CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

A Hybrid ALNS Heuristic for the Bid Construction Problem in Transportation Procurement Auctions

Farouk Hammami
Monia Rekik
Leandro C. Coelho

January 2018

CIRRELT-2018-04

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2018-002.

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

A Hybrid ALNS Heuristic for the Bid Construction Problem in Transportation Procurement Auctions

Farouk Hammami*, Monia Rekik, Leandro C. Coelho

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, Université Laval, 2325 de la Terrasse, Québec, Canada G1V 0A6

Abstract. In combinatorial auctions for the procurement of transportation services, bid construction problems (BCPs) are studied since the 1990's but the problem is still open due to its NP-hard nature. Our work addresses the BCP for truckload (TL) operations. During the auction, each carrier has to solve a BCP in order to choose the set of lanes that are the most profitable to bid on. We propose a new formulation for the BCP in the TL context inspired by mathematical models used for some variants of the vehicle routing problem. To solve the BCP, a hybrid heuristic framework is developed. It is based on a destroy-repair principle coupled with local search procedures. A variant integrating an optimization layer is also proposed in which a set packing problem is solved for the solutions obtained during the heuristic search. Computational results show that our heuristic performs very well in terms of CPU time and solution quality when compared to a newly published exact branch-price-and-cut method and to a commercial solver.

Keywords. Bid construction problem, large neighborhood search, combinatorial auctions, truckload transportation, metaheuristic.

Acknowledgements. This project was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 2014-05764 and 2016-04482. This support is greatly acknowledged.

---

* Corresponding author: Farouk.Hammami@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2018

1 Introduction

Auction mechanisms are an important tool to improve service levels and decrease costs in transportation procurement systems [14]. In freight transportation, a billion dollar industry in North America [7], auctioning systems allow shippers to make their transportation requests (contracts) available to the market, and carriers bid on a set of these requests [9]. This is known as the bid construction problem (BCP) in which each carrier determines which transportation requests it is interested to perform and the price asked for serving these requests. Shippers must then determine the winning bids on what is called the winner determination problem [9, 23].

Trucking operations are either Truckload (TL) or Less-Than-Truckload (LTL). In TL operations, the carrier drives directly from the origin location of a shipment to its correspondent destination for a delivery. In LTL operations, shipments consolidation is permitted and the carrier can perform different pickups at different shipments origins before delivery to shipments' destinations [9].

Our paper addresses the BCP for TL transportation services. We consider a long-term contract market assuming long-term commitments (one to three years) between a shipper and a carrier. The trading mechanism is a one-sided sealed bid reverse combinatorial auction in which carriers compete by submitting combinatorial bids on a set of shipping contracts requested by one or more shippers. More specifically, a shipping contract is defined by a pickup location, a destination location and a volume to be shipped directly from the origin to the destination. When selecting the requests to bid on, a carrier is assumed to know the maximum price the shipper is ready to pay for each request. The carrier's objective is thus to optimize its profit by adding a number of new auctioned contracts to its existing transportation activities. Profitable contracts are then combined and submitted in a single package bid enabling the carrier to express its interest on a package of contracts rather than on each contract individually. In auction jargon, this is known as combinatorial bidding. The combinatorial bid construction is known to be an

NP-hard problem [22].

In this paper we focus on: (1) the selection of profitable new contracts to serve, and (2) the allocation of the available vehicle fleet to serve them. Our objective is to propose an efficient solution approach to help a carrier determine the set of profitable auctioned contracts. When added to the carrier existing network, the selected contracts must respect its fleet size and other operating constraints. Our paper does not address the pricing problem. To that end, a method similar to that of [20] can be used to determine the requested price for the submitted bids.

The contributions of this paper are threefold. First, we provide an in depth comparison of the BCP and related routing problems. We show that the BCP shares many features with the *vehicle routing problem with profits* (VRPPs) [4], the *multi-vehicle profitable pickup and delivery problems* (MVPPDPs) [15] and the *team orienteering problem* (TOP) [12]. Second, we propose a hybrid heuristic combining an adaptive large neighborhood search (ALNS) layer with a local search procedure followed by a set packing model that optimally combines partial solutions. Finally, through detailed computational experiments we show that our heuristic remains very competitive when compared to a state-of-the-art exact solution approach. Large instances are further generated and prove that our hybrid heuristic generates good feasible solutions in relatively short times while a commercial solver fails to identify a feasible solution within two days of running time.

The remainder of this paper is organized as follows. Section 2 describes the main recent publications on BCP for transportation procurement auctions. Section 3 formally defines the BCP and proposes a mixed integer linear programming formulation to model it. It also highlights the similarities and differences between our problem and several well-known routing problems. In Section 4, we describe the proposed hybrid heuristic framework. Section 5 discusses the computational performance of the proposed heuristic on sets of instances either reported in the literature or newly generated. Section 6 concludes the paper.

2 Literature review

Lee et al. [17] proposed a non-linear quadratic formulation for the BCP. Their model allows to determine the best bid packages in a combinatorial auction with a homogeneous vehicle fleet. For each request, a price announced by the auctioneer is given. Column generation, Lagrangian relaxation and a decomposition heuristic are used to solve this optimization problem which is then tested on instances with up to 335 requests. The reported computational results show that despite good quality solutions, the average computational time remains very large.

Chang [10] developed a bidding advisor which helps carriers evaluate the most profitable bidding packages within a BCP. The bidding advisor consists in converting the BCP into a synergetic minimum cost network flow problem by estimating the average synergy values between requests. This was solved using a column generation method equipped with a synergetic shortest path algorithm which generates all optimal paths on a synergetic network. The proposed approach only provides approximate solutions since it is based on average synergies approximations.

Buer [6] formulated the BCP with a homogeneous vehicle fleet for LTL requests. Initially, an exact bidding strategy denoted *elementary bundle bid search* (EBBS) is proposed. The exact strategy provides the same output as bidding on each possible requests package. Then, for larger instances, two heuristic approaches are used to identify profitable request combinations: a pairwise synergies clustering (PSC) based on the Clarke and Wright algorithm [13] and a capacitated p-median-based request clustering (CPMC). Reported computational results demonstrate that the EBBS strategy can be used to solve instances with up to 40 requests. The PSC heuristic required 36% of the bundle bids generated by the EBBS in order to achieve 91% of the possible revenue. The CPMC computes just 5% fewer bundle bids than the EBBS and achieves 81% of the possible revenue. None of the three developed approaches take into consideration the carrier's pre-existing requests.

Song and Regan [22] consider requests in a TL context with no assumption of vehicle

fleet capacity and deal with two scenarios: with and without consideration of pre-existing requests. They proposed an optimization-based approximation algorithm in which the BCP is modeled as a set partitioning problem. The objective is to minimize the total empty travel cost where each tour in the solution is then interpreted as a candidate bid. The proposed model does not take into consideration vehicles' time constraints and considers a homogeneous vehicle fleet. The algorithm performance is analysed through a simulation-based experiment. Computational results report near-optimal solutions only for instances with no more than 10 auctioned contracts.

Recently, Rekik et al. [20] consider the BCP with a homogeneous vehicle fleet and taking into account pre-existing requests. The authors consider a route-based formulation of the problem and use a branch-price-and-cut method to solve it. This proposed approach identifies optimal solutions for instances with up to 26 vehicles and 173 contracts. The authors reported that optimal solutions are obtained within a computation time of at most 730 seconds.

3 Problem definition and formulation

We consider a BCP in which a carrier needs to determine the set of the most profitable transportation contracts to bid on while taking into account its existing transportation requests, its fleet capacity and a number of operational constraints. The objective is to build routes covering all existing and possibly new contracts in order to maximize the carrier's total profit. The latter is computed as the difference between the contracts' prices (assumed to be known) and the total transportation cost.

The BCP is formally defined as follows. Let $K = \{1, ..., |K|\}$ denote the set of all contracts. Set K is partitioned into the subset K^n of new contracts and the subset K^e of pre-existing contracts. To each contract $k \in K$ are associated an origin node o_k, a destination node d_k and a price p_k. Sets $O = \{o_k; k \in K\}$ and $D = \{d_k; k \in K\}$ denote, respectively, the set of origin and destination nodes associated with the contracts. Nodes s and s' represent

copies of the carrier's depot accommodating a vehicle set $L = \{1, ..., |L|\}$. The BCP is then defined on a directed graph $G = (V, A)$, where $V = O \cup D \cup \{s, s'\}$ represents the set of $2|K| + 2$ nodes, and A is the set or arcs.

Given the TL context, routes must be constructed such that:

- from the depot s, only contracts' origin nodes can be visited,
- from a destination node $d_k \in D$ associated with contract k, only the end depot s' or an origin node associated with a contract $k' \neq k$ can be visited,
- from an origin node $o_k \in O$ associated with a contract k, only its corresponding destination node d_k can be visited.

Hence, the set of arcs is defined as $A = \{(s, i) : i \in O\} \cup \{(j, i) : j = d_k \in D, i = o_{k'} \in O, k \neq k'\} \cup \{(j, s'), j \in D\} \cup \{(o_k, d_k), k \in K\}$. To each arc $(i, j) \in A$ are associated a travelling time t_{ij} and a travelling cost c_{ij}.

The carrier's fleet is assumed to be heterogeneous. To each vehicle $l \in L$ are associated a fixed usage fee f^l, a maximum route duration T^l and a service time t_i^l (loading/unloading operations) for each node $i \in V$. Each vehicle must start and end its route at the depot.

The BCP described above can be modelled with two sets of variables: a set of binary variables x_{ij}^l, defined for each arc $(i, j) \in A$ and each vehicle $l \in L$, and a set of continuous variables B_i^l, defined for each node $i \in V$ and each vehicle $l \in L$. Variable x_{ij}^l equals one if and only if vehicle l traverses arc (i, j). Continuous variable B_i^l represents the time at which vehicle l starts its service, if any, at node i. The mathematical model, denoted (M_1) can be formulated as follows:

$$(M_1) : \max \sum_{k \in K} \sum_{l \in L} p_k x_{o_k d_k}^l - \sum_{l \in L} \sum_{j \in O} f^l x_{sj}^l - \sum_{l \in L} \sum_{(i,j) \in A} c_{ij} x_{ij}^l \quad (1)$$

$$\text{s.t.} \sum_{l \in L} x^l_{o_k d_k} = 1 \qquad \forall k \in K^e \tag{2}$$

$$\sum_{l \in L} x^l_{o_k d_k} \leq 1 \qquad \forall k \in K^n \tag{3}$$

$$x^l_{o_k d_k} \leq \sum_{j \in O} x^l_{sj} \leq 1 \qquad \forall l \in L, k \in K \tag{4}$$

$$x^l_{o_k d_k} \leq \sum_{i \in D} x^l_{i,s'} \leq 1 \qquad \forall l \in L, k \in K \tag{5}$$

$$\sum_{j:(j,i) \in A} x^l_{ji} - \sum_{j:(i,j) \in A} x^l_{ij} = 0 \qquad \forall l \in L, i \in V \setminus \{s, s'\} \tag{6}$$

$$\sum_{(i,j) \in A} t^l_i x^l_{ij} + \sum_{(i,j) \in A} t_{ij} x^l_{ij} \leq T^l \qquad \forall l \in L \tag{7}$$

$$B^l_j \geq B^l_i + x^l_{ij}(t_{ij} + t^l_i) - T^l(1 - x^l_{ij}) \qquad \forall l \in L, (i,j) \in A, \tag{8}$$

$$B^l_s = 0 \qquad \forall l \in L \tag{9}$$

$$x^l_{ij} \in \{0,1\} \qquad \forall (i,j) \in A, l \in L \tag{10}$$

$$B^l_i \geq 0 \qquad \forall i \in V, l \in L. \tag{11}$$

The objective function (1) maximizes the carrier's profit defined as the difference between the revenues collected from serving the contracts, the fixed costs associated with vehicles' use and the travelling costs. Constraints (2) ensure that all pre-existing contracts are served once. Constraints (3) allow new contracts to be served at most once. Constraints (4) and (5) imply that each route starts from the depot node s and ends at the depot node s'. Flow conservation is ensured via constraints (6). Constraints (7) impose maximum tour length. Constraints (8) eliminate sub-tours: they ensure that a vehicle l traversing arc (i,j) (i.e., $x^l_{ij} = 1$) starts its service at node j later than the time at which it starts its service at node i. In case $x^l_{ij} = 0$, T^l plays the role of a big M and constraints (8) are inactive. Observe that constraints (8) can be lifted by decreasing the value of T^l as follows: if j is a delivery node associated with a contract k, then T^l can be reduced to $(T^l - \min_{j \in D}(t^l_{js'} + t^l_j))$; if $i = o_k$ is an origin node associated with contract k, T^l can be reduced to $(T^l - \min_{o_k \in O}(t^l_{o_k} + t_{o_k,d_k} + t^l_{d_k} + t_{d_k,s'}))$. Constraints (9) impose that each route starts from the depot at time 0. Finally, constraints (10) and (11) define the domain of the decision variables.

Model (M_1) is inspired by arc-based formulations commonly used to model a number of routing problems. Indeed, the BCP we address can be seen as a new variant of the *pickup and delivery problem* (PDP) and more precisely of the *vehicle routing problem with pickups*

and deliveries (VRPPD). In the VRPPD, we consider a fleet of vehicles which aims to satisfy a set of customer requests. Each request must be served once by one vehicle which visits the origin node before the destination node. The BCP in TL procurement presents, however, important differences with respect to the PDP. First, in the BCP, a vehicle must visit the destination node immediately after the origin one of each contract. Second, in the BCP, only pre-existing requests have to be satisfied. The new auctioned ones are not required to if they are not profitable. Third, a price is associated to each auctioned request which is a pickup-delivery operation between two specified nodes. Finally, the objective function consists in maximizing the difference between the sum of collected prices minus the sum of transportation costs (routing + fixed vehicle costs). Recently, Li et al. [18] address what they call the PDP with time windows, profits and reserved requests (PDPTWPR) in an LTL context. They do not consider fleet heterogeneity and their problem is not defined in the TL context.

The BCP is also close to the *orienteering problem* (OP) and the *team orienteering problem* (TOP). The OP, also known as the Selective Travelling Salesperson (STSP) [16], the maximum collection problem [8] or the bank robber problem [5], is a routing problem in which a set nodes is given, each with a profit. In the OP, one does not distinguish an origin node and a destination node: this being the first difference with the BCP. Most importantly, in the OP, the goal is to determine a route limited to a time T_{max} that visits some of the nodes in order to maximize the sum of the collected profits. Thus, the OP does not take into account travel costs. In the BCP, these costs have to be considered and a number of pre-existing contracts must be visited regardless of their profits.

The TOP is the extension of the OP to multiple vehicles [11]. It is also known as the Multiple Tour Maximum Collection Problem (MTMCP) [8]. A relevant TOP variant in the context of the BCP is the Team Orienteering Arc Routing Problem (TOARP) [3]. The TOARP combines both the TOP and Arc Routing Problem with profits (ARPP). Similarly to the BCP, the TOARP has a set of required arcs and a set of profitable arcs that can be traversed [2]. The goal is to maximize the collected profit, traversing all

required arcs, and not exceeding a given time limit.

The TOARP is the most recent ARPP and TOP variant studied in the literature. Only Archetti et al. [3] and Archetti et al. [1] dealt with this problem, proposing an exact and a heuristic solution approach to solve the problem, respectively. Archetti et al. [3] studied the polyhedron associated with the proposed formulation and designed a branch-and-cut algorithm which was able to solve instances with up to 100 vertices, 800 arcs and four vehicles. It is useful to note that the authors used an initial lower bound, which helped them speed up the resolution using the solution provided by the heuristic proposed in Archetti et al. [1].

In a TL context, the BCP can be seen as TOARP since each request k defining an origin node o_k and a destination node d_k can simply be modeled by an arc (o_k, d_k) and each vehicle can serve one request at a time. However, there are two major details that differ between these problems. First, in the TOARP, there is no consideration of arc costs and the available fleet is assumed to be homogeneous. Second, whereas the depot node can be visited from any destination node in the BCP, this is not the case in the TOARP given its arc set definition.

4 Solution approach: hybrid adaptive large neighborhood search

In this section, we describe the hybrid ALNS framework we propose to solve the BCP described in Section 3. ALNS is an extension of the large neighborhood search class of heuristics with an adaptive layer. It adaptively chooses among a number of insertion and removal operators to intensify the search [21]. An ALNS heuristic starts with an initial solution and iteratively destroys and repairs it at each iteration, until a stopping criterion is met. The adaptive layer controls which operator to choose according to its score. At the first iteration, all operators have the same score. Then, the more an operator has

contributed to improve the solution, the larger its score gets and the more likely it is to be selected in the subsequent iterations.

The ALNS heuristic, although inspired by the ones used to solve VRPs, presents a number of new features designed specifically for the BCP. First, our heuristic includes eight removal and three insertion operators. Some of these operators are commonly used for VRP problems while others are newly designed for the BCP. Second, the contracts' removal and insertion rates are not constant but rather adapt to the solution at hand. The third feature we propose is the use of a local search procedure to improve the solution obtained at the end of each iteration. Finally, an optimization layer is added at the end of the ALNS to further improve the solution output by ALNS. In this layer, all the solutions generated by the ALNS heuristic are considered and the corresponding non-dominated routes are incorporated in a set-packing model that is solved with the branch-and-bound procedure implemented in a commercial solver.

The pseudocode for our hybrid heuristic is shown in Algorithm 1 and its main steps are described in details in the following sections. Mainly, Algorithm 1 starts with an initial solution s^0. This initial solution is constructed using a sequential insertion heuristic, described in Section 4.1. At each iteration, the number of new and pre-existing contracts to be removed is determined. Then two removal operators (an operator for the new contracts, R^n, and one for the pre-existing contracts, R^e) and an insertion operator (I) are selected. These operators are selected based on their past performance, which is translated into scores that are updated after a fixed number of iterations, N_{seg}, also referred to as a run segment (j is the index used for a run segment in Algorithm 1). Details on how operators' scores are computed and updated are given in Section 4.2.

A solution s is accepted if it improves the value of the last admissible solution s_{adm}. Otherwise s is accepted following a simulated annealing criterion. Given the last admissible solution s_{adm}, current solution s is accepted with a probability $e^{\frac{f(s)-f(s_{adm})}{T}}$ where $f(\cdot)$ denotes the objective value of a solution and $T \geq 0$ is the current temperature. The algorithm starts by initializing the temperature T to a predefined value T_0. At the end

of each iteration this temperature is decreased by a factor c, where $c \in]0,1[$ is a fixed cooling rate. It should be mentioned that a large value of T helps to diversify the search while a small value of T is a mean of intensification. A solution s_{adm} is modified into a solution s by means of several destroy and repair operators, described in Section 4.3.

When a solution is accepted and improves the last admissible one, a local search procedure is performed to try to further improve it. Three local search procedures are considered. They are explained in Section 4.4.

The ALNS procedure stops when the number of iterations reaches a pre-specified value $iter_{max}$. Then a set packing model including all candidate and non-dominated routes identified through the ALNS iterations is solved with the branch-and-bound procedure of CPLEX within a fixed time-limit. The set packing-based model is presented in Section 4.5.

4.1 Initial solution

According to Li et al. [18], the quality of the initial solution can have an important impact on the final output of the ALNS heuristic. Consequently, we develop a sequential insertion heuristic (SIH) which prioritizes the satisfaction of pre-existing contracts in set K^e and maximizes the exploitation of the fleet by inserting as many new contracts from the set K^n as possible. The structure of SIH is depicted in Algorithm 2. Overall, we assign pre-existing contracts to routes so that the number of pre-existing contracts in a vehicle route is maximized. Then, if there are vehicles that are still available (they have not been assigned to any pre-existing contract), the same approach is used for new contracts and available vehicles.

In Algorithm 2, vehicles $l \in L$ are sorted in an ascending order with respect to the ratio of their maximum duration length T_l and their fixed cost f_l. This is done so that vehicles/routes offering large service times and small fixed costs are firstly considered. Pre-existing contracts are the first to be treated. For each selected vehicle $l \in L$, pre-existing

Algorithm 1 Structure of the hybrid ALNS heuristic for the BCP

Construct initial solution s
$s_{best} \leftarrow s$, $s_{adm} \leftarrow s$, $T \leftarrow T_0$
Initialize the scores of the repair and removal operators
while $iter < iter_{max}$ **do**
  $j \leftarrow 0$
  while $j < Nseg$ **do**
    $s \leftarrow s_{adm}$
    Generate q_n and q_e
    Select 2 removal one insertion operators: R_n, R_e and I
    Remove q_n new contracts from s using R_n
    Remove q_e pre-existing contracts from s using R_e
    Insert the pre-existing then new contracts in s using I
    Generate a random number $\delta \in]0, 1[$
    if $\left(f(s) > f(s_{adm}) \text{ or } \delta \leq e^{\frac{f(s)-f(s_{adm})}{T}}\right)$ and $|K^e|_{removed} = 0$ **then**
      $s_{adm} \leftarrow s$
      if $f(s) > f(s_{best})$ **then**
        Apply local search procedures on s
        $s_{best} \leftarrow s$
      end if
    end if
    if $T \leq T_{min}$ **then**
      $T \leftarrow T_0$
    end if
    $T \leftarrow T * c$, $iter \leftarrow iter + 1$, $j \leftarrow j + 1$
  end while
  Update scores of the ALNS operators
end while
Generate and solve a set packing problem
Update and return s_{best}

Algorithm 2 Sequential Insertion Heuristic (SIH)

1: $L^0 \leftarrow L$, $\tilde{L} \leftarrow \emptyset$
2: **while** $|L| > 0$ **do**
3: $\quad \tilde{K^e} \leftarrow \emptyset$
4: $\quad d \leftarrow s$, $\tilde{l} \leftarrow \arg\max_{l \in L} \left(\frac{T_l}{f_l}\right)$
5: $\quad$ **while** $|K^e \backslash \tilde{K^e}| > 0$ **do**
6: $\quad\quad \tilde{k} \leftarrow \arg\min_{k \in K^e \backslash \tilde{K^e}} (t_{d,o_k})$
7: $\quad\quad$ **if** inserting $\tilde{k}$ in route $\tilde{l}$ respects all constraints **then**
8: $\quad\quad\quad$ Insert $\tilde{k}$ in route $\tilde{l}$, $d \leftarrow d_{\tilde{k}}$, $K^e \leftarrow K^e \backslash \{\tilde{k}\}$, $\tilde{L} \leftarrow \tilde{L} \cup \{\tilde{l}\}$
9: $\quad\quad$ **else**
10: $\quad\quad\quad \tilde{K^e} \leftarrow \tilde{K^e} \cup \{\tilde{k}\}$
11: $\quad\quad$ **end if**
12: $\quad$ **end while**
13: $\quad L \leftarrow L \backslash \{\tilde{l}\}$
14: **end while**
15: **if** $|L^0 \backslash \tilde{L}| > 0$ **then**
16: $\quad K^e \leftarrow K^n$, $L \leftarrow L^0 \backslash \tilde{L}$. Go to Step 2.
17: **end if**

contracts are gradually added to the route l by choosing the contract that is closest to its predecessor in the route, starting with the depot s. A contract is included in a vehicle route if the maximum time length constraint is respected. Otherwise, the contract is temporarily put in set $\tilde{K^e}$ to be treated in the following iterations. Once all pre-existing contracts are treated (line 14), the process is restarted for the set of new contracts and the set of vehicles not used for serving the pre-existing contracts. Observe that it may happen that Algorithm 2 fails in assigning all pre-exiting contracts to vehicles. This is handled by the ALNS heuristic, as will be depicted in the following sections, through its removal and insertion operators that give priority to feasible solutions.

4.2 Adaptive scores and removal rate adjustment

At the beginning of each iteration, two random numbers, denoted respectively q_n and q_e, are computed to control how many new and pre-existing contracts, respectively, to remove from the current solution. The value of q_n (q_e) depends on the number of new

(pre-existing), contracts covered by the solution of the current iteration. Both values are generated following a semi-triangular distribution which gives high probabilities to low numbers, and low probabilities to higher ones.

At each iteration, our heuristic selects two removal operators (one for the existing and one for the new contracts) and one insertion operator. Such a combination of operators is chosen based on the operators' past performance: operators having lower scores have lower probability of being selected. As suggested in [21], each operator i is assigned a score $\pi_{i,j}$ in each run segment j (recall that a segment is a number of consecutive iterations). This score is set to 0 at the beginning of the first run segment. Then, at the end of the run segment j, the score of operator i is updated as follows:

$$\pi_{i,j+1} = \lambda \frac{\bar{\pi}_{i,j}}{a_i} + (1 - \lambda)\pi_{i,j}, \tag{12}$$

where a_i is the number of times operator i has been used in segment j, $\bar{\pi}_{i,j}$ is the observed score of operator i for run segment j, and $\lambda \in [0, 1]$ is a decay parameter which adjusts the weights sensitivities to the operators' performances. The observed score $\bar{\pi}_{i,j}$ is obtained from the initial score $\pi_{i,j}$ based on how much operator i played a role in improving the solution at each iteration of the run segment. More precisely, each time operator i is used within run segment j, $\pi_{i,j}$ is incremented by a constant parameter ρ which may take four possible values as follows:

$$\rho = \begin{cases} \rho_1 & \text{if the new solution is a new best,} \\ \rho_2 & \text{if the new solution is better than the last one accepted,} \\ \rho_3 & \text{if the new solution does not improve the last accepted solution,} \\ \rho_4 & \text{if the new solution is infeasible,} \end{cases}$$

where $\rho_1 > \rho_2 > \rho_3 > \rho_4 \geq 0$.

The adaptive mechanism of the ALNS uses the roulette wheel selection to independently choose a removal and an insertion operator based on their scores. Assuming we have n

operators, with operator i having a score $\pi_{i,j}$ at the beginning of run segment j, then the probability to select operator i in run segment j is computed as: $\frac{\pi_{i,j}}{\sum_{i'=0}^{n} \pi_{i',j}}$.

4.3 Removal and insertion operators

The ALNS is often applied to vehicle routing and pickup and delivery problems in which all nodes must be visited. The BCP has the particularity of having pre-existing contracts to satisfy and new contracts with no obligation to be served. Hence, it is necessary to make modifications on the removal and insertion operators proposed in the literature.

4.3.1 Removal operators

1. *Lowest net profit*: the purpose of this operator is to remove q new contracts with the smallest net profit. We define the net profit of contract $k \in K^n$ as the difference between its price p_k and its associated arc cost $c_{o_k d_k}$. To avoid cycling, contracts to be removed are selected using a roulette wheel which gives high probabilities to contracts having the lowest net profits.

2. *Lowest net profit per time unit*: this operator is similar to the *lowest net profit*, except that contracts are selected based on the ratio of their net profit $p_k - c_{o_k d_k}$ and their corresponding travel time $t_{o_k d_k}$. Contracts with the lowest ratios have higher probabilities to be removed.

3. *Random removal*: contracts are randomly selected and removed from the current solution. The aim of using this operator is to diversify the search.

4. *Smallest increment after insertion*: this operator works on both new and pre-existing contracts. For each contract in the current solution, we determine the difference between the objective function values before and after the contract insertion as computed in the insertion stage. Contracts yielding the smallest increment in the objective function after their insertion have a higher chance of being removed.

5. *Smallest decrement after removal*: this operator is used for both pre-existing and new contracts and is inspired from the general framework proposed by Pisinger and Ropke [19]. Given a current solution s, we define $f(s)$ as the value of the objective function in s and s^{-k} as the solution obtained from s by removing contract k. Contracts are sorted in an ascending order with respect to the decrement in the objective function value they induce when dropped off (computed as $f(s) - f(s^{-k})$). The selection of the q contracts to be removed is done randomly using the roulette wheel selection ensuring that contracts yielding a small decrement in the objective function, value, when dropped off, have larger probability to be removed.

6. *Largest saving in travelling cost*: the aim of this operator is to remove contracts resulting in large empty travel costs. Given a solution s, we define $E(s)$ as the cost associated with its empty moves. For each contract k covered by s, we compute the difference in empty travelling costs resulting from removing k, $E(s) - E(s^{-k})$. A roulette wheel is then used which gives larger probability to remove the contracts inducing larger decrements in empty move costs when dropped off.

7. *Largest residual time*: for each vehicle route l in a solution s, we compute its residual time, $R(s, l)$ as the difference between its maximum permitted duration T^l and the total time required to serve all the covered contracts. The total residual time associated with s, $R(s)$, corresponds to the sum of the residual times of the routes it includes. For each contract k covered by s, we compute the total residual time resulting from removing k, that is, $R(s^{-k})$. A roulette wheel is then used which gives larger probability to remove the contracts inducing the largest residual time when removed.

8. *Route removal*: this operator selects the vehicle route l with the smallest ratio $\frac{T_l}{f_l}$, that is, a vehicle inducing a large fixed cost and offering lower chance to include multiple contracts because of its restrictive maximum duration. Observe that for this operator, all the contracts covered by the selected vehicle are removed.

4.3.2 Insertion operators

1. *Sequential greedy insertion*: this operator inserts contracts in the most profitable position, one at a time. Contracts, either pre-existing or new, are tested following the "First Removed First Inserted" (FRFI) rule. Pre-existing contracts are inserted in the most "profitable" position even if the resulting solution deteriorates the objective function value. New contracts are inserted only if their insertion improves the current solution.

2. *Random greedy insertion*: this operator is similar to the sequential greedy insertion one except that the contacts evaluation is done randomly rather than with the FRFI rule. This operator is implemented in order to diversify the order of the contracts' insertion.

3. *Regret insertion*: the noticeable disadvantage of the greedy insertion operators is that they only consider the improvement incurred by the insertion of one contract in its best position without taking into account the impact of such an insertion on subsequent decisions. The regret insertion operator tries to circumvent this problem [21, 19]. For each contract $k \in K$, let Δf_k^g denote the variation in the objective function value resulting from inserting the contract k in its gth best position. That is, $\Delta f_k^g \geq \Delta f_k^{g\prime}$ for $g \leq g\prime$. In our case, we consider a *regret-2* operator (i.e., $g = 2$). At each iteration, the contract k^* to be first inserted is such that

$$k^* := \arg\max_{k \in K} (\Delta f_k^1 - \Delta f_k^2).$$

Contract k^* is inserted in the best position and the process iterates until the required number of contracts is inserted.

4.4 Local search procedures

Three local search procedures are sequentially applied to the solution after the insertion step to try to improve it. First, we start with pre-existing contracts by trying to relocate them in another position within the available routes. Second, we randomly select a new contract and we test two possibilities: we remove the contract if this improves the solution, otherwise we relocate this contract to another position. The last move swaps one pre-existing and one new contract randomly selected, if this improves the solution.

4.5 Set packing layer

Let R denote the set of all non-dominated routes generated during the ALNS iterations and p_r the net profit associated with route $r \in R$. Observe that route r^1 dominates route r^2 if they serve the same set of contracts and $p_{r^1} > p_{r^2}$. A set packing problem (SPP) is then considered to select the subset of routes that maximizes the total profit. To this end, a binary variable y_r is defined for each $r \in R$. Variable y_r equals 1 if route r is chosen, and $y_r = 0$ otherwise. We also define a constant parameter for each route $r \in R$ and each contract $k \in K$ such that $a_{rk} = 1$ if route r serves contract k, and 0 otherwise. The problem is formulated as follows:

$$\max \quad \sum_{r \in R} p_r y_r \tag{13}$$

$$\text{s.t.} \quad \sum_{r \in R} y_r a_{rk} = 1 \qquad \forall k \in K^e \tag{14}$$

$$\sum_{r \in R} y_r a_{rk} \leq 1 \qquad \forall k \in K^n \tag{15}$$

$$\sum_{r \in R} y_r \leq |L| \tag{16}$$

$$y_r \in \{0, 1\} \qquad \forall r \in R. \tag{17}$$

The objective function (13) maximizes the total profit. Constraints (14) ensure that all pre-existing contracts are served once by selected routes. Constraints (15) allow new contracts to be served at most once. Constraint (16) limits the number of selected routes

to the size of the available vehicle fleet. Finally, constraints (17) are the binary constraints on the y variables.

5 Computational results

The proposed hybrid ALNS heuristic is coded in Java and the experiments are conducted on a PC equipped with 64 Gigabyte RAM and Intel(R) Core(TM) i7-6700 processor clocked at 3.40 GHz running Ubuntu 16.04.2 LTS. The mathematical models are solved using the Optimization Programming Language (OPL) and the CPLEX 12.7.1 solver.

5.1 Problem tests and parameters setting

Our experimental study considers two sets of instances: sets I and II. Set I consists of the 20 generated proposed in Rekik et al. [20]. All the instances were solved to optimality by the branch-price-and-cut method presented therein. These instances were obtained by varying the carriers' pre-exiting contracts, the new auctioned contracts and the vehicles' fixed costs. Contracts are defined from a list of origin and destination locations corresponding to existing cities in Canada and the USA: the larger the number of cities is, the larger the covered territories are. All instances consider a homogeneous fleet. Each vehicle is assumed to have the capacity to serve any contract. These instances contain up to 173 contracts and we refer the reader to Rekik et al. [20] for details on instances generation. Set II includes 14 new instances that we generate following the same principle as in [20] in order to test the performance of our heuristic on larger instances, containing up to 500 contracts.

In Table 1 we provide a description of the instances characteristics. We report for each instance, the number of cities from which contracts' origins and destinations locations were generated (N), the maximum tour length (T_{max}), the fleet size ($|L|$), the fixed vehicle cost (f_l), the number of new auctioned contracts ($|K^n|$), and the number of pre-existing

contracts ($|K^e|$).

To tune the ALNS parameters, we have run several preliminary tests. We observed that parameters q_n and q_e have a significant impact on the average computational time. The initial temperature T_0 has a significant impact on the solution quality: the larger the value of T_0 is, the longer is required to find good solutions.

In our case, we randomly generate q_n within the interval $[1, 0.4|K_s^n|]$ and q_e within $[1, 0.3|K_s^e|]$, where K_s^n and K_s^e denote respectively the number of new and pre-existing contracts covered by the current solution s. Hence, the values of parameters q_n and q_e dynamically adapt to solutions obtained through the algorithm iterations. To set the final values for the remaining ALNS parameters, we consider different combinations. Each combination is obtained by varying the value of each parameter, one at a time. We ran our heuristic 20 times for each parameter combination. The combination yielding the best average performance in terms of solution quality and computational time was chosen. The values of the parameters of the best combination are shown in Table 2. Observe that for the maximum number of iterations, we used two values ($iter_{max}$ = 50,000 and 250,000) to highlight the trade-off between solution quality and computing time.

To solve the SPP, we set a time limit of 1,800 (3,600) seconds for the instances in set I including fewer than 100 (200) contracts. For the instances in set II, the time limit is set to 7,200 seconds.

Observe that, contrarily to instances in set I, no optimal solution is known for the new instances in set II. Hence, to evaluate the quality of the solutions obtained with our heuristic for these new instances, we solve model ($M1$) described in Section 3 with the branch-and-bound procedure of CPLEX with a time limit of either 12, 24 or 48 hours so that *at least* an upper bound is provided by CPLEX.

Table 1: Instances characteristics

Sets	**Instance**	**N**	T_{max}	$\|L\|$	f_l	$\|K^n\|$	$\|K^e\|$
Set I	1	6	1500	2	500	24	12
	2	6	1500	2	1000	24	12
	3	6	1500	4	500	24	12
	4	6	1500	4	1000	24	12
	5	8	1500	4	500	35	18
	6	8	1500	4	1000	35	18
	7	8	1500	7	500	35	18
	8	8	1500	7	1000	35	18
	9	10	1500	7	500	48	25
	10	10	1500	7	1000	48	25
	11	10	1500	10	500	48	25
	12	10	1500	10	1000	48	25
	13	12	1500	10	500	76	28
	14	12	1500	10	1000	76	28
	15	12	1500	15	500	76	28
	16	12	1500	15	1000	76	28
	17	15	1500	18	500	131	42
	18	15	1500	18	1000	131	42
	19	15	1500	26	500	131	42
	20	15	1500	26	1000	131	42
Set II	21	16	1800	28	500	140	33
	22	16	1800	28	500	140	44
	23	16	1800	28	500	186	44
	24	16	1800	36	500	186	44
	25	17	2000	24	500	210	52
	26	17	2000	24	1000	210	52
	27	17	2000	34	500	210	52
	28	17	2000	34	1000	210	52
	29	17	2000	34	1000	248	52
	30	20	3200	24	500	336	64
	31	20	3200	24	1000	336	64
	32	20	3200	30	500	336	64
	33	20	3200	30	1000	336	64
	34	20	3200	40	1000	400	100

Table 2: Parameter tuning of the ALNS heuristic.

Parameter	Signification	Best value
$iter_{max}$	Maximum number of iterations	50,000 and 250,000
q_n	Number of new contracts to remove	$q_n \in [1, 0.4\|K_{n_s}\|]$
q_e	Number of pre-existing contracts to remove	$q_e \in [1, 0.3\|K_{e_s}\|]$
T_0	Simulated annealing initial temperature	1000
T_{min}	Simulated annealing minimum temperature	0.001
c	Cooling rate for the simulated annealing	0.99992
ρ_1	Operator score increment case 1	20
ρ_2	Operator score increment case 2	4
ρ_3	Operator score increment case 3	1
ρ_4	Operator score increment case 4	0
λ	Decay parameter to adjust operators weights sensitivities	0.999
$Nseg$	Size of a run segment before adjusting operators weights	500

5.2 Results for instances from set I

As already mentioned, the instances in set I were proposed by Rekik et al. [20] and were solved to optimality with their branch-price-and-cut method. Table 3 reports, for each instance, the optimal objective value, and the CPU time reported in Rekik et al. [20]. It also shows the optimality gap yielded by the ALNS heuristic with 50,000 iterations (column Gap_H), the iteration number at which the best solution was obtained (column $best_{iter}$) and the corresponding CPU time ($best_{time}$), and the total CPU time required to execute the 50,000 iterations. The columns under ALNS 50k+SPP display the optimality gap and the total CPU time when the set packing layer is considered. Similar results are reported when the maximum number of iterations is set to 250,000 under the columns ALNS 250k and ALNS 250k+SPP.

The results of Table 3 show that for small-sized instances (instances 1 to 8), our heuristic was able to find the optimal solution without integrating the SPP (columns ALNS 50k and ALNS 250k). For instances 1 to 4, the optimal solution was identified at the first iterations (91 iterations for the worst case) within very short computing times (0.21 seconds for the worst case). Hence, fixing the maximum number of iterations to 50,000 was largely sufficient for these instances. For instances 5 to 8, more iterations were required (up to 210,400 for instance 5) to find the optimal solution, without the SPP layer, but still within

Table 3: Computational results for instances in set I

Instance	Optimal	CPU(s)	ALNS 50k				ALNS 50k+SPP		ALNS 250k				ALNS 250k+SPP	
			Gap_H (%)	$best_{iter}$	$best_{time}(s)$	$CPU(s)$	Gap_{H+SPP} (%)	$CPU(s)$	Gap_H (%)	$best_{iter}$	$best_{time}(s)$	$CPU(s)$	Gap_{H+SPP} (%)	$CPU(s)$
1	13377	24.4	0.00	91	0.21	7	0.00	11	0.00	91	0.21	45	0.00	50
2	12377	20.4	0.00	75	0.15	7	0.00	11	0.00	75	0.15	45	0.00	50
3	22285	13.7	0.00	3	0.04	8	0.00	25	0.00	11	0.06	47	0.00	85
4	20285	5.2	0.00	39	0.09	8	0.00	24	0.00	39	0.09	47	0.00	85
5	21700	257.5	0.16	2860	2.49	20	0.00	24	0.00	210400	82.00	100	0.00	115
6	19700	354.3	0.17	2547	2.36	20	0.00	24	0.00	199583	70.00	100	0.00	115
7	33734	8.7	0.26	38362	18.3	25	0.00	50	0.00	38362	18.3	120	0.00	155
8	30234	8.0	0.29	29555	18.00	27	0.00	53	0.00	29555	18.00	121	0.00	159
9	32992	61.8	0.19	43149	36.00	50	0.00	55	0.12	229958	158.00	170	0.00	177
10	29492	55.0	0.26	28341	21.9	40	0.00	47	0.26	28341	21.9	174	0.00	178
11	42634	14.3	1.62	49702	52.4	53	0.57	1853	1.50	206510	170.00	211	0.00	2011
12	37634	17.8	1.10	24310	28.29	57	0.02	1857	1.10	24310	28.29	219	0.00	2019
13	45363	115.6	1.90	49992	90.3	90	0.29	1890	1.04	249721	409.44	410	0.29	2210
14	40363	144.1	1.65	44444	98.00	114	0.33	1914	1.17	227981	378.00	418	0.33	2218
15	61022	202.8	2.25	40766	82.47	99	0.22	1030	1.76	150283	306.00	487	0.01	4087
16	53522	134.0	2.12	45302	100.3	114	0.25	1002	2.06	213590	501.00	549	0.01	4149
17	71611	282.3	2.93	46705	250.00	270	0.01	295	1.92	66639	339.00	1140	0.00	1290
18	62611	283.9	2.96	47583	265.00	275	0.01	330	2.07	228640	1087.00	1159	0.00	1269
19	97750	730.2	2.75	49522	368.00	370	0.07	1550	1.85	126305	955.00	1487	0.00	5087
20	84750	706.4	2.24	48319	351.00	380	0.10	1104	2.12	247999	1563.00	1598	0.00	5200
Average		172.02	1.14	29583	89.27	101.70	0.09	657.45	0.85	123920	305.27	432.35	0.03	1535.45

relatively short computing times (between 18 and 82 seconds). Observe, however, that for instances 5 to 8, integrating the SPP layer to the variant where $iter_{max}$ = 50,000 enables identifying optimal solutions within smaller CPU times, on average, when compared to ALNS 250k. For all the small-sized instances the total CPU time is relatively small: 15.25 seconds on average for ALNS 50k, and 78.13 on average for ALNS 250k.

For the remaining instances (instances 9 to 20), the ALNS heuristic with no SPP layer fails to find the optimal solutions even when $iter_{max}$ = 250,000. It results in an average optimality gap of 1.83% for ALNS 50k and 1.41% for ALNS 250k. This slight improvement in the optimality gap comes however at the expense of larger CPU times: as can be deduced from columns $best_{time}$ and CPU, the ALNS variant with 50,000 iterations requires 145.3 seconds on average to find the best solution and 159.3 seconds on average to run all the iterations. On a counterpart, ALNS 250k needs an average time of 493.5 seconds to identify the best solution and 668.5 seconds on average to run the 250,000 iterations.

Adding the SPP layer for instances 9 to 20 enables reaching optimal solutions for eight instances and very near-optimal solutions (an optimality gap between 0.01% and 0.33%) for the remaining four instances. Except for instances 9 and 10, 250,000 iterations were required by the ALNS to perform as well. Given that the SPP layer solves a MIP with a large number of decision variables, the total CPU time increases reaching 5,200 seconds for instance 20.

Overall, our experimental results demonstrate that the proposed heuristic with its different variants is able to identify optimal solutions for 16 instances over the 20 of set I. The total CPU time is comparable with that required by the exact method proposed in [20] for half of the instances (instances 1 to 10). For the remaining instances, there is a trade-off to be managed between solutions quality and CPU times when deciding on the heuristic variant to be applied. Indeed, good quality solutions with an optimality gap lower than 2.96% could be obtained with ALNS 50k within computing times either comparable (instances 13, 14, 16, 17 and 18), larger but still small (instances 11 and 12), or much smaller (instances 19 and 20) than the exact method of [20].

In [20], the proposed exact method was tested on instances including up to 131 new contracts, 42 exiting contracts, that are generated from 15 locations. The results reported therein show that the larger these values get, the more difficult is the instance to be solved to optimality. In the following section, we aim to test the performance of our heuristic (with its different variants) on a new set of instances (set II) including up to 400 new contracts, 100 existing ones and that are generated from 20 cities. Recall that we use the same instances generator as in [20]. So, the larger the number of cities is, the larger the covered territories are and the more constrained the problem becomes with respect to the maximum duration restriction.

5.3 Results for instances from set II

In order to test the performance of our approach, and given that no solutions are known for these newly generated instances, we compare the results obtained for the different variants of the ALNS to that obtained with the branch-and-bound procedure of CPLEX applied to model $M1$. Observe that CPLEX was not able to identify any feasible solution for all the instances in Set II within the time limit, set to 12, 24 and 48 hours, respectively, for instances 21-24, 25-31, and 32-34, respectively. Hence, we display in Table 4 the best dual bound (*Bound*) reported by CPLEX. The gap displayed under the columns "ALNS 50k","ALNS 50k + SPP", "ALNS 250k" and "ALNS 250k + SPP" represents thus the

relative gap with respect to the CPLEX bound. One should note that CPLEX was unable to run instance 34 given its large size (more than 10 million variables).

Table 4: Computational results for instances in set II

Instance	CPLEX		ALNS 50k				ALNS 50k+SPP		ALNS 250k				ALNS 250k+SPP	
	$Bound$	$CPU(s)$	$Gap_{UB}(\%)$	$best_{iter}$	$best_{time}(s)$	$CPU(s)$	$Gap_{UB}(\%)$	$CPU(s)$	$Gap_{UB}(\%)$	$best_{iter}$	$best_{time}(s)$	$CPU(s)$	$Gap_{UB}(\%)$	$CPU(s)$
21	145984	43200	15.63	25345	204	437	13.45	7637	15.34	132527	1114	2428	13.06	9628
22	149373	43200	16.04	40975	732	877	13.21	8077	13.11	248544	4187	4240	12.85	11440
23	158625	43200	14.63	22814	387	756	12.77	7956	12.72	244315	3708	3901	12.66	11101
24	193797	43200	14.57	39949	825	950	13.92	8150	14.03	89036	1700	4900	13.82	12100
25	149705	86400	12.02	46165	860	920	10.43	8120	10.42	247689	3967	4180	10.20	11380
26	149700	86400	20.63	49210	897	910	18.47	8115	18.76	237912	3755	4110	18.21	11319
27	200930	86400	14.17	44724	1070	1129	12.70	8329	14.14	249673	5211	5271	12.05	12471
28	200430	86400	22.76	49877	1090	1096	20.58	8296	22.41	242501	5101	5166	20.31	12366
29	188260	86400	25.26	49815	1980	1991	24.34	9191	24.52	76935	2935	9300	23.91	16500
30	222521	86400	13.73	17328	875	2600	13.48	9800	13.37	92784	4621	13300	13.11	20500
31	222021	86400	18.96	17268	856	2622	18.87	9822	18.69	95073	4675	13150	18.32	20350
32	272379	172800	14.69	38418	1670	2500	14.40	9700	14.40	221019	10990	12520	13.95	19720
33	271879	172800	20.10	49311	2400	2500	19.49	9700	19.46	220986	10987	12600	19.30	19800
34	-	-	-	37892	4720	6150	-	13350	-	218530	27850	-	-	37600
		Average	17.17	36782	989	1441	15.85	8642	16.26	193505	5001	7147	15.52	14348

The results of Table 4 show that unlike CPLEX, our heuristic was able to identify feasible solutions for all instances with its simplest variant, ALNS 50k. Increasing the number of iterations to 250,000 helps improve the solution quality for almost all instances but results in an important increase in solution times. As expected, CPU times are further increased when the SPP layer is incorporated. But once again, applying the SPP model results in better feasible solutions with an average gap of 15.85% when $iter_{max} = 50{,}000$ and of 15.52% when $iter_{max} = 250{,}000$. Observe that the upper bound obtained with CPLEX may still be of poor quality. So the reported gaps may not really reflect the effective quality of the solutions of our heuristic.

6 Conclusion

In this paper, we address the bid construction problem (BCP), a decisional problem arising in transportation procurement auctions. We propose a new formulation for the problem as a variant of a pickup and delivery problem. We propose a hybrid ALNS heuristic including new features adapted to the BCP. Two main variants of ALNS are developed and tested: the first variant is inspired by common ALNS heuristics but considers new insertion and removal operators specific to the BCP as well as other dynamic parameters adjustments.

The second variant incorporates a set packing model at the end of the ALNS iterations that considers all the promising solutions obtained during the ALNS search. Our results show that our heuristic with its different variants performs very well for all the instances. There is however a trade-off between solution quality and solution times for the variant choice.

We have addressed one important aspect of the BCP which is the new contracts selection. However, a bid should also include an ask price for the package of new contracts selected by the heuristic. In this paper, we assumed that the maximum price that could be received by a carrier winning a new contract is known. Our objective was then to select new contracts in order to maximize the carrier' profit based on the new contracts maximum prices. A future research avenue is to address the BCP with stochastic prices. This constitutes a challenging problem hardly addressed in the literature.

References

[1] Archetti, C., Corberán, Á., Plana, I., Sanchis, J. M., Speranza, M. G., 2015. A matheuristic for the team orienteering arc routing problem. European Journal of Operational Research 245 (2), 392–401.

[2] Archetti, C., Speranza, M. G., 2015. Arc routing problems with profits. In: Laporte, G., Corberán, Á. (Eds.), Arc Routing: Problems, Methods, and Applications. Vol. 20. Society for Industrial and Applied Mathematics, Philadelphia, PA, pp. 281–299.

[3] Archetti, C., Speranza, M. G., Corberán, Á., Sanchis, J. M., Plana, I., 2013. The team orienteering arc routing problem. Transportation Science 48 (3), 442–457.

[4] Archetti, C., Speranza, M. G., Vigo, D., 2014. Vehicle routing problems with profits. In: Vigo, D., Toth, P. (Eds.), Vehicle Routing: Problems, Methods, and Applications. Vol. 18 of Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia, PA, pp. 273–297.

[5] Arkin, E. M., Mitchell, J. S., Narasimhan, G., 1998. Resource-constrained geometric network optimization. In: Proceedings of the Fourteenth Annual Symposium on Computational Geometry. ACM, pp. 307–316.

[6] Buer, T., 2014. An exact and two heuristic strategies for truthful bidding in combinatorial transport auctions. arXiv preprint arXiv:1406.1928.

[7] Bureau of Transportation Statistics, April 2017. National transportation statistics. URL `https://www.bts.gov/product/national-transportation-statistics`

[8] Butt, S. E., Cavalier, T. M., 1994. A heuristic for the multiple tour maximum collection problem. Computers & Operations Research 21 (1), 101–111.

[9] Caplice, C., Sheffi, Y., 2006. Combinatorial auctions for truckload transportation. Combinatorial Auctions 21, 539–571.

[10] Chang, T.-S., 2009. Decision support for truckload carriers in one-shot combinatorial auctions. Transportation Research Part B: Methodological 43 (5), 522–541.

[11] Chao, I.-M., Golden, B. L., Wasil, E. A., 1996. A fast and effective heuristic for the orienteering problem. European Journal of Operational Research 88 (3), 475–489.

[12] Chao, I.-M., Golden, B. L., Wasil, E. A., 1996. The team orienteering problem. European Journal of Operational Research 88 (3), 464–474.

[13] Clarke, G., Wright, J. W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12 (4), 568–581.

[14] Gansterer, M., Hartl, R. F., 2016. Request evaluation strategies for carriers in auction-based collaborations. OR Spectrum 38 (1), 3–23.

[15] Gansterer, M., Küçüktepe, M., Hartl, R. F., 2017. The multi-vehicle profitable pickup and delivery problem. OR Spectrum 39 (1), 303–319.

[16] Laporte, G., Martello, S., 1990. The selective travelling salesman problem. Discrete Applied Mathematics 26 (2-3), 193–207.

[17] Lee, C.-G., Kwon, R. H., Zhong, M., 2007. A carrier's optimal bid generation problem in combinatorial auctions for transportation procurement. Transportation Research Part E: Logistics and Transportation Review 43, 173–191.

[18] Li, Y., Chen, H., Prins, C., 2016. Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. European Journal of Operational Research 252 (1), 27–38.

[19] Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Computers & Operations Research 34 (8), 2403–2435.

[20] Rekik, M., Desaulniers, G., Saddoune, M., Elhallaoui, I., 2017. An exact solution approach for bid construction in truckload transportation auction with side constraints. Tech. Rep. G-2017-51, GERAD, Montreal, Canada.

[21] Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40 (4), 455–472.

[22] Song, J., Regan, A., 2005. Approximation algorithms for the bid construction problem in combinatorial auctions for the procurement of freight transportation contracts. Transportation Research Part B: Methodological 39 (10), 914–933.

[23] Triki, C., Oprea, S., Beraldi, P., Crainic, T. G., 2014. The stochastic bid generation problem in combinatorial transportation auctions. European Journal of Operational Research 236 (3), 991–999.