Solving a Real-World Multi-attribute VRP Using a Primal-Based Approach

Mayssoun Messaoudi[1,3(✉)], Issmail El Hallaoui[1,3], Louis-Martin Rousseau[1,2], and Adil Tahir[1,3]

[1] Mathematics and Industrial Engineering Department, Polytechnique Montréal, Montréal, QC H3C 3A7, Canada
mayssoun.messaoudi@polymtl.ca
[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montréal, QC H3C 3A7, Canada
[3] Group for Research in Decision Analysis (GERAD), Montréal, QC H3C 3A7, Canada

Abstract. Through this paper we focus on a real-life combinatorial problem arising in emergent logistics and transportation field. The main objective is to solve a realistic multi-attribute rich Vehicle Routing Problem using a primal-based algorithm embedded in column generation framework. The mathematical model is formulated as a Set Partitioning Problem (SPP) while the subproblem is the shortest path problem with resource constraints (SPPRC). The numerical study was carried out on real instances reaching 140 customers. The successful results show the effectiveness of the method, and highlight its interest.

Keywords: VRP · Column generation · Primal algorithm

1 Introduction

Supply chain encompasses several integrated activities and hand-offs allowing physical and information flows to be routed from the source (supplier) to the final destination (customer). Many believe that nearly two thirds of the supply chain total cost is related to transportation, more specifically, trucking is the dominant spend component.

Indeed, in a customer-centric era, transportation industry is becoming increasingly complex, and firms are facing a serious imperative: being able to deliver efficiently a whatever-whenever-wherever while respecting time, cost and quality. However, traditional networks, inefficient and fragile systems with limited computing performance are crippling firms to provide high-quality service, and maintain growth. In that respect, logistics providers are called upon to sharpen their practices through resilient tailor-made approaches.

Our main objective is to solve a real-world routing problem subject to a set of constraints and specific business rules commonly encountered in logistics industry markets. Our solution approach is based on a primal-based algorithm in

M. Baïou et al. (Eds.): ISCO 2020, LNCS 12176, pp. 286–296, 2020.
https://doi.org/10.1007/978-3-030-53262-8_24

column generation framework. The aim is not only to efficiently generate optimal dispatching plans, but also to shed light on the opportunity offered by such methods, especially when deployed on large-scale problems. In the remainder of this paper, note that all the mathematical formulations will be presented in their maximization form, but results also hold for a minimization scenario.

2 Related Literature

Vehicle Routing Problems (VRP) are so popular, and are the subject of an intensive literature due to their wide application in logistics and freight industry. Since its introduction by [1,2], several approaches regarding modelling and solution methods have been proposed for many VRP variants. Classical VRP aims to design minimal cost routes for a fleet of identical vehicles such that each customer is served exactly once, the capacity is respected and each vehicle starts and ends its route at the depot. In a-demand-driven-supply chain, many requirements and constraints arise, thus we switched from the classical VRP to new and more combinatorial and difficult models [3] which combine not only the usual restrictions such as time windows and fleet structure, but also specific business rules that vary according to the context. A detailed study of those variants can be found in [4]. Since VRP is NP-hard [5], heuristics and metaheuristics are more suitable than exact methods which are difficult to implement on large-scale problems. According to the classification of [6], solution methods are classified in two main classes: dual fractional and primal methods. Dual fractional methods [6] maintain iteratively optimality and feasibility until integrality is achieved. Whereas primal or augmentation methods maintain both feasibility and integrality and stop when optimality is reached.

One of the most known dual fractional methods is the branch-and-price (B&P) algorithm which combines branch-and-bound ($B\&B$) and column generation. The latter is an iterative process that solves the linear relaxation of the problem called a master problem (MP) for a restrictive subset of variables (columns), then called the restricted master problem (RMP). The duals related to the RMP's constraints are sent to a subproblem (SP) in order to generate positive-reduced cost variables, to be added into the RMP. The process stops when no such variables exist, and an optimal solution is obtained. If the latter is fractional, $B\&B$ is applied using a suitable branching strategy. However, despite its overall success, convergence problems can occur and affect the method's efficiency [7], they could be circumvented by using the stabilization strategies found in the literature and a fine-tuning strategy.

For VRPs, column generation is one of the notable exact methods that have successfully solved large and complex problems [8]. In such a context, the master problem is often modelled as a Set Partitioning Problem (SPP) and the subproblem is the Shortest Path Problem with Resource Constraints (SPPRC) defined on a directed graph [9], and usually solved with the dynamic programming algorithm introduced by [10].

Interestingly, exact primal methods have attracted very little interest in the literature, furthermore, concrete and adapted realizations of these approaches

remain marginal. In a simple way, a primal procedure moves from one integer solution to an improving adjacent one until optimality, and such a sequence of moves leading to an improving solution is called a descent direction. With respect to the usual notation, a descent direction refers to augmentation direction in a maximization context as well. The first primal approach was introduced by [11], then [12] set the concept of the integral augmentation problem. An interesting combined approach was proposed by [13,14] that showed how integral simplex can be properly embedded in column generation context. They used an adequate branching strategy to obtain an optimal solution. Although, these models are restricted to small instances as they haven't been able to overcome the high degeneracy of SPPs. Integral Simplex Using Decomposition (ISUD) is one of the most promising primal methods, it was introduced by [15] to solve large-scale SPPs. Based on the improved primal simplex algorithm [16], ISUD takes advantage of degeneracy and finds strict descent directions at each iteration leading to an optimal solution. ISUD's performance has been enhanced by adding secant plans to penalize fractional descent directions [17], and also by exploring neighborhood search [18]. ISUD performed excellent computational results for SPPs with up to 500.000 variables. Recently, [19] introduced integral column generation (ICG). This three-stage sequential algorithm combines ISUD and column generation to solve SPPs. Experiments on large-scale instances of Vehicle and Crew Scheduling Problem (VCSP) and Crew Pairing Problem (CPP) showed that ICG exceeds two well-known column generation-based heuristics. The authors invoked the possibility to adapt ICG even on SPP with side constraints.

The remarkable performance of ISUD and ICG algorithms makes them worth pursuing. We believe that such methods have to be experimented on complex and well-known problems of literature such as rich routing problems.

2.1 Organization of the Paper and Contributions

To the best of our knowledge, it is the first time that a primal algorithm based on an exact method has been used to solve a rich vehicle routing problem. It is also an opportunity to discuss the procedure's performance on a real-world combinatorial problem. On the one hand, experimentation on real instances showed that the used primal method finds very good results in a short computing time. Indeed, it outperforms a well-known branch-and-price heuristic. On the other hand, the solution has led to a positive impact on the company's outcome indicators. This paper is organized as follows. In Sect. 3, we describe our problem and give the related notation. In Sect. 4, we give the mathematical formulation of master problem and subproblem. In Sect. 5, we introduce some theoretical notions related to the primal approach. The solution method (ICG) is described in the Sect. 6. The computational results are reported in the Sect. 7. Finally, concluded remarks are presented in the Sect. 8.

3 Problem Statement

In the remainder of the paper, we use the notation organized in the Table 1 bellow.

Table 1. Definition of the parameters and variables

Notation	Type	Description
Ω	–	Set of feasible routes
N	–	Set of customers to visit
K	–	Set of heterogeneous k-type vehicles
a_{ir}	Binary	Is equal to 1 if and only if customer i is served by route r
d_i	Real	Demand associated with customer $i \in N$
l_k	Real	Travelled distance between origin and destination by k-vehicle $\in K$
p_r	Real	Profit collected by the route $r \in \Omega$
q_k	Real	Capacity of vehicle $k \in K$
s_i	Real	Service time at customer $i \in N$
w^k	Real	Accumulated working time of k-vehicle $\in K$
θ_r	Binary	Is equal to 1 if and only if route r is selected

Given a set of customers $i \in N$ geographically scattered, a logistic hub O handles their transport operations. The daily task is to ensure next-day deliveries within specific time frames imposed by either customer convenience or/and urban traffic regulation, using heterogeneous vehicles with different capacities, types and operating costs. In addition, specific business rules such as the driving hours set up by work unions, loading rate, and urban accessibility regulation must be satisfied.

The objective is to maximize the profit collected by routes, resulting in the difference between freight rates and freight costs, while satisfying the following constraints:

1. Each customer $i \in N$ is visited by a single route $r \in \Omega$
2. Each customer $i \in N$ is visited within the time window $[a_i, b_i]$
3. Each customer $i \in N$ is visited by an allowed k-vehicle
4. The total load $\sum_{i \in N} d_i$ on the route travelled by k-vehicle does not exceed the vehicle's capacity q_k
5. The total travelling time l_k, including service times s_i, does not exceed the allowed working time w^k of k-vehicle

4 Mathematical Models

The master problem and the subproblem are formulated as SPP and SPPRC, respectively.

4.1 Master Problem

With respect to the below-mentioned notation, we formulate the problem as follows:

$$(SPP): \quad \max_{\theta} \quad \sum_{r\in\Omega} p_r\theta_r \tag{1}$$

$$\text{s.t.} \quad \sum_{r\in\Omega} a_{ir}\theta_r = 1 \qquad \forall i \in N, \tag{2}$$

$$\theta_r \in \{0,1\} \qquad \forall r \in \Omega \tag{3}$$

Each variable θ_r is associated with a feasible route $r \in \Omega$ which specifies a sequence of customers $i \in N$ to be served. The objective function (1) aims to maximize the profit made by the feasible route r. The constraints (2) guarantee that each customer is delivered exactly once. The choice of binary variables is imposed by (3).

4.2 SPPRC on $G(V, A)$

The subproblem is modelized as a shortest path problem with resource constraints, and is solved using a labelling algorithm as shown in [20]. We have one supbroblem for each $k-$vehicle, put simply we omit the k-index. The reduced cost of feasible route r travelled by $k-$vehicle, starting and ending at the depot O and visiting a sequence of customers $i \in N$ is computed as:

$$\overline{p}_r = p_r - \sum_{i\in N} a_{ir}\pi_i > 0$$

The dual variable π is associated to the partitioning constraints (2). If all columns have negative reduced cost, the algorithm stops and an optimal solution is obtained for the linear relaxation of (SPP) (1–2). For each k-vehicle, SPPRC is defined on a cyclic graph $G = (V, A)$. V contains $|N| + 2$ vertices, one vertex for each customer $i \in N$, and (s, t) pair where s and t both refer to the depot O. A contains departure arcs (s, j), $\forall j \in N$, arrival arcs (i, t), $\forall i\ N$ and connecting arcs (i, j), $\forall (i, j) \in N$ so that the client j can be reached after the client i by a realistic route as indicated by the actual road map. Let $\mathbf{\Gamma} = \{\mu_1, \mu_2, \ldots, \mu_{|\Gamma|}\}$ be the set of resource constraints, and L_i^{μ} be the label related to the resource $\mu \in \mathbf{\Gamma}$ and associated to the vertex $i \in N$ such that the resource window $[a_i^{\mu}, b_i^{\mu}]$ is respected.

In our case, we consider the following resources:

- $L_i^T \in [a_i, b_i]$: The time resource indicates the arrival time at customer $i \in V$. L_i^T could be less than a_i, i.e., driver might arrive before the starting delivery period.
- $L_i^D \in [0, q_{max}]$: The demand resource specifies the accumulated load until customer $i \in V$.

- $L_i^W \in [0, 480\,\text{min}]$: The working time resource specifies the total time travelled by the driver, from $\{O\}$ to customer $i \in V$.
- L_i^c: The cost resource is unconstrained, and used to compute the reduced cost of every travel arc $(i, j) \in A$.

The label represents a feasible partial path if: $L_i^\mu \in [a_i^\mu, b_i^\mu]$, $\forall\ \mu \in \mathbf{\Gamma}$. In such case, the label is feasible and extended along the $(i, j) \in A$ by calling for a resources-extension function, denoted $f_{ij}(L_i^\mu)$. While the labelling algorithm solves the subproblem, the SPP calls for ICG algorithm as explained in the Sect. 6.

5 Preliminaries

For the sake of clarity, we introduce some preliminaries concerning the primal approach principles, and the ISUD algorithm used in our solution procedure. We remind that detailed literature and examples could be found in [15,16,19].

As mentioned earlier, the primal methods have paid particular attention to SPPs. In addition to their popularity in routing and scheduling, these problems satisfy the Trubin theory. If we denote X the SPP polytope and X_S the set of its integer solutions, [21] shows that every edge of $Conv(X_S)$ is an edge of X, then SPP is said quasi-integral. This property makes it possible to use linear-programming pivots to reach integer extreme points.

Let consider the (SPP):

$$z^* = \max_{\boldsymbol{\theta}} \left\{ \mathbf{p}^\top \boldsymbol{\theta} \;\middle|\; A\boldsymbol{\theta} = \mathbf{e},\ \boldsymbol{\theta} \in \{0,1\}^n \right\} \tag{4}$$

Let $A \in \{0,1\}^{m \times n}$ be the binary constraint matrix, and let $A_1, A_2, \ldots A_n$ be the columns in A indexed in $J = \{1, 2, \ldots, n\}$ such that A_j denotes the j^{th} column in A.

$$A = \begin{bmatrix} A_1 & A_2 & \ldots & A_n \end{bmatrix}$$

$\mathcal{F}_{SPP} \subseteq \{0,1\}^n$ denotes the feasible region of SPP, while $\mathcal{F}_{SPP}^R$ denotes the feasible region of the linear relaxation of SPP. $\theta_r^* \in \mathcal{F}_{SPP}$ denotes the optimal solution and z_{SPP}^* is the optimal solution value.

Definition 1. *A column A_j is said to be **compatible** with S if $A_j \in Span(S)$, i.e., it can be written as a linear combination of the columns in S, otherwise, it is said to be incompatible.*

Definition 2. *The **incompatibility degree** δ_j of a column A_j towards a given integer solution is a metric measure that represents a distance of A_j from the solution. We note that δ_j of a compatible column is zero.*

Based on the definition of compatibility, ISUD decomposes the columns in A into three sets such that:

$$A = \begin{bmatrix} S & C & I \end{bmatrix}$$

Where S, C and I denote respectively the working basis (the support of the current integer solution), compatible columns subset and incompatible columns subset.

As described in Algorithm 1, the RMP is decomposed into two small subproblems: The complementary problem (CP) handles the incompatible columns and finds a descent direction $\mathbf{d}$ leading to an improved solution, while the reduced problem (RP) handles the compatible columns and seeks to improve the current solution.

Algorithm 1: *ISUD* pseudocode

```
Find initial solution θ^0 and set θ̄ ← θ^0
[S  C  I] ← Partition the binary matrix A into columns subsets
do
    Solve RMP(θ̄, C) to improve the current solution
    (Z^CP, d) ← Solve CP to find a descent direction
while Z^CP > 0 and d is not integer
θ̄ = θ̄ + d
return θ̄
```

Given a current solution $\bar{\theta} \in \mathcal{F}_{SPP}$, let $\mathbf{d}$ be the direction leading to the next solution such that $A\mathbf{d} = \mathbf{0}$. In fact, the set of the decent directions generates the null space of A which could be an infinite cone. Thus, normalization constraints $\mathbf{W}^\top \mathbf{d} = \mathbf{1}$ are added to bound the problem where $\mathbf{W}$ denotes the weight vector. The linear program CP finds, if possible, the combination of columns to obtain a feasible integer descent direction $\mathbf{d}$, i.e., that satisfies the following conditions:

1. $\underline{\mathbf{p}}_r^\top \mathbf{d} > 0$ (improving)
2. $\boldsymbol{\theta} + \alpha \mathbf{d} \in \mathcal{F}_{SPP}, \quad \alpha > 0$ (integer)

6 Solution Method

ICG is a three-stage sequential algorithm, which merges primal approach in a column generation context.

The Algorithm 2 summarizes the major steps of the solution method:

1. The initialization step builds an artificial initial solution (θ^0, π^0). The initial primal solution θ^0 is built such that, in Step 1, each customer i is visited by a single vehicle k that bears a very large cost. In the initial dual solution π^0, each dual value is set to this large cost.
2. The first step starts by solving the subproblems $SP(\pi^t)$. Using the duals π^t corresponding to the current solution θ, positive-reduced cost routes are included in RMP. If no such routes are generated, we stop the algorithm and the best solution found θ^* is returned.
3. In the second step, ISUD solves the RMP to improve the solution. The criterion $minImp$ decides whether the improvement is sufficient or not. If so, neighborhood search is explored around θ^t by invoking a small MIP. This improvement step is iterated until the number of consecutive improvement failures $consFail$ reaches $maxConsFail$.

Algorithm 2: ICG pseudo-code

Parameters: $maxConsFail$, $minImp$.
Initialize : $t \leftarrow 0$; $(\theta, \pi) \leftarrow (\theta^0, \pi^0)$; $consFail \leftarrow 0$
Output : (z^*, θ^*)

repeat
  Step 1: CG
  $\Omega' \leftarrow$ Solve the $SP(\pi^t)$
  if $\Omega' = \varnothing$ **then**
    break
  end
  $\Omega \leftarrow \Omega' \cup \Omega$
  $t \leftarrow t + 1$
  Step 2: RMP
  $(\theta^t, z^t, \pi^t) \leftarrow$ Solve the RMP using ISUD
  if $\frac{z^{t-1} - z^t}{z^{t-1}} \leq minImp$ **then**
    $consFail \leftarrow consFail + 1$
    $\theta^t_{NS} \leftarrow$ search an improved solution around θ^t by solving a restricted MIP
    if $z^t_{NS} > z^t$ **then**
      $\theta^t \leftarrow \theta^t_{NS}$
      $(\theta^t, z^t, \pi^t) \leftarrow$ Resolve RMP using ISUD
    end
  else
    $consFail \leftarrow 0$
  end
  $(z^*, \theta^*) \leftarrow (z^t, \theta^t)$
until $consFail \geq maxConsFail$
return (z^*, θ^*)

Remark 1. The theoretical observations and empirical study made by [17], concluded that there is a strong correlation between the choice of the normalization weights vector ($W^\top \mathbf{d} = \mathbf{1}$) and the descent direction obtained. In practice, we use incompatibility degree δ_j as a weight vector to favor integrality.

7 Experimentation

Through this section, we discuss the results obtained by the ICG algorithm on real-life instances. ICG is compared to a well-know branch-and-price diving heuristic (DH) which is a dual-fractional heuristic based on column generation [22]. DH uses a depth-first search by exploring a single branch of the search tree. At each node, candidate columns for selection are evaluated, and the most fractional variable is set to 1. The process stops when the solution of the master problem is integer. The computing was performed on 7 real-life instances using a C++ implementation, under Linux on workstations with 3.3 GHz Intel Xeon

E3-1226 processors, and 8 GB RAM. The algorithms were implemented using *IBM CPLEX* commercial solver (version 12.4). The SPPRC was solved by dynamic programming using the *Boost* library version 1.54.

7.1 Instance Description

The real-life instances are provided by a major logistics provider. The instances correspond to home appliances' distribution of 7 weekdays and involve from $n = 34$ to $n = 140$ customers. We consider heterogeneous fleet of 6 types of vehicles. The fleet size is unlimited since the company can use external vehicles.

For each day, the order form indicates the customer index, his location, the quantity requested, the delivery deadline, and the allowed time frame. Following several tests, the ICG algorithm was implemented with the following parameters values: $minImp = 0.0025$, $maxConsFail = 5$. The Table 2 shows the computational performance of both ICG and DH. Column 1 indicates the name of the instance ($Name$). Column 2 indicates the number of customers (n) in each instance. For both ICG and DH runs, columns 3 and 7 display the number of iterations ($Iter$). Columns 4 and 8 display the total computing time in seconds ($Time$). Column 6 indicates the total number of integer solutions found ($nb.Sol$). Finally, columns 5 and 9 report the optimality gap in percentage (Gap) between the cost of the best integer solution found and the linear relaxation optimal value.

7.2 Numerical Results

Table 2. Computational results on 7 realistic instances

Instances		ICG				DH		
Name	n	Iter	Time	Gap%	nb.Sol	Iter	Time	Gap%
J2	34	4	0.95	0	5	7	0.23	0
J3	40	7	4.48	0.94	8	8	1.14	1.3
J7	66	9	22.6	1.3	28	33	14.4	1.77
J18	106	7	32	0.14	16	17	31.6	1.3
J25	136	12	515	1.62	51	94	1181.4	2.3
J19	140	9	168	0	20	18	321	0.05
J23	126	6	126	0.09	66	48	804	1.03

One can observe that ICG clearly outperforms DH. Indeed, the primal-based method has successfully obtained a feasible solution for all instances, within a competitive computing time $\in [0.95\,\text{s},\ 515\,\text{s}]$. Optimality gap varies from 0.00% to 1.6%. For DH, the computing time $\in [0.23\,\text{s},\ 1181.4\,\text{s}]$, and optimality gap varies from 0.00% to 1.77%. ICG reduces the DH computing time by a factor of

2.7 on average. ICG performs fewer iterations (at most 12) than DH (between 7 and 94). This can be explained by the ISUD algorithm which generates a large set of columns at each iteration. The Fig. 1 displays the time evolution according to the number of customers. ICG shows a remarkable performance on large instances. In fact, DH time was sped up by a factor of 3.5 on average. The authors [19] also noticed this finding while experimenting ICG on large VCSP and CPP instances.

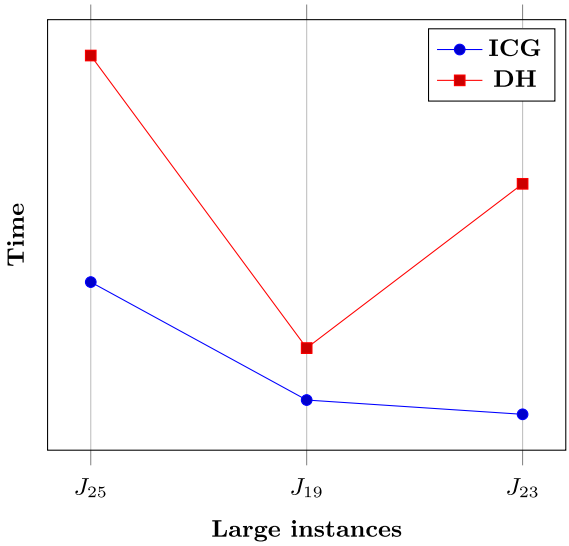

Fig. 1. Comparative computing times on large instances $n = 126, 136, 140$

We easily notice that ICG yields a large number of integer solutions (from 5 to 66), furthermore, ISUD generates a large number of columns. Thus, this important behaviour helps to improve the objective function since the first iterations. We notice that, for all instances solved with ICG algorithm, no branching method has been activated to obtain integer solutions. One recall that branch-and-bound methods could obtain a good solution with a good tuning and a proper branching strategy.

8 Conclusion and Further Work

In this paper, we have experimented for the first time a primal algorithm (ICG) on a complex routing problem. ICG combines the primal algorithm ISUD into column generation framework, and considers a set partitioning formulation. The computational indicators have shown the effectiveness of ICG algorithm while tested on real data reaching 140 customers and a realistic network.

Since the subproblem is a time-consuming, we would like to propose a new intelligent network modelling based on realistic data analysis. We are testing another ICG version that dynamically augments the search space to quickly find integer solutions without any branching recourse.

References

1. Dantzig, G.B., Ramser, J.H.: Manag. Sci. **6**(1), 80 (1959)
2. Clarke, G.U., Wright, J.W.: Oper. Res. **12**(4), 568 (1964)
3. Coelho, L.C., Renaud, J., Laporte, G.: Road-based goods transportation: a survey of real-world applications from 2000 to 2015. Technical report, FSA-2015-007, Québec, Canada (2015)
4. Toth, P., Vigo, D.: Vehicle Routing: Problems, Methods, and Applications, vol. 18. SIAM (2014)
5. Lenstra, J.K., Kan, A.H.G.: Networks **11**(2), 221 (1981)
6. Letchford, A.N., Lodi, A.: Math. Methods Oper. Res. **56**(1), 67 (2002)
7. Vanderbeck, F.: Implementing mixed integer column generation. In: Desaulniers, G., Desrosiers, J., Solomon, M.M. (eds.) Column Generation, pp. 331–358. Springer, Boston (2005). https://doi.org/10.1007/0-387-25486-2_12
8. Lübbecke, M.E., Desrosiers, J.: Oper. Res. **53**(6), 1007 (2005)
9. Irnich, S., Desaulniers, G.: Shortest path problems with resource constraints. In: Desaulniers, G., Desrosiers, J., Solomon, M.M. (eds.) Column Generation, pp. 33–65. Springer, Boston (2005). https://doi.org/10.1007/0-387-25486-2_2
10. Desrochers, M., Soumis, F.: INFOR: Inf. Syst. Oper. Res. **26**(3), 191 (1988)
11. Ben-Israel, A., Charnes, A.: J. Soc. Ind. Appl. Math. **11**(3), 667 (1963)
12. Young, R.D.: Oper. Res. **16**(4), 750 (1968)
13. Thompson, G.L.: Comput. Optim. Appl. **22**(3), 351 (2002)
14. Rönnberg, E., Larsson, T.: Eur. J. Oper. Res. **192**(1), 333 (2009)
15. Zaghrouti, A., Soumis, F., El Hallaoui, I.: Oper. Res. **62**(2), 435 (2014)
16. Elhallaoui, I., Metrane, A., Desaulniers, G., Soumis, F.: INFORMS J. Comput. **23**(4), 569 (2011)
17. Rosat, S., Elhallaoui, I., Soumis, F., Lodi, A.: Integral simplex using decomposition with primal cuts. In: Gudmundsson, J., Katajainen, J. (eds.) SEA 2014. LNCS, vol. 8504, pp. 22–33. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07959-2_3
18. Zaghrouti, A., El Hallaoui, I., Soumis, F.: Annals of Operations Research (2018). https://doi.org/10.1007/s10479-018-2868-1
19. Tahir, A., Desaulniers, G., El Hallaoui, I.: EURO J. Transp. Logist. 1–32 (2018)
20. Feillet, D., Dejax, P., Gendreau, M., Gueguen, C.: Networks **44**(3), 216 (2004)
21. Trubin, V.: Soviet Mathematics Doklady, vol. 10, pp. 1544–1546 (1969)
22. Joncour, C., Michel, S., Sadykov, R., Sverdlov, D., Vanderbeck, F.: Electron. Notes Discret. Math. **36**, 695 (2010)