Lower and Upper Bounds for the Two-Echelon Capacitated Location Routing Problem

Claudio Contardo
Vera Hemmelmayr
Teodor Gabriel Crainic

October 2011

CIRRELT-2011-63

Lower and Upper Bounds for the Two-Echelon Capacitated Location-Routing Problem

Claudio Contardo[1,2,*], Vera Hemmelmayr[1,3], Teodor Gabriel Crainic[1,4]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[2] Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

[3] Institute for Transport and Logistics Management, WU, Vienna University of Economics and Business, Nordbergstraße 15/D/6/621 C, 1090 Vienna, Austria

[4] Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

Abstract. In this paper we introduce two algorithms to solve the two-echelon capacitated location-routing problem (2E-CLRP). We introduce a branch-and-cut algorithm based on the solution of a new two-index vehicle-flow formulation, which is strengthened with the use of several families of valid inequalities. We also propose an adaptive large-neighborhood search (ALNS) method with the objective of finding good quality solutions quickly. The computational results on a large set of instances from the literature show that the ALNS outperforms existing heuristics. Furthermore, the branch-and-cut method provides tight lower bounds and is able to solve small and medium-size instances to optimality within reasonable computing times.

Keywords. Two-echelon capacitated location-routing problem, adaptive large neighborhood search, branch-and-cut.

Acknowledgements. Financial support from the Austrian Science Fund (FWF-project no.J3047) and the Natural Sciences and Engineering Research Council of Canada (NSERC) are gratefully acknowledged. The authors would like to thank Jean-François Cordeau for his valuable comments.

* Corresponding author: Claudio.Contardo@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2011

1 Introduction

The two-echelon capacitated location-routing problem (2E-CLRP) is an important combinatorial optimization problem arising in freight distribution. Formally, the problem can be stated as follows. Given three disjoint sets of nodes, $\mathcal{P}$ (the platforms), $\mathcal{S}$ (the satellites) and $\mathcal{C}$ (the customers), a planner must decide the location of a subset of platforms $\mathcal{P}' \subseteq \mathcal{P}$, the location of a subset of satellites $\mathcal{S}' \subseteq \mathcal{S}$ and to construct vehicle routes to visit each customer exactly once using a vehicle routed from an open satellite, which is also visited exactly once using a vehicle route from an open platform, at minimum total cost. To each platform $p \in \mathcal{P}$ we associate a fixed cost H^1_p and a capacity K^1_p. Similarly, to each satellite $s \in \mathcal{S}$ we associate a fixed cost H^2_s and a capacity K^2_s. Finally, for each customer $c \in \mathcal{C}$ we associate a demand d_c. We distinguish two echelons, one containing nodes in $\mathcal{P} \cup \mathcal{S}$ and the other containing nodes in $\mathcal{S} \cup \mathcal{C}$. At the first echelon we consider a graph $G^1 = (V^1, E^1)$, with $V^1 = \mathcal{P} \cup \mathcal{S}$ and $E^1 = \{\{u, v\} : u, v \in V^1,\ u \text{ and } v \text{ not both in } \mathcal{P}\}$. Similarly, at the second echelon we consider a graph $G^2 = (V^2, E^2)$ with $V^2 = \mathcal{S} \cup \mathcal{C}$ and $E^2 = \{\{u, v\} : u, v \in V^2,\ u \text{ and } v \text{ not both in } \mathcal{S}\}$. Two fleets of vehicles are used, one at each echelon. At each echelon the fleet is supposed to be homogeneous with vehicle capacities Q^1 and Q^2, respectively. To each edge $e \in E^1 \cup E^2$ we associate a routing cost equal to γ_e. The problem studied in this paper can also be extended to the more general problem in which customers and/or satellites can be served by several vehicles, which is known as multiple-sourcing. To distinguish from the multiple-sourcing case, we refer to the problem addressed in this paper as the single-sourcing variant. Other generalizations of the problem studied here include multi-commodity problems in which several different classes of products must be transported to customers, stochastic problems in which certain parameters are known only in distribution and dynamic problems in which customers demand is not known in advance but revealed in real time. A classification of the different classes of 2E-CLRP has been addressed in Boccia et al. [4].

The 2E-CLRP is a generalization of several other logistics problems. The capacitated location-routing problem (CLRP) is a particular case of the 2E-CLRP in which the location of a single platform of infinite capacity is known in advance and the costs on the first echelon can be neglected. The problem is to find the optimal satellite locations and to build vehicle routes around those satellites to satisfy the demand of the customer set. Recent algorithms for solving the CLRP include some exact methods [2, 1, 7, 6] and heuristics [18, 17, 19, 15, 10, 8].

The two-echelon capacitated vehicle routing problem (2E-CVRP) is also a particular case of the 2E-CLRP in which the location of a single platform is known in advance and the satellites are uncapacitated with no setup costs. A fleet of trucks is routed from the depot to the satellites, and then from these satellites smaller trucks are used to deliver the commodities to the final customers. Several models and algorithms have been designed for the 2E-CVRP with multiple-sourcing at the first echelon, both exact [14, 13] and heuristics [9, 10]. Perboli et al. [14] designed a branch-and-cut algorithm based on a three-index formulation of the problem. A third index is added to the vehicle-flow variables at the second echelon to specify the satellite serving a node. The formulation is strengthened with the use of subtour-elimination constraints and some flow-conservation constraints. Their algorithm is able to solve to optimality instances containing up to 21 customers. Perboli and Tadei

[13] strengthen the previous formulation with the use of new cuts, including capacity cuts, which allow their algorithm to scale and solve instances with up to 50 customers. Crainic et al. [9] developed multi-start heuristics for the solution of the 2E-CVRP. In the heuristics, an intensification phase in which feasible solutions are improved by local search is followed by a diversification phase to avoid local optima. Hemmelmayr et al. [10] developed an ALNS heuristic for the 2E-CVRP and the CLRP which is shown to provide better solutions than previous approaches. The algorithm is based on the destroy-and-repair principle in which two sets of operators (destroy operators and repair operators) are alternated. Nguyen et al. [12] introduced a GRASP complemented with path relinking method to solve the 2E-CVRP with single-sourcing in both echelons. In their method, a GRASP is used to build a set of solutions $\mathcal{P}$. Then, for each pair of solutions $S, T \in \mathcal{P}$, a path-relinking method tries to build an alternative better solution by applying different operators on S when trying to obtain T as a result of these operators. A learning process is used to guide the GRASP by restricting the opening of satellites to those which seem more promising during the first iterations. In Nguyen et al. [11] the same authors provide a multi-start iterated local search with tabu list and path relinking. The new algorithm outperforms their previous approach and they also report results on the instances by Boccia et al. [3]. In the ILS two search space are considered: 2E-CVRP solutions and a giant tour covering the main depot and the customers. A path-relinking procedure is presented that can used for intensification and/or post-optimization.

The 2E-CLRP has been formally introduced by Boccia et al. [4]. The authors proposed a two-index and a three-index vehicle-flow formulation as well as a set-partitioning formulation. While the third model is not included in their experimental experience, the first two models contain a polynomial number of variables and constraints and are solved using a general-purpose optimization solver. These models are shown to be effective for dealing with very small instances with up to 10 customers, 5 satellites and 3 platforms within reasonable computing times. For larger instances, they report average gaps of above 25%. Due to the highly combinatorial structure of the problem, a tabu search procedure has been developed by Boccia et al. [3]. In their algorithm, the 2E-CLRP is decomposed in two CLRP's. Each CLRP is then also decomposed into a capacitated facility location-problem (CFLP) and a multiple depot vehicle routing problem (MDVRP).

To our knowledge, only two articles address the 2E-CLRP with location decisions at both levels [3, 4]. Moreover, no exact procedure has been previously introduced to effectively deal with medium and large-size instances. In addition, only a few heuristics have been introduced in the literature [3, 11]. In this paper, we aim to fill these gaps by introducing an exact procedure to deal with medium and large-size instances and a new heuristic procedure to find good solutions of the 2E-CLRP quickly.

The main contributions of this paper can be summarized as follows.

i. To introduce a new two-index vehicle-flow formulation of the 2E-CLRP, which is shown to provide tight lower bounds, and to develop an efficient branch-and-cut algorithm based on the formulation which allows the solution of small and medium size instances containing up to 50 customers and 10 satellites and provides tight gaps for larger instances.

ii. To introduce a new heuristic solution method which outperforms previously proposed heuristic methods from the literature.

The scopes of both approaches introduced in this paper are complementary. While the exact algorithm can provide insightful information about the structure of the 2E-CLRP and provide optimal solutions for small to medium size instances, it does not scale for large instances in which even finding a feasible solution of reasonable quality can be prohibitive. Alternatively, the ALNS proposed in this paper provides good quality solutions quickly. Moreover, both methods validate each other using the lower bounds obtained with the branch-and-cut method and the upper bounds obtained with the metaheuristic. The remainder of the paper is organized as follows. Section 2 provides a mathematical formulation of the 2E-CLRP. The branch-and-cut is described in section 3 and the ALNS in section 4. Computational results are presented in section 5. Finally, section 6 concludes the paper.

2 Mathematical formulation

We now introduce a new two-index vehicle-flow formulation for the 2E-CLRP which is shown to provide tighter gaps in much shorter computational times than existing formulations of the 2E-CLRP [4]. In Boccia et al. [4] the authors introduced a three-index vehicle-flow formulation with a cubic number of variables. However, that formulation presents lots of symmetries and provides weak bounds which make it of little use within a branch-and-bound framework. Also, a two-index formulation is introduced, which produces even weaker bounds than the three-index one. The goal of introducing a new formulation is to be able to deal with medium to large-size instances and to obtain tighter bounds within reasonable computing times.

Let G^1, G^2 be the graphs at both echelons of the problem. For each platform $p \in \mathcal{P}$, we let w_p be a binary variable equal to 1 iff platform p is selected for opening. For each satellite $s \in \mathcal{S}$ we let z_s be a binary variable equal to 1 iff satellite s is chosen for opening. For each edge $e \in E^1$ we let u_e be a binary variable equal to 1 iff edge e is used exactly once, and v_e be a binary variable equal to 1 iff edge e is used twice (for single-customer routes). Analogously, for each $e \in E^2$ we let x_e be a binary variable equal to 1 iff edge e is used once, and y_e be a binary variable equal to 1 iff edge e is used twice. Finally, for each satellite $s \in \mathcal{S}$ we let $g_s \geq 0$ be a continuous variable equal to the volume of commodity shipped to/from satellite s.

For every vertex set U and for each echelon $k \in \{1, 2\}$, $\delta^k(U) \subseteq E^k$ denotes the edge subset at echelon k that contains one endpoint in U, $(T :^k U)$ denotes, for two disjoint subsets T and U, the edge subset that contains edges in E^k with one endpoint in T and the other in U. $E^k(U)$ denotes the subset of edges in E^k with both endpoints in U. For a given edge set $F \subseteq E^1$ let $x(F) = \sum_{e \in F} x_e$, $y(F) = \sum_{e \in F} y_e$ and similarly for $F \subseteq E^2$ and $u(F), v(F)$. Also, for a given satellite subset $\mathcal{S}' \subseteq \mathcal{S}$ we let $g(\mathcal{S}') = \sum_{s \in \mathcal{S}'} g_s$ and $z(\mathcal{S}') = \sum_{s \in \mathcal{S}'} z_s$. For a given customer set $U \subseteq \mathcal{C}$, let $d(U) = \sum_{c \in U} d_c$, $r(U) = \lceil d(U)/Q^2 \rceil$, $\overline{U} = \mathcal{C} \setminus U$, and also for a given satellite subset $\mathcal{S}'$ we let $\overline{\mathcal{S}'} = \mathcal{S} \setminus \mathcal{S}'$, $\rho(U, \mathcal{S}') = \lceil (d(U) - b(\mathcal{S}'))/Q^2 \rceil$. The quantities $r(U)$ and $\rho(U, \mathcal{S}')$ represent lower bounds on the number of second echelon vehicles needed to serve the customer subset U, and on the number of second echelon vehicles needed to serve U from facilities other than those in $\mathcal{S}'$. A valid formulation of the 2E-CLRP is the following:

$$\min \quad \sum_{p\in\mathcal{P}} H^1_p w_p + \sum_{s\in\mathcal{S}} H^2_s z_s + \sum_{e\in E^1} \gamma_e u_e + 2\sum_{e\in\delta^1(\mathcal{P})} \gamma_e v_e + \sum_{e\in E^2} \gamma_e x_e + 2\sum_{e\in\delta^2(\mathcal{S})} \gamma_e y_e \tag{1}$$

$$u(\delta^1(s)) + 2v(\mathcal{P} :^1 \{s\}) = 2z_s \qquad s\in\mathcal{S} \tag{2}$$
$$x(\delta^2(c)) + 2y(\mathcal{S} :^2 \{c\}) = 2 \qquad c\in\mathcal{C} \tag{3}$$
$$u(\delta^1(T)) + 2v(\mathcal{P} :^1 T) \geq 2\left(\frac{g(T)}{Q^1}\right) \qquad T\subseteq\mathcal{S}, |T|\geq 2 \tag{4}$$
$$x(\delta^2(U)) + 2y(\mathcal{S} :^2 U) \geq 2r(U) \qquad U\subseteq\mathcal{C}, |U|\geq 2 \tag{5}$$
$$u_{ps} + v_{ps} \leq w_p \qquad s\in\mathcal{S}, p\in\mathcal{P} \tag{6}$$
$$x_{sc} + y_{sc} \leq z_s \qquad s\in\mathcal{S}, c\in\mathcal{C} \tag{7}$$
$$u(\mathcal{P} :^1 \{s\}) + v(\mathcal{P} :^1 \{s\}) \leq z_s \qquad s\in\mathcal{S} \tag{8}$$
$$x(\mathcal{S} :^2 \{c\}) + y(\mathcal{S} :^2 \{c\}) \leq 1 \qquad c\in\mathcal{C} \tag{9}$$
$$u((\mathcal{P}\setminus\{p\})\cup\overline{T} :^1 T) + 2v(\mathcal{P}\setminus\{p\} :^1 T) \geq 2\left(\frac{g(T)-K^1_p}{Q^1}\right) \qquad p\in\mathcal{P}, T\subseteq\mathcal{S} \tag{10}$$
$$x((\mathcal{S}\setminus\{s\})\cup\overline{U} :^2 U) + 2y(\mathcal{S}\setminus\{s\} :^2 U) \geq 2\left(\frac{d(U)-g_s}{Q^2}\right) \qquad s\in\mathcal{S}, U\subseteq\mathcal{C} \tag{11}$$
$$u(\delta^1(T)) \geq 2(u(\{p\} :^1 \mathcal{P}') + u(\{p'\} :^1 \mathcal{P}\setminus\mathcal{P}')) \qquad T\subseteq\mathcal{P}, |T|\geq 2,\; p,p'\in T, \mathcal{P}'\subset\mathcal{P} \tag{12}$$
$$x(\delta^2(U)) \geq 2(x(\{c\} :^2 \mathcal{S}') + x(\{c'\} :^2 \mathcal{S}\setminus\mathcal{S}')) \qquad U\subseteq\mathcal{C}, |U|\geq 2,\; c,c'\in U, \mathcal{S}'\subset\mathcal{S} \tag{13}$$
$$0 \leq g_s \leq K^2_s z_s \qquad s\in\mathcal{S} \tag{14}$$
$$g(\mathcal{S}) = d(\mathcal{C}) \tag{15}$$
$$w_p \in \{0,1\} \qquad p\in\mathcal{P} \tag{16}$$
$$z_s \in \{0,1\} \qquad s\in\mathcal{S} \tag{17}$$
$$u_e \in \{0,1\} \qquad e\in E^1 \tag{18}$$
$$v_{ps} \in \{0,1\} \qquad p\in\mathcal{P}, s\in\mathcal{S} \tag{19}$$
$$x_e \in \{0,1\} \qquad e\in E^2 \tag{20}$$
$$y_{sc} \in \{0,1\} \qquad s\in\mathcal{S}, c\in\mathcal{C} \tag{21}$$

Constraints (2)-(3) are the degree constraints for the satellite nodes and the client nodes at the first and second echelons, respectively. Constraints (4)-(5) are the capacity constraints at both echelons that ensure the connectivity for the tours and make sure that the vehicle capacities are respected. Constraints (6)-(7) ensure that edges incident to a platform (for the first level) or to a satellite (for the second level) may only be used when the corresponding facility is opened. Constraints (8)-(9) are the so-called path elimination constraints for single satellite or single customer routes, respectively. They forbid routes that start at one facility, visit one customer or satellite and go back to a different facility. Constraints (10)-(11) are the facility capacity inequalities for the first and second echelon, respectively. Constraints (12)-(13) are the path elimination constraints that forbid routes that start at one facility and end at another facility. These constraints are complementary to constraints (8)-(9). Constraints (14) ensure that the flow going through satellites does not exceed their capacities. Finally, constraints (15) ensure that the total volume of commodity going through satellites coincide with the total demand.

This formulation includes a quadratic number of variables ($O(|E^2|)$) and an exponential number of constraints. Therefore, a method using it must rely on branch-and-cut techniques. Note also that the use of flow variables g at constraints (4), (10) and (11) link the use of the satellites at the two echelons. In the following section we introduce a branch-and-cut method using formulation (1)-(21). We introduce separation algorithms to dynamically add cuts as well as new valid inequalities to strengthen the formulation, thus providing tighter lower bounds. To better evaluate the strength of this formulation, we have also implemented the three-index formulation introduced by Boccia et al. [4]. In such a formulation, a third index is added to the vehicle-flow variables to take into account the vehicle performing a trip. This formulation contains a cubic number of variables and constraints, however it also presents many symmetries and provides weak bounds, as shown by our computational results.

3 Branch-and-cut method

We have developed a branch-and-cut algorithm based on the previously introduced formulation, which we strengthen with the use of several families of valid inequalities. The branch-and-cut algorithm is based on a relaxation of the original model in which integrality is dropped as well as some constraints. The resulting linear problem provides a lower bound on the optimal solution of the 2E-CLRP, however in rare cases this solution coincides with a feasible solution of the original problem. Indeed, this case divides in two possible subcases. Either some constraints of the problem (among the ones that were initially dropped) are violated, and are dynamically added to it, or the solution is not integer, in which case a branching method divides the problem in two complementary subproblems and the same method is applied recursively to both subproblems until one of the following three occurs: integrality is reached, so a new feasible solution is found; the subproblem is infeasible, and it can be discarded; or the subproblem provides a lower bound greater than the incumbent solution, so it can also be discarded for further exploration. Additionally, at each iteration the constraints which were dropped at the beginning are dynamically added to the problem, thus improving the resulting lower bound.

The algorithm introduced in this paper is inspired from the branch-and-cut methods introduced by Belenguer et al. [2] and Contardo et al. [7] for the CLRP. Indeed, the key observation is that, flow variables $(g_s)_{s\in\mathcal{S}}$ at the first echelon correspond to satellite demands, while at the second echelon they correspond to the satellite capacities. Hence, each echelon can be seen as a CLRP by giving variables $(g_s)_{s\in\mathcal{S}}$ the proper role. Therefore, we derive valid inequalities from the CLRP and make use of the separation algorithms described in the previously mentioned papers. In what follows, we describe the valid inequalities used in this article, including some derived from the CLRP plus some newly introduced. We also describe the separation algorithms, the node selection strategy, the branching strategy and the separation strategy used through the search tree.

3.1 Valid inequalities

In this section we introduce valid inequalities for the 2E-CLRP. They are subdivided into those which are specific for the first echelon and those which are specific for the second echelon. We introduce some new families of valid inequalities, and we also derive valid inequalities from the two-index vehicle-flow formulation of the CLRP, which have been introduced by Belenguer et al. [2] and Contardo et al. [7]. They include lifted cover inequalities (LCI), co-circuit constraints (CoCC), flow-assignment inequalities (FAI), y-capacity cuts (y-CC), strengthened facility capacity inequalities (SFCI), strengthened effective facility capacity inequalities (SEFCI), location-routing comb inequalities (LRCI), y-generalized large multistar inequalities (y-GLM), strengthened comb inequalities (SCI) and framed capacity inequalities (FrCI). For details on the inequalities as well as the separation algorithms used to identify violated inequalities we refer to [2, 7].

3.1.1 Inequalities on the first echelon

Note that variables g on this echelon represent the actual satellite demands. Therefore, all the inequalities valid for the CLRP are also valid for the first echelon of the 2E-CLRP when taking these quantities as demands. However, many of them become non-linear, such as those containing expressions combining the rounding operator $\lceil \cdot \rceil$ with the flow variables g, or those containing products of the demands g with vehicle-flow variables u, v (like several types of multistar inequalities) and thus cannot be introduced to the problem without losing linearity. We have then restricted the inclusion of valid inequalities at the first echelon to LCI, CoCC and FAI, plus the two following families of valid inequalities

$$u((\mathcal{P} \setminus \mathcal{P}') \cup \overline{\mathcal{S}'} :^1 \mathcal{S}') + 2v(\mathcal{P} \setminus \mathcal{P}' :^1 \mathcal{S}') \geq 2 \left(\frac{g(\mathcal{S}') - \sum_{p \in \mathcal{P}'} K^1_p}{Q^1} \right) \qquad \mathcal{S}' \subseteq \mathcal{S}, \mathcal{P}' \subset \mathcal{P} \quad (22)$$

$$u_{st} \leq \min\{z_s, z_t\} \qquad s, t \in \mathcal{S} \quad (23)$$

We call the first lifted facility capacity inequalities (LFCI) and the second flow-location inequalities (FLI).

3.1.2 Inequalities on the second echelon

On the second echelon, satellite capacities are given by the flow variables g. However, for a given satellite $s \in \mathcal{S}$ it still holds that no more than K^2_s units of flow can be delivered from it. Therefore, all the inequalities valid for the two-index vehicle-flow formulation of the CLRP are valid on the second echelon (the one restricted to the variables z, x and y). Note that when replacing K^2_s by g_s in those inequalities, many of them become non-linear and thus can not be added to the problem without losing the linearity of the problem. We then make use of the following inequalities: y-CC, SFCI, SEFCI, LRCI, y-GLM, CoCC, LCI, SCI, FAI and FrCI. Additionally, we also add the following family of location-routing generalized large multistar inequalities (LRGLM). Let $U \subseteq \mathcal{C}$ be a customer subset, $\mathcal{S}' \subseteq \mathcal{S}$ be a facility subset and $c \in \mathcal{C} \setminus U$ be a customer not in U. Let us define $\eta(\mathcal{S}', U, c) = \frac{1}{2}x(\mathcal{S}' : \{c\}) + y(\mathcal{S}' :$

$\{c\}) + x(U : \{c\})$. The following inequality is valid for the 2E-CLRP:

$$x((\mathcal{S} \setminus \mathcal{S}') \cup \overline{U} :^2 U) + 2y(\mathcal{S} \setminus \mathcal{S}' :^2 U) \geq \frac{2}{Q^2} \left(d(U) - g(\mathcal{S}') + \sum_{c \notin U} d_c \eta(\mathcal{S}', U, c) \right). \qquad (24)$$

The validity of these inequalities can be derived from the validity proof of the LRGLM introduced in Contardo et al. [7], by replacing the satellite capacities $(K_s^2)_{s \in \mathcal{S}}$ by the tighter capacities $(g_s)_{s \in \mathcal{S}}$. The dominance with respect to the original LRGLM comes from the inclusion of constraints (14).

3.2 Separation algorithms

For the separation of the inequalities directly translated from the CLRP, we make use of the separation algorithms introduced in Belenguer et al. [2] and Contardo et al. [7]. As a remark, note that for a given satellite subset $\mathcal{S}' \subseteq \mathcal{S}$ the degree constraints (2) in the first echelon imply the following identities

$$u(\delta^1(\mathcal{S}')) + 2v(\mathcal{P} :^1 \mathcal{S}') + 2u(E^1(\mathcal{S}')) = 2z(\mathcal{S}') \qquad (25)$$

which differ from the classical identity of other vehicle routing problems in that the right-hand side of the expression above is replaced by $2|\mathcal{S}'|$. As a consequence, separation algorithms must be adapted to make use of the right expression when necessary.

For the separation of the LFCI (22) we also make use of the separations algorithms for the SFCI, by setting the satellite demands to $(g_s)_{s \in \mathcal{S}}$.

The FLI (23) are a polynomial number so they are just inspected for violation one by one at each iteration.

Finally, for the LRGLM (24) we make use of the separation algorithms for the LRGLM for the CLRP as described in Contardo et al. [7], by replacing the effective satellite capacities $(z_s K_s^2)_{s \in \mathcal{S}}$ used in the original inequalities by the stronger ones $(g_s)_{s \in \mathcal{S}}$.

3.3 Node selection strategy

As our objective is to obtain the tightest possible lower bound for the problem, we use a best-bound strategy. After the exploration of the current node and the creation of the two children subproblems, the next node to explore is the uninspected node with the smallest lower bound (also known as best bound).

3.4 Branching strategy

The branching strategy we use is a hybrid mixing branching on variables and on cutsets. For the branching on cutsets, we add additional slack variables to the problem and the y-capacity cuts on the second echelon are added as identities. We then branch on these slack variables, similarly as in Belenguer et al. [2], Contardo et al. [7]. The branching is performed in the following order:

i. location variables w.

ii. location variables z.

iii. cutsets on the second echelon.

iv. vehicle-flow variables u, v.

v. vehicle-flow variables x, y.

3.5 Separation strategy

In our algorithm, we distinguish between first and second echelon inequalities. Indeed, after some preliminary computational experiments we decided to separate inequalities on the second echelon in the first place. If we are unable to identify any violated inequality, then we run the separation algorithms on the first echelon.

For the separation, we have implemented a dynamic separation strategy as explained in Contardo et al. [7]. We differentiate between the cuts that are needed to impose feasibility of integer solutions and those which can be seen as cuts to strengthen the problem. For each of the latter, we consider a counter representing the number of times that the corresponding family of cuts has been successfully separated and added to the problem. We keep track of this counter during the branching tree. At certain depths (as suggested by Contardo et al. [7] for the CLRP, we first check at depth 10, and then at multiples of 5), we deactivate from a branch (and thus from all of its children) the cuts for which the counter is zero, and for the remaining we reset the counter to zero. Using this strategy, we rapidly deactivate cuts that do not seem promising during certain branches of the tree, but we keep them where they seem to be useful. As a matter of fact, we have realized that cuts are rarely added after depth 25 in the branching tree.

4 Adaptive large neighborhood search

ALNS was proposed by Ropke and Pisinger [20] for the pickup and delivery problem. It extends the large neighborhood search algorithm of Shaw [22] and is also inspired by the ruin-and-recreate principle [21]. A general ALNS for several classes of routing problems was developed in Pisinger and Ropke [16], where the authors show that ALNS can outperform existing solution methods.

In ALNS there are two types of neighborhoods. Destroy neighborhoods that remove a certain number of customers from a solution and repair neighborhoods that reinsert these customers in the partial solution. ALNS works with an adaptive weight adjustment, where each operator is chosen based on its past success.

Our method is based on a previously proposed solution method for the 2E-CVRP and the CLRP [10]. In our method, destroy and repair operators are applied to customer and satellite nodes only, in the second echelon. The evaluation of such operators is subject to the partial re-optimization on the first echelon, which includes opening or closing platforms and re-assigning vehicle routes.

The main differences to the previously proposed ALNS [10] are the new acceptance strategy after the satellite configuration is changed, and the re-optimization on the first level, both of which are explained below in more detail. Finally, minor modifications were undertaken in the parameters and operators. These modifications were necessary to better exploit the structure of the problem, considering the interaction between the two levels.

We introduce a two-stage algorithm that deals with two types of destroy operators. There are larger operators, D_L, that change the current configuration of opened satellites and smaller operators, D_S, that remove only a certain number of customers, but do not explicitly open or close satellites. The operators from set D_L are called every ω iterations without improvements. For these operators, the algorithm starts from the best found solution. Only if the incumbent solution is within η % of the best found solution, it can serve as the new incumbent with a probability of Θ. Moreover, a solution yielded by these operators is accepted, even if they are not improving. The goal of this procedure is that the algorithm can explore different configurations of satellites with the smaller operators of set D_S. After a certain number of non-improving iterations, the configuration is changed and the algorithm starts from either the incumbent or the best found solution.

The customers that were removed by the destroy operators are then inserted by means of an insertion operator to minimize the objective value. After the destroy and repair operators have been executed, the first level is improved by recursively calling again the ALNS to solve the first-level CLRP. For the instances containing only one platform (like those used in Nguyen et al. [12]), a simplified procedure based on single customer moves and swaps is executed. In these instances, the first level is not a CLRP and hence it is not necessary to call the ALNS to solve a CLRP at every iteration.

The destroy and repair operators are selected by a roulette wheel selection mechanism. Every operator j has a score π_j. This score is updated by adding σ, every time that operator j finds a new global best solution. The probability of being selected in the roulette wheel is based on $\pi_j / \sum_{k=1}^{p} \pi_k$, where p is the number of operators considered. Algorithm 1 shows a pseudocode for our ALNS procedure.

4.1 Search space

During the search, we allow the exploration of infeasible solutions. More precisely, violations of the constraints on vehicle capacity, satellite and platform capacity are penalized by a weighted penalty function. The objective function is $f(s) = c(s) + \alpha d(s) + \beta e(s)$, where $c(s)$ corresponds to the routing cost and the opening cost of satellites and platforms, $d(s)$ represents violations of the vehicle capacity, and $e(s)$ represents violations of the satellite and platform capacity. The parameters α and β are the corresponding weights. The weights are adjusted dynamically during the search. If a violation occurs, the corresponding weight is multiplied by a factor $\delta > 1$, if the solution is feasible, it is divided by δ. The weights are restricted to an interval $[\iota; \kappa]$ that guarantees that the search starts with a reasonable high weight and also prevents the weight from going to infinity.

4.2 Initial solution

For the initial solution, we open the configuration of satellites that yields the lowest cost and can serve the total customer demand. Then, customers are randomly assigned to a satellite with a bias towards the shortest distance and vehicle routes are constructed by

Algorithm 1 Basic steps of the ALNS algorithm

```
procedure ALNS-2ECLRP
    $s, s', s^* \leftarrow InitialSolution, InitializeScores(\boldsymbol{\pi})$
    repeat
        if $s'$ was $\omega$ iterations without improvement then
            $N^- \leftarrow ChooseDestroyOperator(D_L, \boldsymbol{\pi})$
            if $f(s') < (1+\eta)f(s^*)$ then
                only update $s' \leftarrow s^*$ with probability $\Theta$
            else
                $s' \leftarrow s^*$
            end if
        else
            $N^- \leftarrow ChooseDestroyOperator(D_S, \boldsymbol{\pi})$
        end if
        $N^+ \leftarrow ChooseRepairOperator(R, \boldsymbol{\pi})$
        $s' \leftarrow DestroyAndRepair(s, N^-, N^+)$
        solve the first-level LRP by ALNS-2ECLRP
        if $s'$ was $\omega$ iterations without improvement then
            $s' \leftarrow LocalSearch(s')$
            $s \leftarrow s'$
        else if $f(s') < (1+\theta)f(s^*)$ then
            $s' \leftarrow LocalSearch(s')$
        end if
        if $f(s') < f(s)$ then
            $s \leftarrow s'$
        end if
        if $f(s) < f(s^*)$ then
            $s^* \leftarrow s$
        end if
        Update scores $(\boldsymbol{\pi})$
    until the stopping condition is met
    return $s^*$
end procedure
```

means of the Clarke and Wright [5] Savings Algorithm. To construct a first level solution, a random platform is opened and satellites are assigned to it, no matter if platform capacity is violated. The vehicle routes at that platform are build with the Clarke and Wright [5] Savings Algorithm.

4.3 Destroy and repair operators

We use eight destroy operators and four repair operators in our algorithm. Three of the destroy operators explicitly open or close a satellite. These are *Satellite Removal*, *Satellite Opening*, and *Satellite Swap*. They are the "larger" neighborhoods of set D_L. *Satellite Removal* chooses one random satellite. This satellite is closed by removing all the customer routes originating from it. Furthermore, the satellite is removed from the first level routes. In the *Satellite Opening* operator, a random satellite is opened. The q customers that have the minimum distance to this satellite are removed from their current routes. *Satellite Swap* closes one satellite and opens another one. The satellite that will be closed is chosen randomly and the satellite that will be opened is chosen randomly with a probability that is inversely proportional to the distance to the closed satellite.

The following destroy operators from the set D_S only remove a limited number of customers, but do not explicitly change the satellite configuration. However, they can close satellites if all the customers of a satellite are removed and no customer is inserted at this satellite anymore. It can also happen that a satellite is opened by the diversification mechanism in *Route Removal*. The operator *Random Removal* is a very simple operator that removes q random customers. *Worst Removal* selects the q "worst" customers. These are the customers that are in the most expensive insertion positions, i.e., the positions where the difference between the cost with the customer in the solution compared to the cost without the customer in the solutions is large. We normalize this gain by dividing it by the average cost of the ingoing arcs. Moreover, a perturbation factor d is added, $d \in [0.8, 1.2]$. In the *Related Removal* operator, a random seed customer and the $q - 1$ related customers are removed. We define "related" by the distance to the seed customer. In *Route Removal* one route is removed. All customers that were contained in the removed route are put in the customer pool. In order to avoid cycling, it is forbidden to open a new route at the corresponding satellite. Another route is opened at a random satellite. This a mechanism that prevents the rare cases where all satellites are closed, because all customers are served by a single route originating from the only open satellite. Moreover, it is important for diversification. *Route Redistribution* is the largest destroy operator that removes between 1 and 3 random routes. The routes are selected based on their distance between their current satellite and any other open satellite. This selection mechanism reflects the idea that customers close to several satellites may benefit more from a reassignment than those that are only close to one satellite. Moreover, a perturbation factor d is added, $d \in [0.8, 1.2]$.

Finally, we use four insertion operators. Their goal is to select a satellite among the set of open satellites, to select a route and to select an insertion position for every customer that has to be inserted. The opening of a new route is considered too, unless this is forbidden because the *Route Removal* operator was applied to the considered satellite in the same iteration. In *Greedy Insertion* the customers are inserted in a a random order into the position that minimizes the total insertion cost over all satellites and routes. There are two versions of *Greedy Insertion*: *Greedy Insertion Perturbation* uses an additional perturbation factor d, $d \in [0.8, 1.2]$, that provides diversification. In the *Greedy Insertion Forbidden* operator, the satellite from which the customer has just been removed, cannot be selected for insertion in the same iteration. *Greedy Insertion Perturbation* and *Greedy Insertion Forbidden* guarantee that the insertion position is not determined too greedily based on second-level insertion cost. *Regret Insertion* uses a more sophisticated insertion scheme. Customers for which the difference between the best and the k-th best insertion position is large, are favored for insertion. From the set of untreated customers U, a customer i is chosen for insertion according to $i := \arg\max_{i \in U}(\sum_{h=2}^{k} \Delta f_i^h - \Delta f_i^1)$. When a customer is inserted, the insertion positions for the remaining customers have to be recomputed.

4.4 Local search

The goal of local search is to improve the CLRP solution on the second level. It is performed after the *Satellite Removal*, *Satellite Swap* or *Satellite Opening* operators and for promising solutions. Promising solutions are solutions for which the objective value is within η % of the best found solution. The following operators are used within the local search framework: split, move, swap, 2-opt and 2-opt*. They are performed sequentially, in a first improvement manner. For more details on each of these operators, we refer to [10].

5 Computational results

Our methods were coded in C++, compiled with the Intel C++ compiler v11.0 and run on an Intel Xeon E5462, 3.0 Ghz processor with 16GB of memory. For the solution of linear and integer problems, we use CPLEX 12.2.

5.1 Test instances

We have tested our methods on several sets of standard instances from the literature. Nguyen et al. [12] introduced two sets of instances that contain only one platform at the first level, i.e.,$|\mathcal{P}| = 1$. The first set is an extension of the set "Prodhon" from the CLRP and contains 30 instances with 20 to 200 customers and 5 to 10 depots. The second set contains 24 newly generated instances and is referred to as set "Nguyen". The number of customers in these instances ranges from 25 to 200 and the number of satellites from 5 to 10. The last three sets of instances are instances used by Nguyen et al. [11], generated by Sterle [23] according to the specifications explained in the paper of Boccia et al. [3] and contain a total of 93 instances. Note that these sets do not correspond to the instances sets used in Boccia et al. [3] or Boccia et al. [4] which were not available from the authors but have been regenerated. The three sets of instances, I_1, I_2 and I_3, differ in the location of the satellites and the platforms. The number of customers in the instances ranges from 8 to 200, the number of satellites from 3 to 20 and the number of platforms from 2 to 5. In total, our experiments are run on 147 instances.

5.2 Parameter settings

For the branch-and-cut method, the parameters associated to the separation of each family of inequalities are set as in Contardo et al. [7] for both echelons.

For the ALNS, the parameters were set according to experimental tests. The parameters for solution acceptance are set to the following values. The threshold η, which defines that incumbent solutions that have an objective function value within η % of the best solution found, can be accepted, is set to 1%. The probability of acceptance, Θ, is set to 0.5. For the CLRP, values for ω in the range [100; 2000] yielded the best performance. To solve the 2E-CLRP, ω was set to 1000. The number of customers to remove is a random integer between ρ and τ. We set ρ to 1 and τ to $\lceil 0.6|J| \rceil$. For the weighted penalty function, δ was set to 1.1, ι to 5 and κ to 10,000. The parameter that is added to the score π_j every time a new best solution is found, σ, is set to 1. As a stopping condition, we choose the number of iterations. We decided that 500,000 iterations are a good trade-off between runtime and solution quality. A regret-3 heuristic is used in *Regret Insertion* and θ, which is the threshold that identifies promising solutions that are selected for local search, is set to 0.2.

5.3 Numerical Results

Due to the randomness incorporated in the ALNS, we have performed ten runs of our method on each instance. In Table 1 we compare the ALNS against the methods GRASP+PR [12] and MS-ILS+PR [11]. In these papers, the authors do not report the average solution quality of their algorithms, but only the best solutions found after five runs for each instance. To make a fair comparison to these methods we restrict to our best results. For the sets I_1, I_2 and I_3, Nguyen et al. [11] only report results for the instances with 50 and more customers, which we also include in our comparison. In this table, headers gap_{avg} and gap_{min} correspond to the average and the minimum gaps, respectively, for each algorithm, when available. The gap between a solution of value z and a best known solution of value z^* is computed as $(z - z^*)/z^* \times 100$. Header t_{avg} stands for the average CPU time spent, in seconds, by each method (times are not scaled and comparisons based on time should be done with care). As we can see in the results obtained, our method is robust and clearly outperforms the previous heuristics of Nguyen et al. [12] and Nguyen et al. [11]. Indeed, our ALNS provides solutions which are 0.56%, 0.67%, 2.85%, 2.07% and 1.64% better than the best upper bounds on sets Prodhon, Nguyen, I_1, I_2 and I_3, respectively (1.68% on average among all the instances included in the comparison). In terms of CPU time, our method is comparable to that of Nguyen et al. [11].

In Table 2 we report the aggregated results obtained by our branch-and-cut algorithm and compare against our implementation of the three-index vehicle-flow formulation of Boccia et al. [4] which has been run on the same machine as our methods. For the computations, we use as cutoff values the best upper bounds obtained by the ALNS (as reported by Tables 8-10 in the appendix). In this table, header *#opt* stands for the number of instances that were solved to optimality. Headers gap_{lr} and t_{lr} correspond to the gap (in %) and the CPU time (in seconds) spent at the solution of the linear relaxation of the problem. Analogously, columns gap and t stand for the gap and CPU time spent after a maximum of two hours of computation. Given a lower bound z_{lb} and an upper bound z_{ub} the gap is computed

Set	GRASP+PR		MS-ILS+PR		ALNS		
	gap_{min}	t_{avg}	gap_{min}	t_{avg}	gap_{avg}	gap_{min}	t_{avg}
Prodhon	0.36	14.2	0.10	178.3	-0.05	-0.56	465.82
Nguyen	0.80	13.1	0.00	112.20	-0.34	-0.67	191.97
$I_1^\dagger$	–	–	0.00	917.10	-2.33	-2.85	839.60
$I_2^\dagger$	–	–	0.00	928.00	-1.49	-2.07	913.70
$I_3^\dagger$	–	–	0.00	935.10	-1.17	-1.64	909.85

† Restricted to instances containing 50 or more customers.

Table 1: Aggregated results of the ALNS

Set	B&C					3-index model†		
	$\#opt$	gap_{lr}	t_{lr}	gap	t	$\#opt$	gap	t
Prodhon	8/30	7.26	981.01	3.58	5671.41	0/30	21.83	7200
Nguyen	11/24	8.94	516.68	3.05	4170.71	0/24	23.45	7200
I_1	17/31	11.33	584.37	2.96	3378.64	6/31	19.24	5905.62
I_2	19/31	8.41	177.17	2.48	3002.14	6/31	17.55	5629.40
I_3	20/31	9.29	84.06	1.85	2664.55	6/31	17.05	5639.27

† Instances with 150 or more customers could not be loaded into memory

Table 2: Aggregated results of the B&C algorithm

as $(z_{ub} - z_{lb})/z_{ub} \times 100$. As shown in Table 2 our algorithm can solve more and larger instances than the previous three-index formulation of Boccia et al. [4]. For the instances that remain unsolved, our branch-and-cut method provides much tighter gaps in lower computing times. Also, it is possible to establish a comparison between our ALNS and the branch-and-cut. Indeed, if we consider the upper bounds provided by the ALNS and the lower bounds obtained with the branch-and-cut method, they are at an average distance of 2.77%, which leaves little space for further improvements and also validates both algorithms introduced in this paper.

6 Conclusion

In this paper we have presented lower and upper bounds for the 2E-CLRP. We have introduced a compact two-index vehicle-flow formulation, proposed several families of valid inequalities and embedded it into a branch-and-cut solver. To the best of our knowledge, this is the first time that an exact method has been proposed for this problem class. The method is able to solve to optimality small and medium size instances containing up to 50 customers, and still provides tight lower bounds for the instances that cannot be solved. We have also introduced an adaptive large neighborhood search (ALNS) method capable of providing good lower bounds in short computing times. The ALNS outperforms previous heuristics for the 2E-CVRP with single-sourcing constraints in terms of upper bound quality and also provides good quality upper bounds for the instances of 2E-CLRP. Moreover, our ALNS was able to improve the best known solutions on 134 instances out of 147. When comparing our methods, we observe that the lower bounds obtained by the branch-and-cut method lie no further than 2.77% on average below the best solutions found by the ALNS, which validates the robustness of both approaches. As an avenue of future research, we believe that exploring other heuristic approaches like matheuristics combining integer-programming methods with pure metaheuristics could lead to better upper bounds. On the other hand, we believe that embedding the inequalities used in this paper into a branch-and-cut-and-price solver could result in a more robust exact method being able to scale better on large instances. Also, the methodologies used in this paper can still be used to solve some related problems combining location with routing decisions.

Acknowledgments

Financial support from the Austrian Science Fund (FWF-project no.J3047) and the Natural Sciences and Engineering Research Council of Canada (NSERC) are gratefully acknowledged. The authors would like to thank Jean-François Cordeau for his valuable comments.

References

[1] Baldacci, R., Mingozzi, A., and Wolfler-Calvo, R. (2011). An exact method for the capacitated location-routing problem. *Operations Research*. Forthcoming.

[2] Belenguer, J. M., Benavent, E., Prins, C., Prodhon, C., and Wolfler-Calvo, R. (2010). A branch-and-cut algorithm for the capacitated location routing problem. *Computers & Operations Research*, 38:931–941.

[3] Boccia, M., Crainic, T., Sforza, A., and Sterle, C. (2010). A metaheuristic for a two echelon location-routing problem. In Festa, P., editor, *Experimental Algorithms*, volume 6049 of *Lecture Notes in Computer Science*, pages 288–301. Springer.

[4] Boccia, M., Crainic, T., Sforza, A., and Sterle, C. (2011). Location-routing models for designing a two-echelon freight distribution system. Technical Report CIRRELT-2011-06, Université de Montréal.

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

[6] Contardo, C., Cordeau, J.-F., and Gendron, B. (2011a). A branch-and-cut-and-price algorithm for the capacitated location-routing problem. Technical Report CIRRELT-2011-44, Université de Montréal.

[7] Contardo, C., Cordeau, J.-F., and Gendron, B. (2011b). A computational comparison of flow formulations for the capacitated location-routing problem. Technical Report CIRRELT-2011-47, Université de Montréal.

[8] Contardo, C., Cordeau, J.-F., and Gendron, B. (2011c). A GRASP + ILP-based metaheuristic for the capacitated location-routing problem. Technical Report CIRRELT-2011-52, Université de Montréal.

[9] Crainic, T., Mancini, S., Perboli, G., and Tadei, R. (2011). Multi-start heuristics for the two-echelon vehicle routing problem. In Merz, P. and Hao, J.-K., editors, *Evolutionary Computation in Combinatorial Optimization: 11th European Conference, EvoCOP 2011, Torino, Italy, April 27-29, 2011, Proceedings*, volume 6622 of *Lecture Notes in Computer Science*, pages 179–190.

[10] Hemmelmayr, V. C., Cordeau, J.-F., and Crainic, T. G. (2011). An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Technical Report CIRRELT-2011-42, Université de Montréal.

[11] Nguyen, V.-P., Prins, C., and Prodhon, C. (2011a). A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem. *Engineering Applications of Artificial Intelligence*. In press.

[12] Nguyen, V.-P., Prins, C., and Prodhon, C. (2011b). Solving the two-echelon location routing problem by a grasp reinforced by a learning process and path relinking. *European Journal of Operational Research*, In Press, Accepted Manuscript.

[13] Perboli, G. and Tadei, R. (2010). New families of valid inequalities for the two-echelon vehicle routing problem. *Electronic Notes in Discrete Mathematics*, 36:639–646.

[14] Perboli, G., Tadei, R., and Vigo, D. (2011). The two-echelon capacitated vehicle routing problem: models and math-based heuristics. *Transportation Science*. doi 10.1287/trsc.1110.0368.

[15] Pirkwieser, S. and Raidl, G. R. (2010). Variable neighborhood search coupled with ILP-based very large-neighborhood searches for the (periodic) location-routing problem. In *Hybrid Metaheuristics - Seventh International Workshop, HM 2010*, volume 6373 of *Lecture Notes in Computer Science*, pages 174–189, Vienna.

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

[17] Prins, C., Prodhon, C., Ruiz, A., Soriano, P., and Wolfler-Calvo, R. (2007). Solving the capacitated location-routing problem by a cooperative Lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41:470–483.

[18] Prins, C., Prodhon, C., and Wolfler-Calvo, R. (2006). Solving the capacitated location-routing problem by a GRASP complemented by a learning process and path relinking. *4OR*, 4:221–238.

[19] Prodhon, C. and Prins, C. (2008). A memetic algorithm with population management (MA|PM) for the periodic location-routing problem. In Blesa, M., Blum, C., Cotta, C., Fernández, A., Gallardo, J., Roli, A., and Sampels, M., editors, *Hybrid Metaheuristics*, volume 5296 of *Lecture Notes in Computer Science*, pages 43–57. Springer Berlin / Heidelberg.

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

[21] Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., and Dueck, G. (2000). Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139–171.

[22] Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In Maher, M. and Puget, J.-F., editors, *Principles and Practice of Constraint Programming – CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin / Heidelberg.

[23] Sterle, C. (2011). Private communication.

Appendix

In this appendix we provide detailed results for both algorithms introduced in this paper and provide a brief discussion on our results.

In Tables 3-7, we report detailed results obtained by our ALNS. In these tables, header z^*_{BKS} stands for the best known solution found by the methods of Nguyen et al. [12] and Nguyen et al. [11]. Header z^*_{avg} stands for the average solution value found by our method. Header gap_{avg} stands for gap between our average values and the best known solution. It is computed as $(z^*_{avg} - z^*_{BKS})/z^*_{BKS} \times 100$. Header z^*_{min} stands for the best solution found in the 10 runs. Header gap_{min} stands for the gap between the best known solution and our best solution, which is computed as $(z^*_{min} - z^*_{BKS})/z^*_{BKS} \times 100$.

In Tables 8-10 we report the best solutions found by our ALNS, including the parameter calibration phase, and compare against the previous upper bounds when possible. As shown in these tables, our ALNS was able to improve the upper bound in a total of 41 instances out of 54 for sets Prodhon and Nguyen, and in all instances for sets I_1, I_2 and I_3. Solutions for which optimality was proven by the branch-and-cut method are marked with an asterisk. Moreover, according to our results, our ALNS was always able to improve the previous upper bounds or at least find the same value. In these tables, the legend z^*_{BKS} stands for the best known solutions, z^*_{ALNS} stands for the best solutions found by our ALNS and *gap* stands for the gap between both solutions, computed as $(z^*_{ALNS} - z^*_{NPP})/z^*_{NPP} \times 100$.

In Tables 11-15 we report detailed results of our branch-and-cut method. In these tables, columns labeled z_{UB} stands for the upper bound value. Columns labeled z_{lr} stand for the lower bound at the linear relaxation. Columns labeled gap_{lr} stand for the gap at the linear relaxation, computed as $(z_{UB} - z_{lr})/z_{UB} \times 100$. Columns labeled t_{lr} represent the CPU time (in seconds) spent for solving the linear relaxation. Analogously, columns labeled z, *gap* and t represent the final lower bound, the final gap and the total CPU time (in seconds) after a maximum time of two hours. Finally, columns labeled *#nodes* stand for the number of nodes inspected by the branch-and-cut algorithm. In bold characters we highlight instances on which we were able to prove optimality. Our branch-and-cut method is able to solve to optimality small and medium size instances with up to 50 customers and 10 satellites. In total, we proved optimality for 75 out of the 147 instances considered in our study.

Instance	z^*_{BKS}	z^*_{avg}	gap_{avg}	z^*_{min}	gap_{min}	t_{avg}	t^*_{min}
20-5-1	89075	89075.0	0.00	89075	0.00	26.3	2.8
20-5-1b	61863	61863.0	0.00	61863	0.00	26.1	0.2
20-5-2	85290	84478.0	-0.95	84478	-0.95	28.9	2.8
20-5-2b	60838	60838.0	0.00	60838	0.00	29.1	0
50-5-1	134855	131036.6	-2.83	130843	-2.98	96.9	8.2
50-5-1b	101530	101530.0	0.00	101530	0.00	94.2	8.7
50-5-2	132159	131878.3	-0.21	131825	-0.25	130.6	37.2
50-5-2b	110547	110332.0	-0.19	110332	-0.19	160.7	21.1
50-5-2BIS	122654	122626.5	-0.02	122599	-0.04	121.2	43.2
50-5-2bBIS	105776	105719.9	-0.05	105696	-0.08	131.6	57.4
50-5-3	128379	128379.0	0.00	128379	0.00	88.4	17.8
50-5-3b	104006	104006.0	0.00	104006	0.00	114.9	14.2
100-5-1	320130	320511.2	0.12	319137	-0.31	646.9	445
100-5-1b	258205	258540.8	0.13	257349	-0.33	1179.9	607.9
100-5-2	234179	231305.0	-1.23	231305	-1.23	316	43.7
100-5-2b	195426	194771.0	-0.34	194729	-0.36	1641	602.4
100-5-3	245944	244418.0	-0.62	244194	-0.71	375.5	167.9
100-5-3b	195254	194239.4	-0.52	194110	-0.59	377	210.9
100-10-1	358939	365036.1	1.70	358068	-0.24	158.9	65.6
100-10-1b	302584	303089.8	0.17	297167	-1.79	155.2	118.7
100-10-2	306303	307762.9	0.48	305402	-0.29	270	163
100-10-2b	264389	266642.1	0.85	265138	0.28	348.2	225.9
100-10-3	313249	318499.8	1.68	313517	0.09	215.7	106
100-10-3b	266383	270326.5	1.48	264096	-0.86	256.9	181.6
200-10-1	554598	559774.8	0.93	552816	-0.32	1039.2	648.7
200-10-1b	452286	459033.1	1.49	448236	-0.90	1811.9	927.6
200-10-2	502173	498659.4	-0.70	498199	-0.79	576.4	339.3
200-10-2b	425311	423517.6	-0.42	423048	-0.53	1723.8	1161.5
200-10-3	533732	535823.8	0.39	534569	0.16	741.3	285.2
200-10-3b	418800	407070.2	-2.80	404284	-3.47	1091.9	439.1
Average			-0.05		-0.56	465.82	231.79

Table 3: Results of the ALNS on the instances of set Prodhon

Instance	z^*_{BKS}	z^*_{avg}	gap_{avg}	z^*_{min}	gap_{min}	t_{avg}	t^*_{min}
25-5N	80370	80370	0.00	80370	0.00	38	0.1
25-5Nb	64562	64562	0.00	64562	0.00	36.4	0
25-5MN	78947	78947	0.00	78947	0.00	50.6	0.5
25-5MNb	64438	64438	0.00	64438	0.00	30.6	0
50-5N	138126	138093.2	-0.02	137815	-0.23	74.2	33.9
50-5Nb	111062	110094	-0.87	110094	-0.87	113.8	32.3
50-5MN	123484	123484	0.00	123484	0.00	105.2	13.1
50-5MNb	105401	105401	0.00	105401	0.00	77.5	23.1
50-10N	116132	115843	-0.25	115725	-0.35	90.3	21.3
50-10Nb	87315	87315	0.00	87315	0.00	102	20.7
50-10MN	135748	135556.6	-0.14	135519	-0.17	76.3	39
50-10MNb	110613	110636.1	0.02	110613	0.00	49.2	19.1
100-5N	196910	194012.3	-1.47	193228	-1.87	224.5	154.3
100-5Nb	159086	159029.9	-0.04	158927	-0.10	234.9	133
100-5MN	207119	204819.6	-1.11	204682	-1.18	271.2	135.7
100-5MNb	166115	165863.1	-0.15	165744	-0.22	220.9	112.9
100-10N	215792	216285.5	0.23	212847	-1.36	150.8	70.8
100-10Nb	156401	156261.6	-0.09	155489	-0.58	177.2	70.3
100-10MN	205964	202491.9	-1.69	201275	-2.28	155.4	104.7
100-10MNb	170706	170985.4	0.16	170625	-0.05	178.7	114.2
200-10N	353685	351770	-0.54	347395	-1.78	420.5	237.2
200-10Nb	262072	258397	-1.40	256171	-2.25	492.4	340.8
200-10MN	332345	329913.7	-0.73	326454	-1.77	547.3	354.7
200-10MNb	292523	292357.6	-0.06	289742	-0.95	689.5	481.7
Average			-0.34		-0.67	191.97	104.72

Table 4: Results of the ALNS on the instances of set Nguyen

Instance	z^*_{BKS}	z^*_{avg}	gap_{avg}	z^*_{min}	gap_{min}	t_{avg}	t^*_{min}
2-3-8		575.70		575.70		15.3	0.2
2-4-8		549.34		549.34		18.0	0.2
2-3-9		878.69		878.69		36.4	0.0
2-4-10		806.72		806.72		30.1	0.0
3-5-10		696.94		696.94		22.8	0.0
3-8-10		596.56		596.56		40.4	4.2
2-10-15		732.48		732.48		48.6	2.4
3-10-15		686.71		686.71		49.0	1.7
2-4-15		1064.52		1064.52		47.5	0.3
3-5-15		933.75		933.75		49.1	0.2
3-8-15		730.36		730.36		55.7	0.9
2-10-20		937.40		937.40		62.1	10.3
3-10-20		761.29		761.29		65.5	0.9
4-10-20		1149.72		1149.72		66.6	5.9
2-8-20		1029.59		1029.59		67.0	6.3
3-8-20		848.31		848.31		64.8	1.7
2-10-25		1030.40		1030.40		74.9	6.7
3-10-25		1062.30		1062.30		75.1	2.8
4-10-25		1607.94		1607.94		76.3	12.3
2-8-25		950.87		950.87		71.8	2.7
3-8-25		870.69		870.69		76.4	0.1
5-10-50	1207.31	1133.74	-6.09	1132.63	-6.19	341.8	178.8
5-8-50	1171.69	1163.45	-0.70	1162.44	-0.79	321.5	119.6
5-10-75	1561.5	1540.91	-1.32	1540.23	-1.36	507.2	219.9
5-15-75	1700.32	1700.38	0.00	1686.21	-0.83	527.3	247.6
5-10-100	2192.14	2136.82	-2.52	2124.09	-3.10	659.9	499.5
5-20-100	1989.48	1983.05	-0.32	1973.08	-0.82	705.2	605.2
5-10-150	1953.55	1893.92	-3.05	1883.44	-3.59	1187.8	718.9
5-20-150	1905.81	1889.07	-0.88	1869.53	-1.90	1203.1	901.5
5-10-200	2601.33	2461.77	-5.36	2443.80	-6.06	1431.0	1135.6
5-20-200	2307.53	2238.06	-3.01	2219.54	-3.81	1511.5	1073.8
Restricted average			-2.33		-2.85	839.6	570.0
Total average						306.76	185.81

Table 5: Results of the ALNS on the instances of set I_1

Instance	z^*_{BKS}	z^*_{avg}	gap_{avg}	z^*_{min}	gap_{min}	t_{avg}	t^*_{min}
2-3-8		575.7		575.7		15.9	0.0
2-4-8		604.13		604.13		24.5	0.0
2-3-9		386.15		386.15		17.5	0.0
2-4-10		629.38		629.38		18.4	0.0
3-5-10		551.45		551.45		27.3	0.0
3-8-10		504.2		504.2		39.6	0.0
2-10-15		709.1		709.1		53.7	1.0
3-10-15		777.49		777.49		56.5	0.5
2-4-15		827.81		827.81		45.3	0.4
3-5-15		1075.22		1075.22		56.1	0.0
3-8-15		652.58		652.58		49.5	1.9
2-10-20		766.24		766.24		62.3	2.3
3-10-20		744.27		744.27		62.0	3.0
4-10-20		793		793		68.5	0.6
2-8-20		772.29		772.29		65.5	14.2
3-8-20		758.06		758.06		65.6	1.4
2-10-25		961.59		961.59		81.5	3.7
3-10-25		987.57		987.57		79.6	0.8
4-10-25		1125.56		1125.56		79.0	4.0
2-8-25		912.02		912.02		77.9	9.3
3-8-25		979.62		979.62		77.9	0.0
5-10-50	1265.73	1256.44	-0.73	1256.44	-0.73	410.8	96.5
5-8-50	1123.42	1121.13	-0.20	1121.13	-0.20	386.6	51.5
5-10-75	1718.25	1691.15	-1.58	1691.15	-1.58	638.7	208.6
5-15-75	1751.14	1748.62	-1.58	1742.25	-1.58	670.1	446.6
5-10-100	2290.64	2242.28	-2.11	2231.21	-2.59	870.9	535.5
5-20-100	2039.25	2018.74	-1.01	1996.34	-2.10	872.0	592.8
5-10-150	1768.79	1734.64	-1.93	1728.05	-2.30	1004.6	683.8
5-20-150	1664.2	1654.83	-0.56	1630.29	-2.04	1070.3	832.3
5-10-200	2292.47	2158.24	-5.86	2147.51	-6.32	1477.2	1002.8
5-20-200	2097.74	2081.58	-0.77	2049.01	-2.32	1736.0	1369.6
Restricted average			-1.49		-2.07	913.7	582.0
Total average						331.01	189.13

Table 6: Results of the ALNS on the instances of set I_2

Instance	z^*_{BKS}	z^*_{avg}	gap_{avg}	z^*_{min}	gap_{min}	t_{avg}	t^*_{min}
2-3-8		578.33		578.33		15.3	0.1
2-4-8		450.71		450.71		17.2	0.0
2-3-9		454.63		454.63		15.6	0.0
2-4-10		540.61		540.61		20.3	0.0
3-5-10		745.48		745.48		30.5	0.1
3-8-10		412.91		412.91		20.4	0.0
2-10-15		605.11		605.11		48.6	0.0
3-10-15		546.61		546.61		53.0	2.8
2-4-15		688.87		688.87		45.1	0.3
3-5-15		1005.02		1001.28		50.5	2.7
3-8-15		578.23		578.23		50.8	2.3
2-10-20		744.82		744.82		62.7	2.8
3-10-20		728.17		728.17		65.0	8.2
4-10-20		1190.95		1190.95		63.1	5.6
2-8-20		846.07		846.07		68.0	7.0
3-8-20		643.89		643.89		63.9	6.5
2-10-25		834.34		834.23		75.0	24.3
3-10-25		820.12		820.12		75.3	2.9
4-10-25		1057.63		1057.63		74.3	7.9
2-8-25		951.59		951.56		74.0	40.1
3-8-25		774.36		774.36		73.7	5.4
5-10-50	1208.43	1207.31	-0.09	1207.31	-0.09	341.2	36.9
5-8-50	1171.35	1164.64	-0.57	1162.44	-0.76	353.2	107.6
5-10-75	1732.33	1723.41	-0.51	1721.47	-0.63	691.0	406.3
5-15-75	1491.31	1483.42	-0.53	1483.14	-0.55	634.7	392.6
5-10-100	2238.7	2183.31	-2.47	2178.35	-2.70	922.4	584.5
5-20-100	2053.12	2048.35	-0.23	2035.37	-0.86	755.1	585.1
5-10-150	1307.19	1282.13	-1.92	1274.44	-2.51	1040.0	710.1
5-20-150	1266.83	1254.38	-0.98	1235.86	-2.44	1062.9	829.4
5-10-200	1822.5	1776.47	-2.53	1766.46	-3.07	1690.0	1079.6
5-20-200	2604.56	2557.05	-1.82	2531.21	-2.82	1608.0	1157.7
Restricted average			-1.17		-1.64	909.85	588.98
Total average						327.77	193.83

Table 7: Results of the ALNS on the instances of set I_3

Instance	z_{BKS}	z_{ALNS}	*gap*
20-5-1	89075	89075	0.00
20-5-1b	61863	61863	0.00
20-5-2	85290	**84478**	-0.95
20-5-2b	60838	60838	0.00
50-5-1	134855	**130843**	-2.98
50-5-1b	101530	101530	0.00
50-5-2	132159	**131825**	-0.25
50-5-2b	110547	**110332**	-0.19
50-5-2BIS	122654	**122599**	-0.04
50-5-2bBIS	105776	**105696**	-0.08
50-5-3	128379	128379	0.00
50-5-3b	104006	104006	0.00
100-5-1	320130	**318761**	-0.43
100-5-1b	258205	**256878**	-0.51
100-5-2	234179	**231305**	-1.23
100-5-2b	195426	**194729**	-0.36
100-5-3	245944	**244194**	-0.71
100-5-3b	195254	**194110**	-0.59
100-10-1	358939	**353133**	-1.62
100-10-1b	302584	**297167**	-1.79
100-10-2	306303	**305154**	-0.38
100-10-2b	264389	**263876**	-0.19
100-10-3	313249	**310200**	-0.97
100-10-3b	266383	**261796**	-1.72
200-10-1	554598	**549718**	-0.88
200-10-1b	452286	**445802**	-1.43
200-10-2	502173	**498199**	-0.79
200-10-2b	425311	**423031**	-0.54
200-10-3	533732	**531548**	-0.41
200-10-3b	418800	**402130**	-3.98
Average			-0.77

Table 8: Best known results for the set Prodhon

Instance	z_{BKS}	z_{ALNS}	*gap*
25-5N	80370	80370	0.00
25-5Nb	64562	64562	0.00
25-5MN	78947	78947	0.00
25-5MNb	64438	64438	0.00
50-5N	138126	**137815**	-0.23
50-5Nb	111062	**110094**	-0.87
50-5MN	123484	123484	0.00
50-5MNb	105401	105401	0.00
50-10N	116132	**115725**	-0.35
50-10Nb	87315	87315	0.00
50-10MN	135748	**135519**	-0.17
50-10MNb	110613	110613	0.00
100-5N	196910	**193228**	-1.87
100-5Nb	159086	**158927**	-0.10
100-5MN	207119	**204682**	-1.18
100-5MNb	166115	**165744**	-0.22
100-10N	215792	**210449**	-2.48
100-10Nb	156401	**155489**	-0.58
100-10MN	205964	**201275**	-2.28
100-10MNb	170706	**170625**	-0.05
200-10N	353685	**347395**	-1.78
200-10Nb	262072	**256171**	-2.25
200-10MN	332345	**324006**	-2.51
200-10MNb	292523	**287076**	-1.86
Average			-0.78

Table 9: Best known results for the set Nguyen

Instance	I_1		I_2		I_3	
	z_{BKS}	z_{ALNS}	z_{BKS}	z_{ALNS}	z_{BKS}	z_{ALNS}
8x3x2		**575.70**		**575.70**		**578.33**
8x4x2		**549.34**		**604.13**		**450.71**
9x3x2		**878.69**		**386.15**		**454.63**
10x4x2		**806.72**		**629.38**		**540.61**
10x5x3		**696.94**		**551.45**		**745.48**
10x8x3		**596.56**		**504.20**		**412.91**
15x10x2		**732.48**		**709.10**		**605.11**
15x10x3		**686.71**		**777.49**		**546.61**
15x4x2		**1064.52**		**827.81**		**688.87**
15x5x3		**933.75**		**1075.22**		**1001.28**
15x8x3		**730.36**		**652.58**		**578.23**
20x10x2		**937.40**		**766.24**		**744.82**
20x10x3		**761.28**		**744.26**		**728.17**
20x10x4		**1149.72**		**793.00**		**1190.95**
20x8x2		**1029.59**		**772.29**		**846.07**
20x8x3		**848.31**		**758.06**		**643.89**
25x10x2		**1030.40**		**961.59**		**834.23**
25x10x3		**1062.30**		**987.57**		**820.12**
25x10x4		**1607.94**		**1125.56**		**1057.63**
25x8x2		**950.87**		**912.02**		**951.56**
25x8x3		**870.69**		**979.62**		**774.36**
50x10x5	1207.31	**1132.63**	1265.73	**1256.44**	1208.43	**1207.31**
50x8x5	1171.69	**1162.44**	1123.42	**1121.13**	1171.35	**1162.44**
75x10x5	1561.5	**1540.23**	1718.25	**1691.15**	1732.33	**1721.47**
75x15x5	1700.32	**1686.21**	1751.14	**1742.25**	1491.31	**1483.14**
100x10x5	2192.14	**2124.9**	2290.64	**2231.21**	2238.7	**2178.35**
100x20x5	1989.48	**1973.08**	2039.25	**1996.34**	2053.12	**2035.37**
150x10x5	1953.55	**1883.44**	1768.79	**1728.05**	1307.19	**1274.44**
150x20x5	1905.81	**1869.53**	1664.2	**1630.29**	1266.83	**1235.86**
200x10x5	2601.33	**2443.80**	2292.47	**2147.51**	1822.5	**1766.46**
200x20x5	2307.53	**2219.54**	2097.74	**2049**	2604.56	**2531.21**

Table 10: Best known results for the sets I_1, I_2 and I_3

Instance	z_{UB}	z_{lr}	gap_{lr}	t_{lr}	z	gap	t	#nodes
ppw-20x5-1a	89075	82642.7	7.22	0.26	**89075**	**0.00**	2.94	219
ppw-20x5-1b	61863	61793.2	0.11	0.19	**61863**	**0.00**	0.24	4
ppw-20x5-2a	84478	79253.1	6.18	0.23	**84478**	**0.00**	4.91	350
ppw-20x5-2b	60838	58355.9	4.08	0.23	**60838**	**0.00**	0.34	4
ppw-50x5-1a	130843	116292	11.12	18.51	129072	1.35	7328.87	5258
ppw-50x5-1b	101530	94577	6.85	2.24	**101530**	**0.00**	5692.59	10952
ppw-50x5-2a	131825	123481	6.33	5.76	**131825**	**0.00**	2746.50	9442
ppw-50x5-2b	110332	104488	5.30	2.14	**110332**	**0.00**	71.93	713
ppw-50x5-2Bis	122599	117815	3.90	22.69	122014	0.48	7269.58	8202
ppw-50x5-2bBis	105696	88717.5	16.06	3.11	90158.5	14.70	7352.63	7338
ppw-50x5-3a	128379	119734	6.73	8.14	127291	0.85	7293.22	5121
ppw-50x5-3b	104006	101702	2.22	2.23	**104006**	**0.00**	41.26	499
ppw-100x5-1a	318761	301246	5.49	755.63	311157	2.39	7349.25	502
ppw-100x5-1b	256878	241311	6.06	256.24	251659	2.03	7434.04	129
ppw-100x5-2a	231305	221840	4.09	191.68	228183	1.35	7390.39	750
ppw-100x5-2b	194729	189324	2.78	34.27	193423	0.67	7442.73	976
ppw-100x5-3a	244194	220717	9.61	60.55	240846	1.37	7332.84	1190
ppw-100x5-3b	194110	183744	5.34	26.52	192562	0.80	7344.08	400
ppw-100x10-1a	353133	306764	13.13	277.90	325239	7.90	7444.96	98
ppw-100x10-1b	297167	261305	12.07	80.44	274956	7.47	7349.33	134
ppw-100x10-2a	305154	272802	10.60	329.10	283189	7.20	7415.92	596
ppw-100x10-2b	263876	239525	9.23	34.70	247256	6.30	7340.13	2903
ppw-100x10-3a	310200	270110	12.92	352.98	286702	7.58	7411.82	113
ppw-100x10-3b	261796	235570	10.02	45.30	244278	6.69	7421.63	523
ppw-200x10-1a	549718	484092	11.94	5029.15	484095	11.94	7219.28	1
ppw-200x10-1b	445802	402796	9.65	1675.14	409852	8.06	7397.27	4
ppw-200x10-2a	498199	479140	3.83	7211.65	479140	3.83	7211.65	1
ppw-200x10-2b	423031	409460	3.21	1449.60	411263	2.78	7366.31	2
ppw-200x10-3a	531548	487204	8.34	7211.77	487204	8.34	7211.77	1
ppw-200x10-3b	402130	389009	3.26	4342.03	389011	3.26	7253.99	1
Average			7.26	981.01		3.58	5671.41	

Table 11: Results of the B&C algorithm on the set Prodhon

Instance	z_{UB}	z_{lr}	gap_{lr}	t_{lr}	z	gap	t	#nodes
25-5N	80370	75556.1	5.99	0.33	**80370**	**0.00**	0.63	2
25-5Nb	64562	63587.4	1.51	0.17	**64562**	**0.00**	0.28	1
25-5MN	78947	72134.2	8.63	0.18	**78947**	**0.00**	0.75	18
25-5MNb	64438	63820.3	0.96	0.34	**64438**	**0.00**	0.60	2
50-5N	137815	132439.0	3.90	2.38	**137815**	**0.00**	1127.68	5474
50-5Nb	110094	98365.0	10.65	2.03	**110094**	**0.00**	867.85	2941
50-5MN	123484	110213.0	10.75	4.95	**123484**	**0.00**	48.97	195
50-5MNb	105401	101104.0	4.08	3.09	**105401**	**0.00**	48.67	290
50-10N	115725	101855.0	11.99	3.50	115236	0.42	7292.11	13875
50-10Nb	87315	81064.3	7.16	2.96	**87315**	**0.00**	1494.72	11278
50-10MN	135519	122272.0	9.78	3.78	**135519**	**0.00**	57.77	258
50-10MNb	110613	102301.0	7.51	2.76	**110613**	**0.00**	24.10	113
100-5N	193228	174730.0	9.57	150.00	183459	5.06	7373.66	944
100-5Nb	158927	145960.0	8.16	20.84	154192	2.98	7446.73	2090
100-5MN	204682	176807.0	13.62	65.74	194800	4.83	7465.38	708
100-5MNb	165744	153433.0	7.43	18.18	160204	3.34	7417.81	1000
100-10N	210449	178551.0	15.16	148.71	194243	7.70	7471.49	294
100-10Nb	155489	143996.0	7.39	44.02	152090	2.19	7552.96	1061
100-10MN	201275	174709.0	13.20	55.40	192092	4.56	7480.88	1028
100-10MNb	170625	151322.0	11.31	31.09	166106	2.65	7428.34	1243
200-10N	347395	303348.0	12.68	6558.92	303348	12.68	7227.58	1
200-10Nb	256171	231196.0	9.75	1385.73	235194	8.19	7471.48	11
200-10MN	324006	290368.0	10.38	2583.05	294937	8.97	7356.50	7
200-10MNb	287076	249830.0	12.97	1312.18	259423	9.63	7440.13	10
Average			8.94	516.68		3.05	4170.71	

Table 12: Results of the B&C algorithm on the set Nguyen

Instance	z_{UB}	z_{lr}	gap_{lr}	t_{lr}	z	gap	t	#nodes
8x3x2	575.70	575.70	0.00	0.01	**575.70**	**0.00**	0.01	1
8x4x2	549.34	534.25	2.75	0.01	**549.34**	**0.00**	0.04	2
9x3x2	878.69	874.49	0.48	0.02	**878.69**	**0.00**	0.05	1
10x4x2	806.72	752.79	6.69	0.03	**806.72**	**0.00**	0.05	1
10x5x3	696.94	660.28	5.26	0.07	**696.94**	**0.00**	0.12	3
10x8x3	596.56	538.62	9.71	0.18	**596.56**	**0.00**	0.35	14
15x10x2	732.48	673.23	8.09	0.52	**732.48**	**0.00**	1.46	43
15x10x3	686.71	601.28	12.44	0.30	**686.71**	**0.00**	1.04	49
15x4x2	1064.52	928.80	12.75	0.21	**1064.52**	**0.00**	0.55	44
15x5x3	933.75	829.52	11.16	0.14	**933.75**	**0.00**	0.39	23
15x8x3	730.36	667.28	8.64	0.21	**730.36**	**0.00**	0.64	31
20x10x2	937.40	817.63	12.78	0.45	**937.40**	**0.00**	322.33	12179
20x10x3	761.28	689.74	9.40	0.81	**761.28**	**0.00**	2.06	26
20x10x4	1149.72	967.72	15.83	0.51	**1149.72**	**0.00**	97.17	5783
20x8x2	1029.59	874.65	15.05	1.45	1014.98	1.42	7306.54	34299
20x8x3	848.31	787.66	7.15	0.29	**848.31**	**0.00**	817.87	20863
25x10x2	1030.40	855.45	16.98	1.96	1029.18	0.12	7267.93	55055
25x10x3	1062.30	931.48	12.31	1.25	1049.72	1.18	7265.92	30758
25x10x4	1607.94	1347.46	16.20	1.56	1535.34	4.52	7299.10	20642
25x8x2	950.87	871.45	8.35	0.65	**950.87**	**0.00**	151.67	5547
25x8x3	870.69	775.09	10.98	1.03	**870.69**	**0.00**	370.89	9450
50x10x5	1132.63	935.55	17.40	51.01	1104.62	2.47	7404.23	4849
50x8x5	1162.44	991.17	14.73	29.29	1146.03	1.41	7388.90	5653
75x10x5	1540.23	1340.44	12.97	137.40	1433.71	6.92	7453.14	1575
75x15x5	1686.21	1427.27	15.36	141.51	1582.25	6.17	7499.81	577
100x10x5	2124.90	1855.10	12.70	942.70	1954.54	8.02	7417.19	55
100x20x5	1973.08	1692.66	14.21	580.55	1820.06	7.76	7437.94	179
150x10x5	1883.44	1594.70	15.33	2070.76	1709.42	9.24	7436.38	13
150x20x5	1869.53	1523.69	18.50	1850.22	1576.52	15.67	7335.10	10
200x10x5	2443.80	2188.41	10.45	5095.85	2190.91	10.35	7254.42	3
200x20x5	2219.54	1852.28	16.55	7204.66	1852.28	16.55	7204.66	1
Average			11.33	584.37		2.96	3378.64	

Table 13: Results of the B&C algorithm on the set I_1

Instance	z_{UB}	z_{lr}	gap_{lr}	t_{lr}	z	gap	t	#nodes
8x3x2	575.70	575.70	0.00	0.01	**575.70**	**0.00**	0.02	1
8x4x2	604.13	596.68	1.23	0.01	**604.13**	**0.00**	0.02	4
9x3x2	386.15	386.15	0.00	0.00	**386.15**	**0.00**	0.01	1
10x4x2	629.38	602.86	4.21	0.06	**629.38**	**0.00**	0.09	2
10x5x3	551.45	551.45	0.00	0.02	**551.45**	**0.00**	0.02	1
10x8x3	504.20	503.16	0.21	0.09	**504.20**	**0.00**	0.10	1
15x10x2	709.10	651.21	8.16	0.26	**709.10**	**0.00**	1.39	156
15x10x3	777.49	702.15	9.69	0.29	**777.49**	**0.00**	1.54	84
15x4x2	827.81	799.99	3.36	0.11	**827.81**	**0.00**	0.23	6
15x5x3	1075.22	1048.49	2.49	0.18	**1075.22**	**0.00**	0.36	8
15x8x3	652.58	591.31	9.39	0.28	**652.58**	**0.00**	1.49	208
20x10x2	766.24	684.32	10.69	0.48	**766.24**	**0.00**	3.78	237
20x10x3	744.26	688.77	7.46	0.38	**744.26**	**0.00**	1.48	54
20x10x4	793.00	735.41	7.26	0.40	**793.00**	**0.00**	1.05	22
20x8x2	772.29	698.42	9.57	0.37	**772.29**	**0.00**	5.64	549
20x8x3	758.06	679.75	10.33	0.29	**758.06**	**0.00**	6.00	833
25x10x2	961.59	857.44	10.83	1.67	952.83	0.91	7299.78	23112
25x10x3	987.57	847.05	14.23	1.26	930.24	5.81	7257.52	28573
25x10x4	1125.56	948.37	15.74	1.18	**1125.56**	**0.00**	2716.10	39282
25x8x2	912.02	838.39	8.07	0.84	**912.02**	**0.00**	1272.05	36427
25x8x3	979.62	908.74	7.24	0.69	**979.62**	**0.00**	8.30	708
50x10x5	1256.44	1117.50	11.06	11.11	1222.26	2.72	7442.70	4325
50x8x5	1121.13	990.18	11.68	7.01	1098.02	2.06	7376.86	7657
75x10x5	1691.15	1520.07	10.12	58.28	1597.48	5.54	7393.84	3343
75x15x5	1742.25	1476.73	15.24	84.91	1661.72	4.62	7516.68	487
100x10x5	2231.21	2020.79	9.43	771.23	2073.51	7.07	7451.59	241
100x20x5	1996.34	1778.54	10.91	222.12	1849.18	7.37	7583.58	234
150x10x5	1728.05	1463.90	15.29	475.29	1554.46	10.05	7530.95	51
150x20x5	1630.29	1423.06	12.71	578.49	1489.41	8.64	7486.50	76
200x10x5	2147.51	1912.21	10.96	1976.60	1931.23	10.07	7286.86	13
200x20x5	2049.01	1780.40	13.11	1298.32	1803.03	12.00	7419.87	17
Average			8.41	177.17		2.48	3002.14	

Table 14: Results of the B&C algorithm on the set I_2

Instance	z_{UB}	z_{lr}	gap_{lr}	t_{lr}	z	gap	t	#nodes
8x3x2	578.33	539.21	6.76	0.02	**578.33**	**0.00**	0.04	2
8x4x2	450.71	422.33	6.30	0.02	**450.71**	**0.00**	0.05	1
9x3x2	454.63	454.63	0.00	0.00	**454.63**	**0.00**	0.01	1
10x4x2	540.61	525.79	2.74	0.03	**540.61**	**0.00**	0.07	6
10x5x3	745.48	681.84	8.54	0.05	**745.48**	**0.00**	0.13	15
10x8x3	412.91	412.91	0.00	0.02	**412.91**	**0.00**	0.03	1
15x10x2	605.11	539.45	10.85	0.41	**605.11**	**0.00**	1.25	67
15x10x3	546.61	495.00	9.44	0.46	**546.61**	**0.00**	1.64	175
15x4x2	688.87	636.63	7.58	0.05	**688.87**	**0.00**	0.27	59
15x5x3	1001.28	897.66	10.35	0.26	**1001.28**	**0.00**	1.06	86
15x8x3	578.23	520.34	10.01	0.34	**578.23**	**0.00**	0.86	34
20x10x2	744.82	692.91	6.97	0.53	**744.82**	**0.00**	6.34	536
20x10x3	728.17	648.27	10.97	0.41	**728.17**	**0.00**	4.18	231
20x10x4	1190.95	1004.67	15.64	0.80	**1190.95**	**0.00**	334.74	9796
20x8x2	846.07	776.12	8.27	0.66	**846.07**	**0.00**	37.17	2153
20x8x3	643.89	593.91	7.76	0.45	**643.89**	**0.00**	1.63	71
25x10x2	834.23	768.96	7.82	1.31	**834.23**	**0.00**	35.81	2082
25x10x3	820.12	760.13	7.32	1.24	**820.12**	**0.00**	2.79	37
25x10x4	1057.63	873.48	17.41	0.44	**1057.63**	**0.00**	14.25	389
25x8x2	951.56	799.63	15.97	2.67	910.69	4.30	7314.28	22280
25x8x3	774.36	711.47	8.12	0.95	**774.36**	**0.00**	105.28	4559
50x10x5	1207.31	1062.64	11.98	16.52	1178.74	2.37	7477.56	3751
50x8x5	1162.44	991.17	14.73	29.21	1146.05	1.41	7391.33	5686
75x10x5	1721.47	1528.73	11.20	44.68	1638.52	4.82	7394.76	2406
75x15x5	1483.14	1310.67	11.63	48.08	1411.48	4.83	7628.68	541
100x10x5	2178.35	1984.72	8.89	291.52	2057.97	5.53	7505.30	220
100x20x5	2035.37	1827.62	10.21	171.99	1889.20	7.18	7553.46	331
150x10x5	1274.44	1149.28	9.82	169.20	1198.72	5.94	7463.36	149
150x20x5	1235.86	1102.28	10.81	329.76	1162.03	5.97	7497.05	110
200x10x5	1766.46	1593.59	9.79	539.10	1659.98	6.03	7409.29	54
200x20x5	2531.21	2273.59	10.18	954.70	2304.54	8.96	7418.42	71
Average			9.29	84.06		1.85	2664.55	

Table 15: Results of the B&C algorithm on the set I_3