristics;
- SR: using the Size Reduction heuristic to reduce the problem size, and then solve the resulting problem with mixed-integer linear programming;
- LPF: using the LP and Fix heuristic and mixed-integer linear programming (relax the binary variables x, solve the resulting problem using mixed-integer linear programming, fix the value of variables to '0' or '1', and solve the problem again);
- HA: hybrid algorithm that combines the two heuristics (first, problem data is reduced by the Size Reduction procedure, and then the problem is solved with mixed-integer linear programming, using the LPF heuristic).

As performance measure, we used the relative percentage deviation (RPD) which is calculated as follows:

$$RPD = \frac{v_{METHOD} - v_{BEST}}{v_{BEST}} \times 100 \quad (18)$$

where v_{METHOD} denotes the best objective function value obtained by a given method, while v_{BEST} denotes the best objective function value obtained. To summarize the computational results, we have calculated the average RPD (ARPD) for a given method, taking into account the runs in a given set of instances.

Objective function *cost*

Considering the objective function *cost* for the small instances, in general the MILP and LPF methods achieved results slightly better than the SR and HA methods. This happens because, when the *edges* structure is reduced by a size-reduction procedure, the algorithm has to deal with fewer arcs. The same occurs in the medium and large instances.

Regarding the execution times for small instances, the LPF method achieved the best results, but closely followed by the HA method. The SR method, for the first 5 instances, obtained results a little bit worse than the LPF and HA methods. The execution times of the SR method worsened a little in relation to the LPF and HA methods, from instance 7 to instance 16, getting worse again at instances 19 and 20.

The same effect happens with the MILP method, but, in general, with worse execution times. We can observe yet the presence of an outlier value at instance 11. In relation to the medium-size instances category, we can observe more diversified results, with the LPF method achieving the best results of execution time, followed by the MILP method and by the HA

```
begin
INPUT nodes_number, products_number;
    graph()          //generates a randomized graph according to nodes_number
        INPUT nodes_number;
            repeat
                link nodes by a random way forming arcs;
            until all the nodes be linked
         OUTPUT graph;
    transport_lines();          // generates a randomized transport lines according to the graph
        INPUT graph;
            repeat
                define random sets of sequenced arcs;
                account the number of sets;
             until all the arcs be part of a sequenced set;
            define each set as a transport_line;
        OUTPUT transport_lines;
    data_structure()            //generates the problem parameters to each arc, capacity and vertex
        INPUT graph, transport lines;
            repeat
                generates demand values to each node and product;      // between 1 and 100
                generates cost values to each arc;   //between 1 and 100
                generates vehicles number to each transport line;      // between 5 and 15
                generates capacities to each vehicle of each transport line;          // 24 % to 28% of the total demand to
                each product
                generates time windows to each vehicle of each transport line;      //arrive, departure and travel times
            until all graph and transport line be with the proper information
        OUTPUT demand, cost, vehicles, capacities, time windows;
end
```

Figure 7. Instance generator.

Table 4. Execution times and average values/standard deviations for the *cost* objective function.

		Objective value				Execution time			
Statistical measure	Instance type	MILP	SR	LPF	HA	MILP	SR	LPF	HA
Average	Small	52728.97	60449.91	52728.755	60450.41	251.31	58.23	14.92	17.38
	Medium	52454.71	55905.31	47223.95	55906.28	312.24	482.99	54.62	273.64
	Large	73656.70	81820.79	56492.75	83388.57	879.73	2229.48	719.81	1414.08
Standard deviation	Small	29799.19	31038.18	29798.84	31038.28	790.57	20.98	6.41	5.88
	Medium	24960.75	26590.14	21980.35	26589.22	154.28	233.09	25.64	227.42
	Large	54329.96	64369.08	55629.69	62427.47	467.13	929.74	1280.65	609.11
Average Relative Percentage Deviation	Small	0.00%	14,64%	0,00%	14,64%	1584%	290%	0%	16%
	Medium	11,08%	18,38%	0,00%	18,39%	472%	784%	0%	401%
	Large	30,38%	44,83%	0,00%	47,61%	22%	210%	0%	96%

method, and with the SR method in the last position. The same occurs for the large-size instances.

It should be noted that, for large instances, the size-reduction procedure, present in the SR and HA methods, used almost 50% of the total algorithm time in the SR method and approximately 90% of the time in the HA method. We can therefore conclude that the procedure is not so good (in terms of computational time) when we have a lot of arcs to analyze. Table 4 summarizes these results..

In terms of the average values of the objective function, we can observe (Table 4) that the MILP and LPF methods achieved results slightly better than the SR and HA methods, for the small instances. For the medium-size instances, the LPF method presented the best result, followed by the MILP method, with the SR and HA methods staying close to each other. The same occurs to the large instances, with the difference that the SR average value was significantly better than the HA average value.

However, standard deviations were high for all the instances, this meaning a high level of dispersion among the values.

In what concerns the average execution times, we can see that, for small instances, the LPF method achieved the best value, followed by the HA method. For medium-size instances, the LPF method keeps the first position. followed by the HA method. For the large instances, we have, from the best performance to the worst: LPF, MILP, HA and SR.

The standard deviations in average were larger for the large instances (specially for the LPF and the SR methods) and medium-size instances (except for the LPF method), featuring a high level of data dispersion. For small instances standard deviations were low, except for the MILP method, probably because of the outlier value of instance 11. Table 5 summarizes these results.

From this analysis, we can see that, in general, the best performance for the objective function *cost* was achieved by the LPF method, followed by the HA method.

Table 5. Summary of the best and worst performance for the objective function *cost*.

	Best method			Worst method		
Feature	Small instances	Medium instances	Large instances	Small instances	Medium instances	Large instances
less infeasible instances	equals	MILP and LPF	HA	equals	SR	MILP and LPF
total less infeasible instances		HA			SR	
less GAP values different from zero	MILP, SR and HA	MILP, and LPF	MILP	SR	SR	SR
maximum execution time	HA	LPF	LPF	MILP	SR	SR and HA
minimum execution time	LPF	LPF	LPF	MILP and SR	SR	SR
general objective function results	MILP and LPF	MILP and LPF	MILP and LPF	SR and HA	SR and HA	SR and HA
general execution time results	LPF	LPF	LPF	MILP	SR	SR
average values objective function	MILP and LPF	LPF	LPF	HA	HA	HA
average values execution time	LPF	LPF	LPF	MILP	SR	SR

Objective function *time*

Considering the objective function *time* for the small instances, in general, the MILP and LPF methods achieved results slightly

Table 6. Execution times and average values and standard deviations for the *time* objective function.

		Objective value				Execution time			
Statistical measure	Instance type	MILP	SR	LPF	HA	MILP	SR	LPF	HA
Average	Small	1073.7	1241	1283.2	1442.2	3600.00	3600.00	49.82	45.50
	Medium	429.8	472.4	500.8	558.6	3600.00	3600.00	154.47	200.69
	Large	55626	44515.2	568.8	627.8	3600.00	3600.00	569.33	1578.04
Standard deviation	Small	618.18	711.91	722.35	797.74	0.00	0.00	20.38	24.59
	Medium	185.29	207.86	216.19	259.91	0.00	0.00	45.89	168.11
	Large	56598.34	43524.63	303.37	364.85	0.00	0.00	214.02	616.70
Average Relative Percentage Deviation	Small	0.00%	15,58%	19,51%	34,32%	7812%	7812%	9%	0%
	Medium	0,00%	9,91%	16,52%	29,97%	2231%	2231%	0%	30%
	Large	9679,54%	7726,16%	0,00%	10,37%	532%	532%	0%	177%

Table 7. Summary of the best and worst performance for the objective function *time*.

	Best method			Worst method		
Feature	Small instances	Medium instances	Large instances	Small instances	Medium instances	Large instances
less infeasible instances	HA	HA and LPF	HA	MILP and SR	MILP	MILP
total less infeasible instances		HA				
less GAP values different from zero	LPF	HA	LPF	MILP and SR	MILP and SR	MILP and SR
maximum execution time	LPF	LPF	LPF	MILP and SR	MILP and SR	MILP, SR and HA
minimum execution time	HA	HA	LPF	MILP and SR	MILP and SR	MILP and SR
general objective function results	MILP and LPF	MILP and LPF	LPF and HA	SR and HA	HA	MILP
general execution time results	LPF and HA	LPF	LPF	MILP and SR	MILP and SR	MILP and SR
average values objective function	MILP	MILP	LPF	HA	HA	MILP
average values execution time	HA	LPF	LPF	MILP and SR	MILP and SR	MILP and SR

better than SR and HA. Again, this happens because, due to the size reduction procedure, the algorithm now has to deal with fewer arcs.

The same can be observed in the medium-size instances (with an advantage of SR in relation to HA). For the large instances, the best results were obtained by the LPF and HA methods, with worse results for MILP and SR.

In what concerns execution times, for small instances, LPF and HA obtained the best results (execution times of MILP and SR were equal to the pre-defined time limit). Moreover, for the medium instances, the execution times of LPF were a little better than HA in instance 38. For the large instances, execution times of LPF were the best. Table 6 summarizes these results.

For small and medium-size instances, regarding the average objective values, MILP obtained the best results, followed by the SR method, For the large instances, the LPF method was the best, being slightly better than HA.

Moreover, SR and MILP obtained very bad results, when compared to the other two methods. As in the *cost* case, standard deviations were quite high for all instances.

In Table 6, we can also observe that for the execution times, in the case of the MILP and SR methods, the time limit values were reached for the overall instance categories. In relation to the comparison between the LPF and HA methods, for the small instances, the HA method presented better results. For medium-sized and large-sized instances, the LPF method performed better than the others.

In all instance categories, standard deviations for MILP and SR do not suggest any relevant common behavior. For small instances, LPF and HA presented a small dispersion. In the case of medium-size instances, the dispersion was smaller for LPF and moderated for HA. And for the large instances, it was moderated in LPF, and high in HA. Table 7 presents a summary of these results.

From the previous analysis, we can conclude that, in general, the best performance for the *time* objective function, was achieved by the HA method, followed by the LPF method.

Conclusions

In general terms, we can conclude that the model and the resolution methods presented good, promising results, when applied to a representative set of randomly generated problem instances.

Based on this analysis, we can state that the *cost* objective function is easier to handle than the *time* objective function. This happens because time seems to be more restrictive than cost.

For the *cost* objective function, the method that presented the best performance, in general terms, was the LPF method (with the LP and fix heuristic). But when we consider the number of instances with a feasible solution found (a possibly more important evaluation dimension), the HA method becomes the best. This happens because the HA method incorporates the size reduction heuristic, which helps to reduce the number of arcs from the *edges* structure, thus decreasing the total number of arcs to be analyzed. However, this comes with a cost: the quality of the solution is slightly reduced.

The LP-and-Fix heuristic seems to be a good technique for solving the problem, because when the binary decision variables are relaxed, the resulting problem becomes easier to solve. Then, variable fixing reduces the number of arcs, making the problem much easier to solve.

The SR method (with the size reduction heuristic) did not present good solutions when applied to the *cost* objective function, but has a much better performance when applied to the *time* objective function, when compared to the MILP method (based on mixed integer linear programming). This happens because the *time* objective function increases the complexity of the problem, making it more dependent on the use of heuristic techniques.

However, when combined with the LP-and-Fix technique, the size reduction heuristic presented very good results for the *time* objective function (as seen in the HA method). Therefore, we can conclude that, in general, when considering both objective functions, the HA method presented the best results (followed by the LPF method).

In terms of computational times, the SR heuristic presented good results for small instances, but when the number of arcs increases, the number of possible options to the exclusion of arcs (decision variables) is exponential, making the computational cost higher. The LP-and-Fix heuristic presented the highest efficiency in terms of computational costs, along with the value of solutions. This is, therefore, the heuristic procedure with the best performance for the multi-commodity network flow problem with time windows and multiple transport lines.

However, one significant weakness (limitation) of the model is the need of rather complex data structures. The high number of 'indices' increased the number of possibilities a lot, especially on the 'edges' data structure. We can, therefore, conclude that it would be particularly difficult to develop a heuristic/metaheuristic approach to deal directly with the problem (as modeled), without the use of an optimization tool.

This research naturally generated several ideas for future work:

- since the problem studied here is a new variant of the multi-commodity network flow problem, an important additional contribution would be to apply other heuristics/metaheuristics for the resolution of the model;
- it would be interesting to develop a multi-objective approach, integrating the objective functions *cost* and *time*, in order to find trade-off, practically interesting solutions;

- another important contribution would be to apply some variant of the *tridimensional bin packing problem* to optimize the use of the vehicles' baggage pack; and finally
- it seems there would be a considerable potential in designing a *decision support system* to integrate the information related to urban freight origins/destinations, thus improving network management in practice.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was partially financed by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT - Fundacção para a Ciência e a Tecnologia within project POCI-01-0145-FEDER-032053 and National Council for Scientific and Technological Development (CNPq) through grants [404232/2016-7 and 303594/2018-7]."

ORCID

Jorge Pinho de Sousa http://orcid.org/0000-0002-9292-0386
Bruno de Athayde Prata http://orcid.org/0000-0002-3920-089X

References

Abdulhakim, F. Z. 2013. "Linear Capacity and Heuristic Algorithms for Multicommodity Flow in Computer Networks Engineering." In *Advanced Materials Research*, 926–931. Trans Tech Publications.

Aloise, D., and C. C. Ribeiro. 2011. "Adaptive Memory in Multistart Heuristics for Multicommodity Network Design." *Journal of Heuristics* 17 (2): 153–179. doi:10.1007/s10732-010-9130-6.

Alvarez, A. M., J. L. González-Velarde, and K. de-Alba. 2005. "GRASP Embedded Scatter Search for the Multicommodity Capacitated Network Design Problem." *Journal of Heuristics* 11 (3): 233–257. doi:10.1007/s10732-005-1509-4.

Assad, A. A. 1978. "Multicommodity Network Flows—a Survey." *Networks* 8 (1): 37–91. doi:10.1002/net.3230080107.

Barnhart, C., C. A. Hane, E. L. Johnson, and G. Sigismondi. 1994. "A Column Generation and Partitioning Approach for Multi-commodity Flow Problems." *Telecommunication Systems* 3 (3): 239–258. doi:10.1007/BF02110307.

Barnhart, C., N. Krishnan, D. Kim, and K. Ware. 2002. "Network Design for Express Shipment Delivery." *Computational Optimization and Applications* 21 (3): 239–262. doi:10.1023/A:1013721018618.

Barnhart, C., and Y. Sheffi. 1993. "A Network-based Primal-dual Heuristic for the Solution of Multicommodity Network Flow Problems." *Transportation Science* 27 (2): 102–117. doi:10.1287/trsc.27.2.102.

Crainic, T. G., M. Gendreau, and J. M. Farvolden. 2000. "A Simplex-based Tabu Search Method for Capacitated Network Design." *INFORMS Journal on Computing* 12 (3): 223–236. doi:10.1287/ijoc.12.3.223.12638.

Cruz, F. R. B., J. M. Smith, and G. R. Mateus. 1998. "Solving to Optimality the Uncapacitated Fixed-charge Network Flow Problem." *Computers & Operations Research* 25 (1): 67–81. doi:10.1016/S0305-0548(98)80010-5.

Even, S., Itai, A., & Shamir, A. (1976). On the complexity of time table and multi-commodity flow problems. In 16th Annual Symposium on Foundations of Computer Science, (pp. 184–193). IEEE doi:10.1109/SFCS.1975.21

Farvolden, J. M., and W. B. Powell. 1994. "Subgradient Methods for the Service Network Design Problem." *Transportation Science* 28 (3): 256–272. doi:10.1287/trsc.28.3.256.

Farvolden, J. M., W. B. Powell, and I. J. Lustig. 1993. "A Primal Partitioning Solution for the Arc-chain Formulation of A Multicommodity Network Flow Problem." *Operations Research* 41 (4): 669–693. doi:10.1287/opre.41.4.669.

Frangioni, A., and G. Gallo. 1999. "A Bundle Type Dual-ascent Approach to Linear Multicommodity Min-cost Flow Problems." *INFORMS Journal on Computing* 11 (4): 370–393. doi:10.1287/ijoc.11.4.370.

Gabrel, V., A. Knippel, and M. Minoux. 1999. "Exact Solution of Multicommodity Network Optimization Problems with General Step Cost Functions." *Operations Research Letters* 25 (1): 15–23. doi:10.1016/S0167-6377(99)00020-6.

Guardia, L. T., and G. B. Lima. 2010. "Interior Point Methods for Linear Multicommodity Flow Problems." In *Proceeding of SR International Conference on Engineering optimization*.

Helme, M. P. 1992. "Reducing Air Traffic Delay in a Space-time Network." In *Systems, Man and Cybernetics, 1992., IEEE International Conference on*. 236–242. IEEE.

Holmberg, K., and D. Yuan. 2003. "A Multicommodity Network-flow Problem with Side Constraints on Paths Solved by Column Generation." *INFORMS Journal on Computing* 15 (1): 42–57. doi:10.1287/ijoc.15.1.42.15151.

Kennington, J. L. 1978. "A Survey of Linear Cost Multicommodity Network Flows." *Operations Research* 26 (2): 209–236. doi:10.1287/opre.26.2.209.

Kim, J.-G., H.-B. Jun, and C.-M. Kim. 2011. "A Two-phase Heuristic Algorithm for the Fixed-charge Capacitated Network Design Problem with Turn Penalties." *KSCE Journal of Civil Engineering* 15 (6): 1125–1132. doi:10.1007/s12205-011-1318-2.

Kirby, M. W., W. A. Hager, and P. Wong. 1986. "Simultaneous Planning of Wildland Management and Transportation Alternatives." *TIMS Studies in the Management Sciences* 21: 371–387.

Kleeman, M. P., B. A. Seibert, G. B. Lamont, K. M. Hopkinson, and S. R. Graham. 2012. "Solving Multicommodity Capacitated Network Design Problems Using Multiobjective Evolutionary Algorithms." *IEEE Transactions on Evolutionary Computation* 16 (4): 49–471. doi:10.1109/TEVC.2011.2125968.

Lagos, C., B. Crawford, R. Soto, J.-M. Rubio, E. Cabrera, and F. Parades. 2014. "Combining Tabu Search and Genetic Algorithms to Solve the Capacitated Multicommodity Network Flow Problem." *Studies in Informatics and Control* 23 (3): 266. doi:10.24846/v23i3y201405.

Lim, C., and J. C. Smith. 2007. "Algorithms for Discrete and Continuous Multicommodity Flow Network Interdiction Problems." *IIE Transactions* 39 (1): 15–26. doi:10.1080/07408170600729192.

Lin, Z., and R. Kwan. 2017. "Multicommodity Flow Problems with Commodity Compatibility Relations." In *ITM Web of Conferences, Volume 14, 2017 The 12th International Conference Applied Mathematical Programming and Modelling–APMOD 2016*. EDP Sciences.

Maes, J., J. O. Mcclain, and L. N. Van Wassenhove. 1991. "Multilevel Capacitated Lotsizing Complexity and LP-based Heuristics." *European Journal of Operational Research* 53 (2): 131–148. doi:10.1016/0377-2217(91)90130-N.

Malairajan, R. A., K. Ganesh, T. R. Lee, and S. P. Anbuudayasankar. 2013. "REFING: Heuristic to Solve Bi-objective Resource Allocation Problem with Bound and Varying Capacity." *International Journal of Operational Research* 17 (2): 145–169. doi:10.1504/IJOR.2013.053620.

Masri, H., S. Krichen, and A. Guitouni. 2015. "A Multi-start Variable Neighborhood Search for Solving the Single Path Multicommodity Flow Problem." *Applied Mathematics and Computation* 251: 132–142. doi:10.1016/j.amc.2014.10.123.

Oimoen, S. C. 2009. *Dynamic Network Formation Using Ant Colony Optimization*. Air Force Institute of Technology.

Ouorou, A., P. Mahey, and J.-P. Vial. 2000. "A Survey of Algorithms for Convex Multicommodity Flow Problems." *Management Science* 46 (1): 126–147. doi:10.1287/mnsc.46.1.126.15132.

Tadayon, B., and J. C. Smith. 2014. "Algorithms for an Integer Multicommodity Network Flow Problem with Node Reliability Considerations." *Journal of Optimization Theory and Applications* 161 (2): 506–532. doi:10.1007/s10957-013-0378-5.

Topaloglu, H., and W. B. Powell. 2006. "Dynamic-programming Approximations for Stochastic Time-staged Integer Multicommodity-flow Problems." *INFORMS Journal on Computing* 18 (1): 31–42. doi:10.1287/ijoc.1040.0079.

Wang, I. L. (2018). Multicommodity network flows: A survey, Part I: Applications and Formulations. International Journal of Operations Research, 15(4), 145–153

Wei, K., et al. 2014. "A Simulated Annealing Based Heuristic for the Multi-source Single-path Multi-commodity Network Flow Problem." In *Service Systems and Service Management (ICSSSM), 2014 11th International Conference on*. 1–6. IEEE.