The two-echelon location-routing problem with time windows: Formulation, branch-and-price, and clustering

Mohammad Saleh Farham, Cem Iyigun, Haldun Süral
Industrial Engineering Department, Middle East Technical University, Ankara, Turkey 06800, pharham@tuta.io, iyigun@metu.edu.tr, hsural@metu.edu.tr,

In this study, we consider the two-echelon location-routing problem with time windows (2E-LRPTW) to address the strategic and tactical decisions of the urban freight transportation. In the first echelon, freights are delivered from city distribution centers (CDCs) to intermediate facilities, called satellites, in large batches. In the second echelon, goods are consolidated into smaller vehicles to be delivered to the customers. Therefore, given a set of candidate CDC locations, a set of candidate satellite locations, and a set of customers, the 2E-LRPTW seeks the minimum total transportation cost consisting of CDC and satellite opening costs as well as first and second echelon vehicle routing costs such that all costumer demands are satisfied. The problem is constrained by CDC, satellite, and vehicle capacities as well as customer time windows. We provide the set-partitioning formulation of the problem and propose an exact solution approach based on column generation. To tackle larger problems, we develop two heuristics based on hierarchical structure of the problem. Our computational study shows the performance of three approaches on solving a large set of problem instances with different sizes and characteristics and highlights the benefits of using clustering-based heuristics to solve large-size instances.

Key words: urban freight transportation; decomposition; branch-and-price; column generation; constrained clustering
History:

1. Introduction

In recent years, planning and deploying urban freight transportation systems have received increasing attention due to economic and environmental concerns. The freight transportation problems concern distribution of one or more commodities from a set of distribution centers to a set of demand points. The aim is to “optimize” the flow of goods through the existing network and “improve” the network by choosing the best configuration of the facilities and transportation modes

(Ambrosino and Scutellà 2005). The optimization problem becomes a critical issue in urban areas, where the transportation activity affects the city environment and citizens' quality of life. In addition, road traffic and capacity, vehicle weight regulations, and customer access time windows bring other challenges to the freight delivery in city-centers.

From a network design stand point, urban transportation systems are either *single-* or *multi-echelon*. In the one-echelon systems, freight is delivered from the selected *city distribution centers* (CDCs) to the customers without any intermediate activities. Such systems are used in small cities with limited number of carriers and shippers (Crainic, Ricciardi, and Storchi 2009). Taniguchi and Thompson (2002) and Crainic, Ricciardi, and Storchi (2004) introduce two-echelon distribution systems as a solution to simultaneously reduce pollution, traffic congestion and operating cost of the freight transportation in large cities. In the first echelon, large batches of freight are delivered from CDCs to intermediate facilities called *satellites*. While CDCs are large facilities located on the outskirts of the city, satellites are small inner-city locations where no inventory or staging is possible. In the second echelon, goods are sorted and consolidated into environmental-friendly vehicles to be delivered to the customers in city-center areas. In urban environment, local authorities usually impose restrictive regulations on customer access times or on the weight of delivery vehicles.

In this study, we consider the two-echelon location-routing problem with capacity and time windows constraints (2E-LRPTW) to address strategic and tactical decisions of the two-echelon urban freight transportation systems. The 2E-LRPTW aims to minimize the total transportation cost consisting of facility opening, vehicle utilization, and vehicle traveling costs such that all customer demands are satisfied. It decides on the number and location of CDC and satellite platforms, the number of vehicles used in each echelon, and the vehicle routes and schedules. The problem is defined under facility and vehicle capacity constraints as well as *hard time windows*, where serving a customer is only possible during a specific time interval.

The 2E-LRPTW is one of the core problems in urban freight transportation as it covers both strategic-level decisions (i.e. facility locations) and tactical-level planning (i.e. vehicle routing and scheduling). However, the literature lacks extensive studies on the 2E-LRPTW formulations and exact solution approaches. The related problems in the literature can be classified into two groups: the two-echelon vehicle routing problem (2E-VRP) and the two-echelon location-routing problem (2E-LRP). In 2E-VRPs, the location of available first and second echelon facilities are given, hence no location decisions are made. The first echelon addresses CDC(s)-to-satellites routing problem, while satellites-to-customers delivery routes are decided in the second echelon. Facility and vehicle capacities are common restrictions in the 2E-VRPs. Crainic et al. (2010) considered the 2E-VRP in city logistics and analyzed the effect of customer distribution, facility locations, number and accessibility of satellites, and associated distribution cost on the transportation cost through computational experiments. The results indicate that the 2E-VRP approach leads to lower overall cost compared to the classical VRP in most cases, in particular, when the CDC is located externally with respect to the customer area and a certain number of satellites are located close to the demand points. Perboli, Tadei, and Vigo (2011) provided a mixed-integer linear programming (MIP) formulation and valid inequalities for the 2E-VRP with one CDC. The authors also proposed two math-based heuristics to solve large problem instances more efficiently. Baldacci et al. (2013) proposed an exact method for the similar problem based on decomposition of the 2E-VRP into several multi-depot capacitated VRP with side constraints. The reader is referred to Crainic et al. (2011), Hemmelmayr, Cordeau, and Crainic (2012), Crainic et al. (2013), Jepsen, Spoorendonk, and Ropke (2013), Santos, Mateus, and da Cunha (2015), Breunig et al. (2016), Zhou et al. (2018) for other studies on the 2E-VRP and its variants and solution approaches. Dellaert et al. (2019) is the only study that considers the 2E-VRP with time windows (2E-VRPTW) arising in city logistics and proposes an exact approach based on the branch-and-price framework.

Achieving decision makers' objectives under the environmental and time window constraints is a challenging issue in urban freight transportation problems. Capacity and temporal constraints make these optimization problems intractable when dealing with realistic situations. Boccia et al. (2010) decomposed the 2E-LRP into a capacitated facility location problem and a multi-depot VRP in each echelon. The authors developed a tabu search heuristic to solve problem instances with up to 200 customers and 20 satellites. Contardo, Hemmelmayr, and Crainic (2012) considered the 2E-LRP with several capacitated first and second echelon facilities and developed an exact branch-and-cut algorithm as well as an adaptive large neighborhood search heuristic. The authors claimed that the 2E-LRP can be decomposed into two different LRPs, one at each echelon, connected through the satellite nodes. The literature of the 2E-LRP contains various heuristic solution methods such as greedy randomize adaptive search (Nguyen, Prins, and Prodhon 2010, 2012b), iterative local search (Nguyen, Prins, and Prodhon 2012a), adaptive large neighborhood search (Hemmelmayr, Cordeau, and Crainic 2012), variable neighborhood search (Schwengerer, Pirkwieser, and Raidl 2012), and route construction methods based on customer clustering (Rahmani, Cherif-Khettaf, and Oulamara 2016). The largest 2E-LRP instances that could be solved in a reasonable computing time by an exact approach contain 1 CDCs, 10 satellites, and 50 customers, while heuristic solutions are found for problem instances with 5 CDCs, 20 satellites, and 200 customers (Cuda, Guastaroba, and Speranza 2015). A comprehensive overview on the 2E-LRPs is provided by Prodhon and Prins (2014), Drexl and Schneider (2015).

Although access time window constraints are important in planing urban freight transportation, only a few studies consider the temporal aspect in the two-echelon routing problems. Crainic, Ricciardi, and Storchi (2009) provided a generic path-based model for the 2E-VRPTW considering multiple tours, synchronization, and heterogeneous vehicles. Grangier et al. (2016) formulated a 2E-VRPTW under time synchronization constraints. In this problem, second echelon vehicles can perform multiple trips and, since satellites have no storage capacity, must be synchronized with the first echelon vehicles every time they start their service from a satellite location. Their objective function minimizes the fleet size and the travel costs incurred to satisfy customer demands. Li et al. (2016) formulated a 2E-VRPTW where vehicle routes on different echelons are interacted by time constraints. A second echelon vehicle cannot leave the intermediate facility before a first echelon vehicle delivers the required product. The waiting times, along with the transportation costs, are minimized in the objective function. The authors present a two-stage solution algorithm that incorporates a savings-based heuristic followed by a local search phase. Anderluh, Hemmelmayr, and Nolz (2017) considered a 2E-VRP in which the inner-city delivery on the second echelon is performed by cargo bikes. After loading in a satellite location, the cargo bikes perform their delivery and when they have to reload, they move again to a satellite. In this problem, first and second echelon vehicles must meet in a synchronized way at the same time at the same physical satellite, while their waiting times are minimized. The authors developed a greedy randomized adaptive search heuristic to solve the problem. Gündüz (2015) formulated the 2E-LRPTW where the location of CDCs are known. The author presented a tabu search algorithm and compared its result with a sequential location-allocation-routing approach on a set of instances containing up to 4 depots, 50 candidate intermediate facilities, and 400 customers. Bala, Brcanov, and Gvozdenović (2017) studied a 2E-LRPTW arising in delivering perishable goods. They considered a production schedule system where availability of products at facility locations affects origin and departure time of the routes. The authors proposed a heuristic solution approach and solve instances with up to 2000 customers and four products. Wang et al. (2018) introduced a bi-objective model for the 2E-LRPTW incorporating vehicle routes in both echelons. In addition to conventional cost minimization objective, their model seeks for maximum customer satisfaction measured by customers' demand and delivery times. The authors proposed a three-step heuristic where a k-means clustering technique is used at the initial step to group customers base on their preferences. In the second step, the generated clusters are used to locate facilities. The last step applies a genetic algorithm to find vehicle routes stemmed from a located facility and serving the customers assigned to that facility.

To the best of our knowledge, there is no study on formulating and solving the two-echelon location-routing problem under capacity and time window constraints to optimality. This study is the first that presents an exact method to solve the 2E-LRPTW. We consider a variant of the 2E-LRPTWs where the first echelon vehicles perform direct shipments from open CDCs to selected satellites forming CDC–satellite–CDC routes. This is a valid assumption in urban areas where the first-echelon network consists of one or two CDCs, a limited number of satellites, and high capacity roads far from city centers (e.g. ring roads). While considering the strategic-level decisions in the first echelon, the introduced 2E-LRPTW incorporates the tactical/operational level planing in the second echelon where high concerns about transportation cost, time window feasibility, vehicle utilization, and environmental impacts exit. The second echelon consists of satellites and customers nodes. Vehicles in the second echelon start their route from a satellite location, visit a set of customers, and end their trip by returning to the same satellite location. We present a path-based MIP formulation of the 2E-LRPTW and develop exact and heuristic solution approaches. The exact approach is based on branch-and-price (BP) algorithm that is one of the most successful solution approaches for the constrained routing problems in the literature (see Baldacci, Mingozzi, and Roberti 2012, Dabia et al. 2013, Contardo, Desaulniers, and Lessard 2015, Pecin et al. 2017, Farham, Süral, and Iyigun 2018). The algorithm decomposes the original problem into two: the master problem and the subproblem. The master problem of the 2E-LRPTW consists of the first-echelon decisions, i.e. facility location and CDC-to-satellite vehicle routes, as well as decision on the second echelon routes. In order to generate candidate routes in the second echelon, a number of subproblems is solved. A subproblem corresponds to a constrained shortest path problem which is NP-hard. Different enhancement techniques are implemented to improve the overall performance of the proposed BP.

Vehicle routing problems are strongly NP-hard as they generalize the well-known *traveling salesman problem*. The 2E-LRPTW adds more complexity to the classical VRP by incorporating more decisions (e.g. facility location) and constraints (e.g. capacity and time window limitations). Therefore, even the most efficient exact approaches may fail to solve large-size problem instances in reasonable amount of time. In order to find the solution of the large-size 2E-LRPTW instances, we propose two heuristics. Both heuristics benefit from decomposing the problem based on its strategic and tactical level decisions. The first heuristic, called *top-to-bottom approach*, solves an optimization problem to determine the strategic-level decisions first, and then executes the proposed BP on the reduced problem to find the vehicle routes in both echelons. The second heuristic, on the other hand, starts by determining the domain of the complicated tactical/operational-level decisions, and fixes the remaining decisions later. In this heuristic, called *bottom-to-top approach*, we design and implement a novel constrained clustering technique to group the customers that a second echelon vehicle might visit. We form a one-to-one relation between a cluster and a feasible route. Therefore, time window and capacity restrictions are satisfied while shaping the clusters. Once candidate second echelon routes are generated, a mixed-integer linear program is solved to determine facility locations and vehicle routes.

The contribution of this paper is in four folds. (i) We present a path-based MIP formulation of the 2E-LRPTW as a core problem in designing and planing urban freight transportation systems for large cities. (ii) A branch-and-price solution approach is presented to find the exact solution of the problem. Different enhancements techniques are proposed to improve the performance of the algorithm. (iii) We develop two heuristics based on the hierarchical decisions of the problem to approximate the solution efficiently. A new clustering-based heuristic is proposed to group the customers into candidate vehicle routes. The numerical results indicate that the clustering-based heuristic saves a significant amount of time to solve the problem instances without sacrificing much of the solution quality. (iv) Extensive computational experiments are conducted to study the performance of the proposed solution approaches and the effect of different instance characteristics on the solution of the 2E-LRPTW.

The reminder of this article is organized as follows. §2 presents mathematical formulation of the problem. In §3, we introduce the exact approach to solve the formulated 2E-LRPTW. We propose our heuristic solution algorithms in §4. Problem test instances and the computational study of the proposed solution approaches are provided in §5. Finally, we conclude and provide future research perspectives in §6.

2. Problem Formulation

In this section, we formulate two problems arising in two-echelon urban freight transportation systems. The first problem, namely the two-echelon facility location problem with time windows (2E-FLPTW), focuses on the strategic level decisions of the system and is used as a basis for the second problem, the 2E-LRPTW. In addition to the decisions of the 2E-FLPTW, the 2E-LRPTW covers tactical level decisions about the vehicle routes and schedules in the second echelon.

2.1. Two–Echelon Facility Location Problem with Time Windows

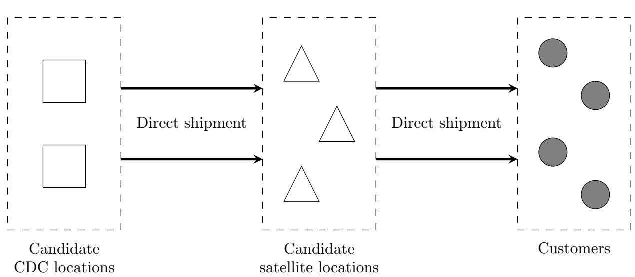


Figure 1 2E-FLPTW.

Facility location problems are well-known optimization problems studied in supply chain and logistics literature (see Klose and Drexl 2005, Farahani et al. 2014). Here, we formulate a two-echelon facility location problem with facility and vehicle capacities as well as customer time windows. The 2E-FLPTW seeks (i) location of the open CDC(s), (ii) location of the open satellite(s), (iii) the flow from open CDCs to open satellites, and (iv) assignment of customers to open satellites, such that the total cost of facility location and freight distribution is minimized. The 2E-FLPTW is illustrated in Figure 1.

The underlying transportation network consists of three sets of nodes: set $\mathcal{I}$ indicating candidate CDC locations, set $\mathcal{J}$ consisting of candidate satellite locations, and set $\mathcal{K}$ denoting customer nodes. Let $\mathcal{M} = \mathcal{I} \cup \mathcal{J}$ and $\mathcal{N} = \mathcal{J} \cup \mathcal{K}$ be the set of first and second echelon nodes, respectively. Each customer $k \in \mathcal{K}$ is characterized by a demand D_k, a time window $[A_k, B_k]$, and a nonnegative service time. No time window is considered for CDCs, but a satellite $j \in \mathcal{J}$ can only be accessed during time interval $[0, B_j]$. If a vehicle arrives to a customer location earlier than the time window, it should wait until the time window starts. A facility $m \in \mathcal{M}$ has opening fixed cost F_m and capacity Q_m. A fixed cost F' (F'') and a capacity Q' (Q'') are associated to each first echelon (second echelon) vehicle. We assume that a customer is served by exactly one satellite. However, multiple CDCs can ship freight to one satellite location. For any two nodes m, n of the same echelon, define C_{mn} as the cost of traveling on arc (m, n). The value of C_{mn} depends on the distance, time, or energy consumption of reaching node n from node m, as well as the type of vehicle in use. Let T_{mn} be the sum of setup or service time at node m and traveling time on arc (m, n). We assume (i) a single commodity in the system, (ii) unsplittable customer demands that are all less than or equal to Q'', (iii) no direct service from a CDC to customers, and (iv) unrestricted number of homogeneous vehicles in each echelon. Traveling times and traveling costs can be asymmetric, but both satisfy triangle inequality. Let $C'_{ij} = F' + C_{ij} + C_{ji}$ be the cost of a first echelon route starting from CDC i, visiting satellite j and returning to CDC i. Similarly, let $C''_{jk} = F'' + C_{jk} + C_{kj}$ be the cost of route j–k–j, $\forall j \in \mathcal{J}, k \in \mathcal{K}$ performed by a second-echelon vehicle. In order to determine open facilities, a binary decision variable z_m is defined that takes a value if facility m is used. Let y_{ij} be a non-negative integer decision variable that determines the number of first echelon vehicles traveling from CDC i to satellite j. Non-negative decision variable w_{ij} represents the amount of flow from CDC i to satellite j. Define x_{jk} as a binary variable indicating whether arc (j, k) in the second echelon is used, i.e whether customer k is assigned to satellite j. The mixed-integer linear programming (MIP) formulation of the 2E-FLPTW is as follows.

$$\text{(2E-FLPTW)} \qquad \text{Minimize} \sum_{m \in \mathcal{M}} F_m z_m + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} C'_{ij} y_{ij} + \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} C''_{jk} x_{jk} \tag{1}$$

$$\text{subject to} \sum_{j \in \mathcal{J}} x_{jk} = 1, \qquad \forall k \in \mathcal{K} \tag{2}$$

$$\sum_{k \in \mathcal{K}} D_k x_{jk} \le \sum_{i \in \mathcal{I}} w_{ij}, \qquad \forall j \in \mathcal{J} \tag{3}$$

$$
\begin{aligned}
&(T_{jk}-B_k)\,x_{jk}\le 0, && \forall j\in\mathcal{J},k\in\mathcal{K} && (4)\\
&(\max(T_{jk},A_k)+T_{kj}-B_j)\,x_{jk}\le 0, && \forall j\in\mathcal{J},k\in\mathcal{K} && (5)\\
&\sum_{j\in\mathcal{J}} w_{ij}\le Q_i z_i, && \forall i\in\mathcal{I} && (6)\\
&\sum_{i\in\mathcal{I}} w_{ij}\le Q_j z_j, && \forall j\in\mathcal{J} && (7)\\
&0\le w_{ij}\le Q' y_{ij}, && \forall i\in\mathcal{I},j\in\mathcal{J} && (8)\\
&(T_{ij}-B_j)\,y_{ij}\le 0, && \forall i\in\mathcal{I},j\in\mathcal{J} && (9)\\
&z_m\in\{0,1\}, && \forall m\in\mathcal{M} && (10)\\
&y_{ij}\in\{0,1,2,\cdots\}, && \forall i\in\mathcal{I},j\in\mathcal{J} && (11)\\
&x_{jk}\in\{0,1\}, && \forall j\in\mathcal{J},k\in\mathcal{K}. && (12)
\end{aligned}
$$

The objective function (1) minimizes total transportation cost consisting of CDC and satellite location costs, first echelon CDC-satellite-CDC route costs, and second echelon satellite-customer-satellite route costs. Constraint (2) guarantees that each customer is assigned to exactly one satellite. Constraint (3) ensures that total incoming flow to a satellite location is not less than the total customer demands it serves. Time windows of customers and closing time of satellites on the second echelon are satisfied by (4) and (5). These constraints set a x_{jk} variable to zero if T_{jk} exceeds the closing time of node k, or if the traveling time of route j–k–j exceeds the closing time of node j, $\forall j\in\mathcal{J},k\in\mathcal{K}$. Capacity limit of open CDCs and satellites are satisfied by (6) and (7), respectively. Constraint (8) holds the lower bounds on the flow variables and sets the correct number of first echelon vehicles with respect to their capacity. By constraint (9), closing time of the satellites in the first echelon is satisfied. (10)–(12) are variable domain constraints.

The 2E-FLPTW (1)–(12) contains polynomial number of variables and constraints and it can be solved efficiently with common MIP solvers.

2.2. Two–Echelon Location–Routing Problem with Time Windows

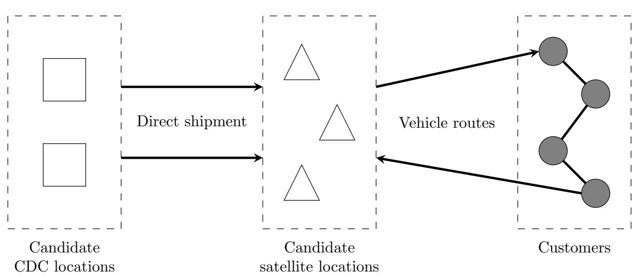


Figure 2 **2E-LRPTW.**

The 2E-LRPTW, illustrated in Figure 2, incorporates detailed decisions on the second echelon freight delivery operation that are excluded in the strategic 2E-FLPTW in §2.1. Adding vehicle routing and scheduling decisions leads to a large number of variables and constraints in the MIP arc-flow formulation of the problem, which makes the problem intractable for standard MIP solvers. The set-partitioning models, on the other hand, have significantly less number of constraints and allow *column generation* approaches to be used. Therefore, they have been commonly studied in vehicle routing and location-routing literature (see Baldacci, Mingozzi, and Roberti 2012, Farham, Süral, and Iyigun 2018). The solution approaches proposed in this study are also based on the set-partitioning formulation of the 2E-LRPTW. Therefore, we provide the set-partitioning formulation of the 2E-LRPTW in this section.

We use notations and variables defined in §2.1. In addition, let $\mathcal{P}_j$ be the set of all feasible second echelon vehicle paths originating and ending at satellite j. A second echelon route is feasible if all the following *route feasibility conditions* (RFCs) hold:

$$\text{RFCs:} \begin{cases} \text{(i)} & \text{route starts and ends at the same satellite node,} \\ \text{(ii)} & \text{each customer is visited exactly once,} \\ \text{(iii)} & \text{serving a customer is started during its time window,} \\ \text{(iv)} & \text{the route is completed before satellite closing time, and} \\ \text{(v)} & \text{vehicle capacity is not exceeded.} \end{cases} \tag{13}$$

Define second echelon arc set $\mathcal{E} = \{(m,n) \in \mathcal{N} \times \mathcal{N}\}$. We exclude satellite-to-satellite arcs from $\mathcal{E}$. Let C_p be the cost of second-echelon path p given by the sum of F'' and the traveling costs of all arcs traversed in the path. Let H_{pk} indicates the number of times customer k is visited in path p. Define λ_p as a binary variable that takes value 1 if and only if path p is selected. The set-partitioning formulation of the 2E-LRPTW is given below.

$$\text{(2E-LRPTW)} \qquad \text{Minimize} \sum_{m\in\mathcal{M}} F_m z_m + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} C'_{ij} y_{ij} + \sum_{j\in\mathcal{J}}\sum_{p\in\mathcal{P}_j} C_p \lambda_p \tag{14}$$

$$\text{subject to (6)–(11)} \tag{15}$$

$$\sum_{j\in\mathcal{J}}\sum_{p\in\mathcal{P}_j} H_{pk}\lambda_p = 1, \qquad \forall k \in \mathcal{K} \tag{16}$$

$$\sum_{p\in\mathcal{P}_j} H_{pk} D_k \lambda_p \le \sum_{i\in\mathcal{I}} w_{ij}, \quad \forall j \in \mathcal{J} \tag{17}$$

$$\lambda_p \in \{0,1\}, \qquad \forall p \in \mathcal{P}_j, j \in \mathcal{J}. \tag{18}$$

The objective function (14) minimizes the total transportation cost consisting of CDC and satellite location costs as well as first and second echelon vehicle routing costs. (15) adds the required first echelon constraints from the 2E-FLPTW model. Constraint (16) guarantees that each customer is visited exactly once. Constraint (17) ensures that total incoming flow to a satellite location is not less than total customer demands it serves. (18) is the integrality constraint.

2.2.1. Valid Inequalities. Farham, Süral, and Iyigun (2018) suggested two valid inequalities for the set-partitioning model of the LRPTW. The first one sets a lower bound on the number of open satellites whereas the second one sets a lower bound on the total number of vehicles required to serve all demands. Define a nonnegative integer variable v_j to indicate the number of vehicle routes from satellite j. Then, we can use valid inequalities (19) and (20), where $\overline{Z}$ ($\overline{V}$) is the minimum number of satellites (second echelon vehicles) required to serve all customers demands. Defining v_j requires additional constraints (21) and (22).

$$\sum_{j\in\mathcal{J}} z_j \geq \overline{Z} \tag{19}$$

$$\sum_{j\in\mathcal{J}} v_j \geq \overline{V} \tag{20}$$

$$v_j = \sum_{p\in\mathcal{P}_j} \lambda_p, \qquad \forall j \in \mathcal{J} \tag{21}$$

$$v_j \in \{0,1,2,\cdots\}, \quad \forall j \in \mathcal{J}. \tag{22}$$

In the following sections, we propose exact and heuristic solution approaches to solve the 2E-LRPTW formulated above.

3. Exact Algorithm

The 2E-LRPTW (14)–(22) can be solved efficiently using standard MIP solvers if $\mathcal{P}_j$ sets are small. However, it is impractical to generate and add all the possible paths in realistic situations. We propose a branch-and-price algorithm (BP) for the 2E-LRPTW formulation. BP embeds column generation (CG) with the branch-and-bound method to solve hard combinatorial optimization problems and is shown to be effective in the context of vehicle routing problems (Desaulniers, Solomon, and Desrosiers 2005). The idea is to initiate the original problem with a limited number of columns and generate new columns as needed instead of enumerating over all path variables. The set-partitioning formulation (14)–(22) with a subset of path variables is called the *restricted master problem* (RMP). In each iteration, BP solves the relaxation of the RMP and finds the dual solution. Next, a number of subproblems, also called as pricing problems, are solved to price out new path variables and extend $\mathcal{P}_j$ sets. If a column with negative reduced cost is found, it is added to the RMP and the relaxed RMP is resolved. Otherwise, the algorithm checks the current solution of the RMP against integrality constraints. If any fractional integer variable exists, a branching rule is applied and the algorithm solves a new RMP. Otherwise, it stops by returning the optimal solution.

In the first step of the proposed BP, the RMP is constructed with initial columns. One way is to obtain trivial columns by solving the 2E-FLPTW (1)–(12) and provide the resulting satellite-customer-satellite paths as the initial columns for the RMP. Another way is to solve the original problem by a fast heuristic to find a solution for the 2E-LRPTW. Any feasible solution can be used to provide initial columns and an upper-bound for the exact algorithm. Providing a good upper-bound for the algorithm in the beginning helps to prune more nodes in the underlying branch-and-bound tree and converge to the optimal solution faster (Farham, Süral, and Iyigun 2018). The outline of the proposed BP for the 2E-LRPTW is given in Figure 3. Below, we describe the main stages of the proposed BP in details.

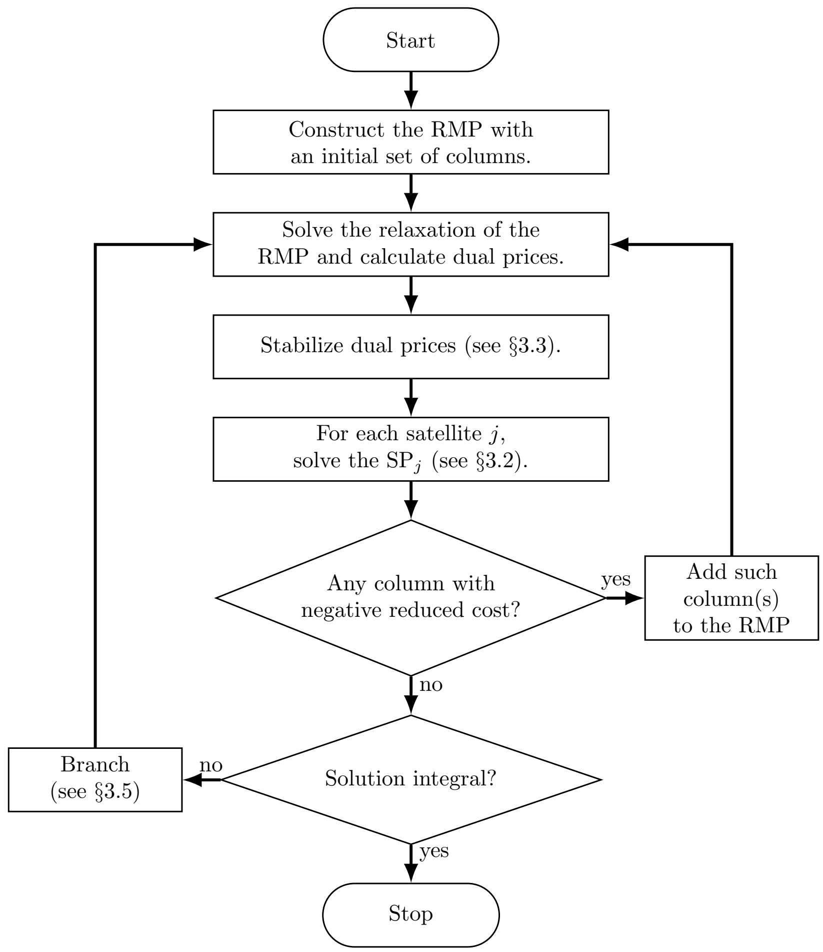


Figure 3 Branch-and-price algorithm for the 2E-LRPTW.

3.1. The Subproblems

In order to generate new path variables for set $\mathcal{P}_j$, we define a subproblem SP_j for all $j \in \mathcal{J}$. Let $\mathcal{E}_j$ be the set of arcs for SP_j. $\mathcal{E}_j$ includes all arcs in $\mathcal{E}$ except the arcs starting/ending at any satellite other than j. Let α_k, β_j, and γ_j be the dual values associated with constraints (16), (17), and (21), respectively. Then, the reduced cost of arc $(m,n) \in \mathcal{E}_j$ for SP_j, denoted by $\widetilde{C}_{jmn}$, is calculated as:

$$\widetilde{C}_{jmn} = \begin{cases} F'' + C_{mn} - \gamma_j, & \text{if } m = j \\ C_{mn} - \alpha_m - \beta_j D_m, & \text{otherwise.} \end{cases} \tag{23}$$

Extend the definition of x_{jk} variables on all second echelon arcs by defining x_{mn} to indicate whether arc $(m,n) \in \mathcal{E}$ is traversed by a vehicle. Define t_k as the arrival time of a vehicle to node n. Therefore, the subproblem for satellite j is formulated as follows.

$$
\begin{aligned}
(\mathrm{SP}_j) \qquad & \text{Minimize} \sum_{(m,n)\in\mathcal{E}_j} \widetilde{C}_{jmn} x_{mn} && (24)\\
& \text{subject to} \sum_{k\in\mathcal{K}} x_{jk} = 1 && (25)\\
& \sum_{m:(m,n)\in\mathcal{E}_j} x_{mn} = \sum_{m:(n,m)\in\mathcal{E}_j} x_{nm}, \quad \forall n \in \mathcal{N} && (26)\\
& \sum_{(m,n)\in\mathcal{E}_j} D_n x_{nm} \le Q'' && (27)\\
& T_{jk} - t_k \le B_{jk}\left(1 - x_{jk}\right), \quad \forall k \in \mathcal{K} && (28)\\
& t_k + T_{kn} - t_n \le B_{kn}\left(1 - x_{kn}\right), \quad \forall k \in \mathcal{K}, n \in \mathcal{N} : (k,n) \in \mathcal{E}_j && (29)\\
& A_n \le t_n \le B_n, \quad \forall n \in \mathcal{N} && (30)\\
& x_{mn} \in \{0,1\}, \quad \forall (m,n) \in \mathcal{E}_j. && (31)
\end{aligned}
$$

The objective function (24) minimizes the cost of selected arcs. Constraint (25) initiates one path from satellite j. (26) is the flow conservation constraint. Constraint (27) ensures that the accumulated demand in a path does not exceed second echelon vehicle capacity. By constraints (28) and (29), vehicle arrival times are set with respect to the order of nodes visited in the path. Here, $B_{mn} = \max\left(B_m + T_{mn} - A_n, 0\right)$ is a sufficiently large constant. (28) and (29) also eliminate sub-tours in a solution. Constraint (30) limits the arrival times to the time windows and constraint (31) meet the binary requirement of the arc-flow variables.

If the optimal objective function value of an SP is negative, column λ_p is generated based on the selected arcs in the optimal solution of the subproblem. In §3.2, we provide the details on how the introduced subproblems are solved.

3.2. Solving a Subproblem

The subproblem (24)–(31) corresponds to an *elementary shortest path problem with resource constraints* (ESPPRC), which is NP-hard (Irnich and Desaulniers 2005). This problem is commonly solved using Labeling algorithm (Feillet et al. 2004, Irnich and Desaulniers 2005). More recently, Lozano, Duque, and Medaglia (2016) proposed an alternative approach, called Pulse algorithm (PA), to solve the ESPPRC arising in vehicle routing problems. The authors show that the algorithm is competitive with Labeling algorithm and can improve solution time when solving benchmark test instances.

PA is an enumeration-based algorithm that comprises two main stages: (i) a bounding scheme to narrow the solution space by finding a lower bound on the objective function value, and (ii) a recursive exploration procedure that finds the optimal solution based on an implicit enumeration of the solution space.

To solve an SP_j, PA initiates a partial path from the starting node $j \in \mathcal{J}$. An elementary forward path p is characterized by the following attributes: (i) the set of visited nodes $\mathcal{N}(p)$, (ii) the cumulative reduced cost of the path $\widetilde{C}(p)$, (iii) the total delivered load $D(p)$, (iv) the cumulative traveling time $T(p)$, and (v) the last and the second last visited nodes on path p, i.e. last(p) and pre(p), respectively. A partial path p for SP_j is initialized with $\mathcal{N}(p) = \{j\}$, $\widetilde{C}(p) = 0$, $D(p) = 0$, $T(p) = 0$, $\mathrm{last}(p) = j$, and $\mathrm{pre}(p) = \mathrm{N/A}$. PA recursively extends the current partial path by propagating throughout the outgoing arcs of last(p). We use different pruning strategies to prevent exploring the inferior search space. When a partial path p ending at node m is extended along an arc $(m,n) \in \mathcal{E}_j$, a new path p^{new} is formed with the following attributes:

$$
\begin{cases}
\mathcal{N}(p^{\mathrm{new}}) & = \mathcal{N}(p) \cup \{n\} \\
\widetilde{C}(p^{\mathrm{new}}) & = \widetilde{C}(p) + \widetilde{C}_{jmn} \\
D(p^{\mathrm{new}}) & = D(p) + D_n \\
T(p^{\mathrm{new}}) & = T(p) + T_{mn} \\
\mathrm{last}(p^{\mathrm{new}}) & = n \\
\mathrm{pre}(p^{\mathrm{new}}) & = \mathrm{last}(p) = m.
\end{cases}
$$

In order to satisfy capacity and time window constraints, we discard a partial path p^{new} if the extension leads to any of the following situations: $D(p^{\mathrm{new}}) > Q''$, $T(p^{\mathrm{new}}) > B_n$, or $T(p^{\mathrm{new}}) + T_{nj} > B_j$. Therefore, once a partial path reaches satellite j, its feasibility is ensured. The algorithm also forbids cost-dominated extensions based on the triangle inequality. Therefore, path p^{new} is also discarded if $\widetilde{C}_{j,\mathrm{pre}(p),m} + \widetilde{C}_{jmn} > \widetilde{C}_{j,\mathrm{pre}(p),n}$.

A key procedure in PA is lower-bounding. It is used to prune the search space by forbidding extension of unpromising paths based on their reduced cost and time consumption. In the pre-processing step of PA, we calculate a lover bound $\overline{C}(n,\overline{T})$ for each node $n \in \mathcal{N}$ and any value $\overline{T} \in \{B_j - \Phi_1, B_j - 2\Phi_1, \cdots, 0\}$, for a given time step Φ_1. $\overline{C}(n,\overline{T})$ denotes the minimum reduced cost that can be achieved by any partial path p that reaches node n with $T(p) \geq \overline{T}$. Therefore, the algorithm checks whether a possible extension on a path p^{new} can improve an upper-bound $\widetilde{C}^*$:

$$
\widetilde{C}(p^{\mathrm{new}}) + \overline{C}\left(n, T(p^{\mathrm{new}})\right) < \widetilde{C}^*. \tag{32}
$$

Here, we initially set $\widetilde{C}^*$ to 0 and update it with $\min\left(\widetilde{C}^*, \widetilde{C}(p)\right)$ as soon as any path p is completed (i.e. reaches the satellite node). If (32) does not hold, partial path p^{new} is discarded. When PA terminates, it returns a path with the most negative reduced cost, if such a path exists.

3.3. Column Generation Enhancements

Since PA enumerates over all outgoing arcs of the current node in a path, it can be time-consuming in initial stages of BP. Hence, heuristic approaches are commonly used in the literature of vehicle routing problems to find path columns more efficiently (see, for example, Contardo, Desaulniers, and Lessard 2015, Lozano, Duque, and Medaglia 2016, Farham, Süral, and Iyigun 2018).

We use two techniques to reduce the search space of PA and improve its run-time. The first approach modifies the underlying graph. The set $\mathcal{E}_j$ used to solve SP_j is reduced to contain only a fixed number of outgoing arcs (denoted by Φ_2) with smallest reduced costs from each node. This method is also used in Desaulniers, Lessard, and Hadjar (2008), Farham, Süral, and Iyigun (2018) to improve overall BP performance for a routing problem.

In the second approach, we reduce the search space of PA in a more greedy fashion to explore paths with larger negative reduced costs. To this end, we replace (32) with the following condition.

$$\widetilde{C}(p^{\text{new}}) + \overline{C}\left(n, T(p^{\text{new}})\right) < \Phi_3 \times \widetilde{C}^*, \tag{33}$$

where the right-hand-side of (32) is scaled using a parameter $\Phi_3 > 1$. In this way, the paths with no significant effect on the best bound are discarded.

PA in Lozano, Duque, and Medaglia (2016) only returns the best route it finds, i.e. the one with the most negative reduced cost. However, it is possible to keep track of all paths that update $\widetilde{C}^*$ and use them as new columns. In this study, we allocate a memory to store such paths and return them when PA terminates. Having more routes provides more information about the solution space and it can improve the convergence of BP.

Algorithm 1 presents the steps of the proposed CG procedure. We only run the exact PA if the heuristic approaches fail to find a promising column.

Algorithm 1: Column generation procedure

Step 1. Construct the reduced graph and run PA. If columns with negative reduced costs are found, **Stop**. Otherwise, go to Step 2.

Step 2. Construct the full graph and run PA with bound scaling in (33). If columns with negative reduced costs are found, **Stop**. Otherwise, go to Step 3.

Step 3. Run PA with original bounding in (32). **Stop**.

3.4. Column Generation Stabilization

Although column generation is very effective in solving hard combinatorial problems, it has its own drawbacks (Irnich and Desaulniers 2005) such as: (i) slow convergence or *tailing-off* effect, (ii) producing poor columns in early iterations due to lack of dual information, (iii) degeneracy in the primal resulting in multiple optimal dual solutions, and (iv) instability in the dual solutions that oscillate from one value to another. Non-smooth convergence of dual prices has been regarded as a major efficiency issue that has attained many attentions in the literature (Lübbecke and Desrosiers 2005).

In this study, we use a dual variable smoothing technique inspired by the work of Neame (1999) and Pessoa et al. (2013). The arc reduced costs in iteration $\tau \geq 2$ of BP, indicated as $\widetilde{C}^{\tau}_{jmn}$ used for SP_j is corrected based on the best reduced cost $\widetilde{C}^{*}_{jmn}$ found so far:

$$\widetilde{C}^{\tau}_{jmn} \leftarrow \Phi_4 \widetilde{C}^{*}_{jmn} + (1-\Phi_4)\widetilde{C}^{\tau}_{jmn} \quad \forall (m,n) \in \mathcal{J}, \tag{34}$$

where $0 \leq \Phi_4 < 1$ parameterizes the level of smoothing. In other words, the pricing problem is solved using the arc reduced cost obtained by taking a step size of $(1-\Phi_4)$ from the current dual prices towards the best dual prices found so far. It is possible that the pricing problem fails to find a solution over the smoothed dual prices, while there exists a solution when real dual values are used. This is a sequence of *mis-pricing*. In this case, the Φ_4 value is reduced iteratively and the pricing problem is resolved until a solution is found or Φ_4 converges to 0 (Pessoa et al. 2013). However, since our pricing problem is a difficult problem to solve, we set $\Phi_4 = 0$ after a mis-pricing in order to solve the pricing problem at most twice in one iteration.

3.5. Branching

BP applies a branch-and-bound method to ensure the solution to the master problem is integral. In the branching step, we prioritize more strategic variables over the other ones. The variables are selected for branching in the following order: (i) fractional z_i variables, $\forall i \in \mathcal{I}$, (ii) fractional z_j variables, $\forall j \in \mathcal{J}$, (iii) fractional y_{ij} variables, $\forall i \in \mathcal{I}, j \in \mathcal{J}$, (iv) fractional v_j variables, $\forall j \in \mathcal{J}$, and (v) fractional (m,n) arcs, $\forall (m,n) \in \mathcal{E}$.

For any satellite j, if $z_j = 0$ or $v_j = 0$ holds in a branch, we simply ignore solving SP_j in that branch. We perform a binary branching on arc $(m,n) \in \mathcal{E}$ that has the closest value to 0.5. In the first branch, we remove arc (m,n) from $\mathcal{E}_j$ for any subproblem j under that branch. In the other branch, we remove all outgoing arcs from node m except arc (m,n) as well as all incoming arcs to node n except arc (m,n) in order to force arc (m,n) to appear in the solution of the corresponding subproblems. Note that branching on the arc-flow variables guarantees integer solution.

4. Heuristic Algorithms

In §2, we identified strategic and tactical level decisions of the 2E-LRPTW. The top-level (strategic) decisions constitute CDC and satellite locations whereas the bottom-level (tactical) decisions involve first echelon allocations and second echelon vehicle routes and schedules. The proposed exact solution approach in this study deals with all decisions simultaneously. However, this can be computationally expensive when solving large-size 2E-LRPTW instances. In this section, we proposed two heuristics based on the hierarchical decomposition of the problem's decisions. The idea is to fix the decisions at one level and solve the (reduced) problem to determine the decisions of the other level.

4.1. Top-to-Bottom Approach

The first heuristic is inspired by the two-stage heuristic in Farham, Süral, and Iyigun (2018). It consists of two main stages. In the first stage, we reduce the problem by fixing CDC and satellite location decisions. Base on these decisions, first echelon allocations and second echelon vehicle routes are determined in the second stage. This method is called *top-to-bottom* approach, denoted by T$\triangleright$B. It starts from the strategic resolution and makes tactical decisions later.

T$\triangleright$B starts by solving a 2E-FLPTW (given in (1)–(12)) to find the location decisions by ignoring any routing decision. In the next stage, it constructs the 2E-LRPTW (see (14)–(22)) by fixing all CDC and satellite location variables to their optimal value obtained in the first stage. Next, the reduced 2E-LRPTW is solved by the proposed BP to determine the remaining variables. The main stages of T$\triangleright$B is given in Figure 4.

T$\triangleright$B is expected to run faster than BP since no branching is required on the location variables. However, solving large problems by T$\triangleright$B can still be time-consuming as the complicated decisions (i.e. vehicle routes) are determined by the exact CG in the algorithm.

4.2. Bottom-to-Top Approach

The 2E-LRPTW can be solved efficiently with off-the-shelf solvers when route sets $\mathcal{P}_j, \forall j \in \mathcal{J}$, are not very large. Branch-and-price-based approaches for the routing problems in the literature start by a small set of routes, commonly containing trivial facility-customer-facility routes, and generate new routes (columns) iteratively until no better route can be found. Different from BP, we may obtain a solution by generating a "good" set of routes first, and then solve the original problem once to find the optimal solution over the generated routes. Therefore, we introduce a two-stage approach, called *bottom-to-top* approach, that starts with the tactical level decisions (i.e. second echelon vehicle routes) and next, determines the strategic decisions (i.e. facility locations). Bottom-to-top heuristic, indicated by B$\triangleright$T, is outlined in Figure 5.

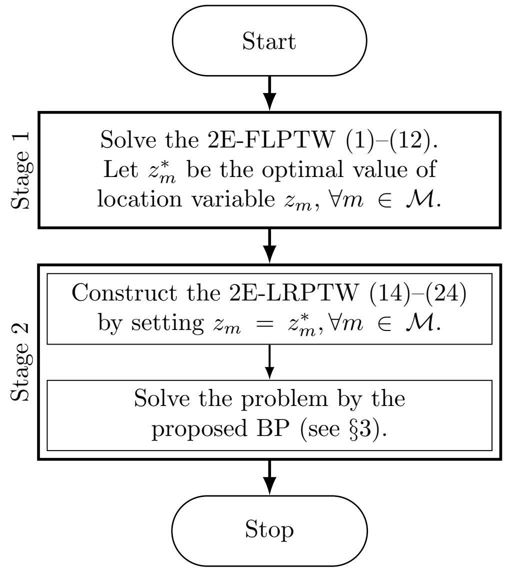


Figure 4 **Steps of the top-to-bottom algorithm.**

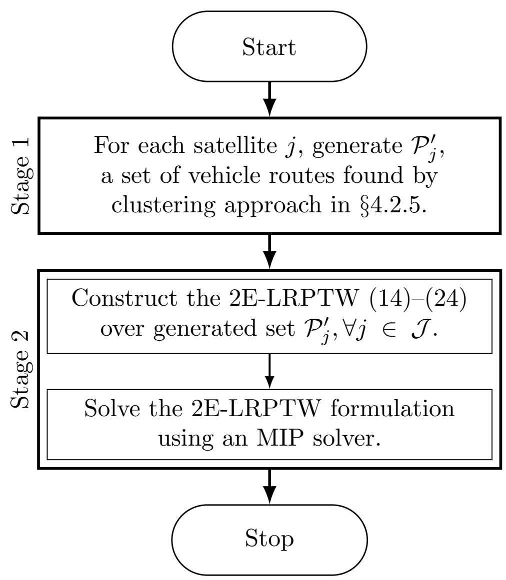


Figure 5 **Steps of the bottom-to-top algorithm.**

The solution quality of B▷T highly depends on the quality of the routes generated in its first stage. The similar approach is used by Ryan, Hjorring, and Glover (1993) to solve the capacitated VRP. The authors used a construction-based heuristic to find vehicle routes and solve a set-partitioning formulation for optimal selection of the generated routes. In this study, we generate second echelon routes for each satellite j by proposing a novel clustering technique that takes both capacity and time window constraints into account. In the following sections, we explain the clustering technique applied in the first stage of B▷T.

4.2.1. Cluster Analysis of the Second Echelon Nodes. The aim of clustering is to divide a given set of data points into a number of groups such that the points in the same group are more similar to each other than to those in other groups. In other words, clustering segregates data points with similar attributes and assigns them into clusters based on a distance measure. Clustering-based heuristics used to solve vehicle routing problems can be categorized as follows (Laporte and Semet 2002).

- *Cluster-first, route-second*: This method is used in three distinct ways. In the first approach, the original problem is decomposed into smaller problems, each dealing with a subset (a cluster) of customers. Next, a VRP is solved for each cluster as a part of the whole problem. The second approach clusters customers into a given number of groups equal to the number of available vehicles. Next, either a traveling salesman problem is solved or a construction heuristic is used to find a vehicle route within each cluster. The last approach aggregates customers into small clusters to make smaller number of nodes, called macro nodes, and finds routes to visit the macro nodes. Then, each macro node is disaggregated and vehicle routes are modified accordingly.
- *Route-first, cluster-second*: This method starts with a giant vehicle route, disregarding the side constraints. Then, this route is iteratively discomposed into smaller routes until all constraints are satisfied.

In the vehicle routing problems, customers can be viewed as data points. They have different attributes such as their location, time window, and demand. There is only a limited number of studies in the literature that cluster customers under time window restrictions. Dondo and Cerdá (2007) and Pugacs (2014) propose clustering approaches to aggregate customers into macro nodes and then find vehicle routes to visit customers in each node. Qi et al. (2012) uses a different cluster-first, route-second approach where customers are clustered using spatiotemporal distances. Then, a VRP with *soft time windows* is solved to find vehicle routes for each cluster. Spatiotemporal distances between two customers consider two factors: the spatial distance (i.e. the Euclidean distance) and the temporal distance based on their time windows. The temporal distance between customer k and customer l is a function of the time at which l is reached from k. This distance increases if the arrival time to l falls outside its time windows. Therefore, in the problems with soft time windows, an additional cost is added to the objective function based on the amount of time window violation. Unlike soft time windows, *hard time windows* affect route feasibility.

As the time windows in the 2E-LRPTW are hard and the capacity of vehicles cannot be violated, we propose a *constrained clustering* technique. A cluster represents a set that accepts a *feasible* vehicle route starting from a satellite j, visiting a set of customers $\mathcal{K}' \subseteq \mathcal{K}$, and returning to satellite j such that vehicle capacity constraint and all costumer time windows are satisfied. The following terminologies are used in our clustering approach.

The Distance Measure. The distance (or dissimilarity) between any two nodes $m, n \in \mathcal{N}$ is calculated as:

$$dist(m,n) = \begin{cases} \|(n,m)\|, & \text{if arcs } (m,n) \text{ and } (n,m) \text{ are both feasible,} \\ \Phi_5 \times \|(n,m)\|, & \text{otherwise,} \end{cases} \tag{35}$$

where, $\|(n,m)\|$ is the Euclidean norm of arc $(n,m) \in \mathcal{E}$ and Φ_5 is a given parameter to penalize the distance between two nodes that are unreachable from each other. An arc (m,n) is called *feasible* if it satisfies all the following *arc feasibility conditions* (AFCs):

$$\text{AFCs:} \begin{cases} \text{(i)} & D_m + D_n \leq Q'' \\ \text{(ii)} & T_{mn} \leq B_n \\ \text{(iii)} & \text{if } m, n \in \mathcal{K}, \text{ then } \exists j \in \mathcal{J}: \text{route } j\text{–}m\text{–}n\text{–}j \text{ satisfies RFCs (13).} \end{cases} \tag{36}$$

Condition (i) ensures that the demand of nodes m and n can be delivered by one vehicle. Conditions (ii) checks whether a vehicle can reach node n from node m before the closing time of node n. Finally, for any two customers m and n, condition (iii) ensures that there exists at least one feasible route that traverses arc (m,n).

Route Construction and Validity of Clusters. A cluster containing satellite j and customer set $\mathcal{K}'$ is denoted by `Cluster`$(j,\mathcal{K}')$. `Cluster`$(j,\mathcal{K}')$ is called a *valid* cluster if there exists a vehicle route that starts from j and visits all nodes in $\mathcal{K}'$ by satisfying RFCs (13). In order to construct such a route, we use *sequential insertion heuristics*, called I1, proposed by Solomon (1987). I1 is shown to produce good results for the VRPTW (Bräysy and Gendreau 2005). Given a depot location, I1 initializes a route with a seed customer and the remaining unvisited customers are added into this route while it yields a feasible route. If any customer remains unvisited, the initialization and insertion procedures are repeated until all customers are served. The quality of routes found by I1 depends on its seeds. The seed customers are commonly selected by finding either the geographically farthest unvisited customer to the depot or the one with the earliest closing time B_k. Given a satellite node j and customer set $\mathcal{K}'$, we construct the routes of `Cluster`$(j,\mathcal{K}')$ by applying I1 over $\mathcal{K}'$ considering satellite j as the depot. Since $\mathcal{K}'$ usually contains a small subset of customers, I1 heuristic can be executed efficiently. Hence, we repeat the insertion heuristic for each of the customers as the seed and select the best route found by all seeds. At the end, if a customer in set $\mathcal{K}'$ remains unvisited, we conclude that no feasible route can be found under the given settings and `Cluster`$(j,\mathcal{K}')$ is called as *invalid*.

The proposed constrained clustering is different from the ones in the literature that are mentioned earlier. Here, the clustering and route construction phases are done simultaneously to generate feasible vehicle routes. In the literature, however, the clustering and the routing phases are done separately in a sequential manner. We propose three different clustering methods to form clusters, namely agglomerative route clustering (ARC), divisive route clustering (DRC), and greedy route clustering (GRC). First two of the proposed approaches generate the clusters recursively in an hierarchical order. The routes obtained by the clustering approaches are provided as $\mathcal{P}'_j \subset \mathcal{P}_j$ sets, $\forall j \in \mathcal{J}$, that represent λ_p columns in the 2E-LRPTW (14)–(22).

4.2.2. Agglomerative Route Clustering. ARC treats each node as a singleton cluster initially, and then successively merges (or agglomerates) pairs of clusters until all clusters have been merged into a single cluster or a stopping criterion is met. The ARC in our study starts with a given set of clusters and applies its *merging* procedure until no further merging is possible. For a given satellite location, two clusters S_1 and S_2 are merged if a feasible vehicle route starting at j and visiting all customers in S_1 and S_2 can be constructed. In this way, the termination point of ARC is naturally determined by the algorithm and there is no need for an external stopping criterion. The pseudo-code of the proposed ARC is given in Algorithm 2.

ARC algorithm may start with any given initial set of clusters denoted by Σ^0. If the initial set is not provided, it can be formed by generating *singleton* clusters. A singleton cluster contains a satellite node j and a customer k such that the trivial route j–k–j is feasible.

In order to merge two clusters S_1 and S_2 in ARC, we use *single-linkage* distance measure defined as:

$$dist(S_1, S_2) = \min_{m \in S_1, n \in S_2 : m,n \in \mathcal{K}} dist(m, n). \tag{37}$$

The algorithm keeps pair-wise distances in a two-dimensional matrix $\mathbf{\Delta}$ and update the corresponding elements of the matrix whenever two clusters are merged. If merging the two clusters results an *invalid* cluster, the distance between them is set to ∞ in order to prevent them from merging in the future. Figure 6 illustrates three different steps of the ARC. In the first step, initial singleton clusters are provided. In the next step, two closest clusters are merged to form a new (valid) cluster. The final step returns the clusters that cannot be merged anymore (see Figure 6c). Figure 6d illustrates the vehicle routes provided by the final clusters.

4.2.3. Divisive Route Clustering. In contrast to ARC, where smaller clusters are merged into larger clusters, DRC is based on the idea of splitting larger clusters into smaller ones. Therefore, DRC starts with a large cluster containing all data points. Then the cluster is split recursively until a stopping criterion is met.

The algorithm runs over a given satellite j and a given set of customers $\mathcal{K}^0 \subseteq \mathcal{K}$. $\mathcal{K}^0$ is initially equal to the set of all customers $k \in \mathcal{K}$ that can form feasible j–k–j routes. At the beginning, DRC creates a cluster to cover all given customers. If such a cluster is *valid*, it is returned and the algorithm terminates. Otherwise, it splits the current set of customers by finding the customer that has the largest average distance to the other customers. Next, this customer, say k^{far}, is removed from $\mathcal{K}^0$, to form a new set $\mathcal{K}^{\text{new}} = \{k^{\text{far}}\}$. Then, $\mathcal{K}^0$ and $\mathcal{K}^{\text{new}}$ sets are *balanced* by moving

Algorithm 2: Agglomerative route clustering procedure

Procedure `ARC`(j, Σ^0)

input : A satellite j, a set of initial clusters Σ^0

output: A set of clusters with routed customers

construct the distance matrix $\mathbf{\Delta}$ where $[\mathbf{\Delta}]_{rr'} = dist(S_r, S_{r'})$, $\forall S_r, S_{r'} \in \Sigma^0, r \neq r'$, using equation (37)

`// merging procedure`

while *minimum of* $\mathbf{\Delta} < \infty$ **do**

  let S_r and $S_{r'}$ be the two closest clusters

  let $\mathcal{K}^{\text{new}}$ be the set of all customers in S_r and $S_{r'}$

  let $S^{\text{new}} \leftarrow$ `Cluster`(j, $\mathcal{K}^{new}$)

  if S^{new} *is valid* **then**

    $S_r \leftarrow S^{\text{new}}$

    remove row/column of $\mathbf{\Delta}$ corresponding to $S_{r'}$

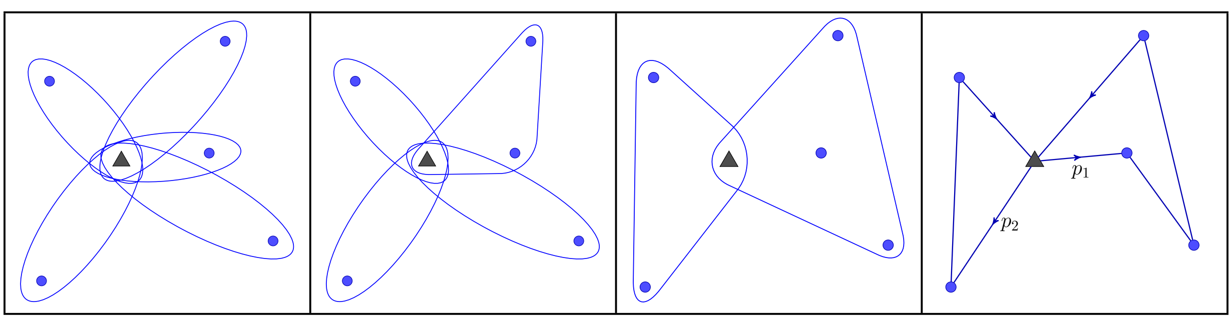
    update row/column of $\mathbf{\Delta}$ corresponding to S_r

  else $[\mathbf{\Delta}]_{rr'} \leftarrow \infty$

end

let Σ be the set of clusters corresponding to the remaining rows (or columns) of $\mathbf{\Delta}$

end



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

Figure 6 **Different steps of ARC.**

Note. ▲ shows the satellite node and • is a customer point. (a) Initial singleton clusters. (b) Merging two closest clusters. (c) Final clusters. (d) Representation of the final clusters into two routes p_1 and p_2.

customers from the larger set to the smaller one. A customer is moved from $\mathcal{K}^0$ to $\mathcal{K}^{\text{new}}$ if its average distance to the customers in $\mathcal{K}^{\text{new}}$ is smaller than its average distance to the customers in $\mathcal{K}^0$. At the end, DRC is recursively applied on both sets $\mathcal{K}^0$ and $\mathcal{K}^{\text{new}}$. When the current set of customers forms a valid cluster, the recursive procedure terminates. Therefore, validity of the generated clusters determines termination point for DRC. The steps of the proposed DRC are presented in Algorithm 3.

Algorithm 3: Divisive route clustering procedure

Procedure `DRC(`j, $\mathcal{K}^0$`)`

  input : A satellites j, a set of customers $\mathcal{K}^0$

  output: A set of clusters with routed customers Σ

  let $S \leftarrow$ `Cluster(`j, $\mathcal{K}^0$`)`

  if *S is valid* **then**

    let $\Sigma \leftarrow \{S\}$

    end procedure

  else

    `// split the set`

    let $k^{\text{far}} \leftarrow$ the customer in $\mathcal{K}^0$ with the largest average distance to the other customers with respect to the distance function (35)

    remove k^{far} from $\mathcal{K}^0$

    let $\mathcal{K}^{\text{new}} \leftarrow \{k^{\text{far}}\}$

    `// balance the two sets`

    foreach *customer $k \in \mathcal{K}^0$* **do**

      if *customer k has smaller average distance to the customers in $\mathcal{K}^{new}$ than to the other customers in $\mathcal{K}^0$* **then** move k from $\mathcal{K}^0$ to $\mathcal{K}^{\text{new}}$

    end

    let $\Sigma_1 \leftarrow$ `DRC(`j, $\mathcal{K}^0$`)`

    let $\Sigma_2 \leftarrow$ `DRC(`j, $\mathcal{K}^{new}$`)`

    let $\Sigma \leftarrow \Sigma_1 \cup \Sigma_2$

  end

end

4.2.4. Greedy Route Clustering. In addition to ARC and DRC introduced above, we also propose a simple greedy clustering method to find customer clusters. Given a satellite node j and a set of customers $\mathcal{K}^0 \subseteq \mathcal{K}$, the GRC starts a cluster containing only satellite j. Then, the algorithm repeatedly adds the closest customer to the current cluster as long as the resulting cluster is *valid*. If the next candidate customer cannot be added to the current cluster, the current cluster is closed, and a new cluster containing satellite j is initialized. Then, the algorithm tries to add the remaining customers to the new cluster. This procedure is repeated until all customers are clustered. The distance between an unassigned customer k and a cluster S is calculated as:

$$dist(k,S) = \min_{n \in S} dist(k,n)\,. \tag{38}$$

The proposed GRC is outlined in Algorithm 4.

Algorithm 4: Greedy route clustering procedure

Procedure `GRC`(j, $\mathcal{K}^0$)
- **input :** A satellite j, a set of customers $\mathcal{K}^0$
- **output:** A set of clusters with routed customers Σ
- let $\mathcal{K}^{\text{unassigned}} \leftarrow \mathcal{K}^0$
- let $\Sigma \leftarrow \emptyset$
- **repeat**
  - let $\mathcal{K}^{\text{current}} \leftarrow \emptyset$
  - let $S \leftarrow \texttt{Cluster}(j, \mathcal{K}^{\text{current}})$
  - **loop**
    - let $k \leftarrow$ closest customer in $\mathcal{K}^{\text{unassigned}}$ to S according to equation (38)
    - let $S^{\text{new}} \leftarrow \texttt{Cluster}(j, \mathcal{K}^{\text{current}} \cup \{k\})$
    - **if** S^{new} *is valid* **then**
      - move k from $\mathcal{K}^{\text{unassigned}}$ to $\mathcal{K}^{\text{current}}$
      - $S \leftarrow S^{\text{new}}$
    - **else**
      - add S to set Σ
      - **break** loop
    - **end**
  - **end**
- **until** $\mathcal{K}^{unassigned}$ *is empty*

end

4.2.5. The Main Clustering Procedure. Algorithm 5 presents the main clustering procedure used to find candidate second echelon vehicle routes in the first stage of B⊳T (see Figure 5). Given a satellite node j and the set of customer nodes $\mathcal{K}$, the algorithm generates a set of second echelon vehicle routes originating at satellite j and visiting customers in $\mathcal{K}$. First, we find the valid singleton clusters for satellite j. If a customer k cannot form a feasible j–k–j route then it cannot be part of any other route for j. Hence, we exclude k from being processed in the clustering stage. The clustering stage takes the advantage of all the clustering methods presented above. It consists of the following steps: (i) run the ARC over j and the set of initial (i.e. singleton) clusters (line 9),

(ii) run the DRC over j and the set of valid customers (line 10), (iii) run the GRC over j and the set of valid customers (line 11), and (iv) run ARC over j and the clusters obtained from DRC (line 12). In line 12, ARC is rerun with a different initial cluster set, i.e. clusters provided by DRC. This enables us to merge clusters in order to find new ones which may not have been generated by other methods.

Algorithm 5: Main clustering procedure

Procedure `Clustering`(j, $\mathcal{K}$)

  input : A satellite j, set of customers $\mathcal{K}$

  output: $\mathcal{P}'_j$, a set of vehicle routes for satellite j

  `// create the valid customer set and initial clusters`

1 let $\mathcal{K}^0 \leftarrow \emptyset$ and $\Sigma^0 \leftarrow \emptyset$

2 **foreach** *customer* k *in* $\mathcal{K}$ **do**

3   let $S \leftarrow$ `Cluster`$(j, \{k\})$

4   **if** S *is valid* **then**

5   add S to Σ^0

6   add k to $\mathcal{K}^0$

7   **end**

8 **end**

9 let $\Sigma_1 \leftarrow$ `ARC`(j, Σ^0)

10 let $\Sigma_2 \leftarrow$ `DRC`(j, $\mathcal{K}^0$)

11 let $\Sigma_3 \leftarrow$ `GRC`(j, $\mathcal{K}^0$)

12 let $\Sigma_4 \leftarrow$ `ARC`(j, Σ_2)

13 **return** $\mathcal{P}'_j$ as the set of all the vehicle routes represented by the cluster set $\Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \Sigma_4$

end

4.2.6. Linking clustering with the master problem. The aim of clustering in B⊳T is not to optimize a clustering objective, but to produce a reliable set of second-echelon vehicle routes for each satellite. In this section, we introduce two methods that potentially improve the quality of B⊳T solutions. In the first method, we explain how $\mathcal{P}_j$ sets generated during the clustering procedure can be extended to include more routes. Larger $\mathcal{P}_j$ sets provide more alternatives for the MIP solver, which enables it to obtain a better composition of the second-echelon routes. In our approach, it is possible to keep not only the final clusters in ARC and GRC, but also the history of all valid clusters already generated through the iterations. In ARC, set Σ can be extended as soon as a new valid cluster is formed. Therefore, one can add a copy of S_r to Σ after it is updated in line 7 of Algorithm 2. In GRC, we can add the current cluster to set Σ once it is expanded to cover a new customer. Therefore, one can add a copy of S to Σ after it is updated in line 11 of Algorithm 4.

Since of the solution obtained in the second stage of B⊳T depends on the quality of the routes generated in the first stage, we can also benefit from a post-processing procedure as the second clustering enhancement method. The final routes in $\mathcal{P}_j$ sets can be improved by using two conventional techniques from the literature: the *intra*-route improvement operators, such as *2-Opt* and *Or-Opt*, on a given route, and the *inter*-route improvement operators such as *2-Opt**, *relocate*, and *exchange* operators on a selected pair of routes. For more information about the route improvements methods, see Bräysy and Gendreau (2005).

The clustering method presented in Algorithm 5 is implemented for each satellite j independent of other satellites. Therefore, in order to boost the run-time of B⊳T, Algorithm 5 can be executed in parallel with respect to j.

5. Computational Experiments

In this section, we implement the proposed exact and heuristic approaches on a set of 2E-LRPTW test instances and present extensive computational results. This section enables us to assess the proposed exact and heuristic algorithms on solving 2E-LRPTWs and analyze the effect of instance characteristics (such as problem size and facility/customer distributions) on the performance of the algorithms. Since there are no benchmark instances for the 2E-LRPTW in the literature, we generate new sets of tests instances based on a set of well-known VRPTW instances in the literature (see §5.1). §5.2 explains a number of preprocessing steps, based on the problem instance characteristics, that can be used to enhance the solution procedure. Parameter adjustments are presented in §5.3. In §5.4, we analyze the effect of a good upper bound on the exact approach (i.e. BP) and T⊳B algorithm. Finally, the computational studies of the proposed exact and heuristic approaches are provided in §5.5 and §5.6, respectively.

5.1. Problem Instances

We use two sets of problem instances in this study. The first set, named Set 1, is based on the benchmark instances of Solomon (1987). The original instances are generated for the capacitated VRPTW with one depot location and a set of customers. We modified these instances to include candidate CDC and satellite points. The test instances in Set 1 are classified into three groups based on the distribution of customers on the plane: clustered (indicated by C), random (R), and a mix of random and clustered (RC). Solomon test instances are of two types: they have either tight time windows and low vehicle capacity or wide time windows and high vehicle capacity. In Set

1, we consider the former type instances with 2 candidate CDC nodes, 2 to 4 candidate satellite nodes, and 15, 20, 25, or 30 customer nodes.

The test instances of Set 2 are based on Dellaert et al. (2019) instances generated for the 2E-VRPTW. Although the original instances contain CDC and satellite facilities, they do not incorporate facility capacity and opening costs. Therefore, we modify these instances by assigning capacity and fixed costs to CDCs and satellites as the potential facility locations. The instances in Set 2 are categorized into four groups based on customer time windows and customer demands. Group-a and Group-b instances have tight time windows but Group-b has more diverse demand distribution. Group-c and Group-d instances have similar demand distribution to Group-a instances but have time windows with larger starting times. Group-c instances has wider time windows than Group-d. Each group contains test instances with 2, 3, or 6 candidate CDC nodes, and 3 to 5 candidate satellite nodes, and 15, 30, 50, or 100 customers. An instance size is indicated by three numbers ordered as $\#_1$-$\#_2$-$\#_3$ denoting the number of candidate CDC locations, the number of candidate satellite locations, and the number of customers, respectively. All instance data files are available in `http://tol.ie.metu.edu.tr`.

5.2. Preprocessing

Desrochers, Desrosiers, and Solomon (1992) suggest tightening customer time windows based on travel times. For each customer node $k \in \mathcal{K}$, the time window width is reduced using (39). The first two terms, adjust the beginning of customer k's time window A_k, by calculating the minimal arrival time from predecessors and minimal arrival time to successors, respectively. The last two terms of (39) fix the end of customer k's time window B_k, based on the maximal departure time from predecessors and maximal departure time to successors, respectively. Tighter time windows apply more restrictions on search space of PA.

$$\begin{cases} A_k \leftarrow \max\left(A_k, \min\left(B_k, \min_{m\in\mathcal{N}} A_m + T_{mk}\right)\right) \\ A_k \leftarrow \max\left(A_k, \min\left(B_k, \min_{n\in\mathcal{N}} A_n - T_{kn}\right)\right) \\ B_k \leftarrow \min\left(B_k, \max\left(A_k, \max_{m\in\mathcal{N}} B_m + T_{mk}\right)\right) \\ B_k \leftarrow \min\left(B_k, \max\left(A_k, \max_{n\in\mathcal{N}} B_n - T_{kn}\right)\right). \end{cases} \tag{39}$$

5.3. Implementation Details

The experiments are done on a Linux v4.15 machine with Intel® Xeon 4×3.20GHz CPUs and 16GB memory. All algorithms are coded in C++ compiled with GCC v7.3 using SCIP optimization suite framework v6.0 (Gleixner et al. 2018) linked to CPLEX v12.8 (IBM 2018) as the linear programming solver. We use single-thread computing in our experiments.

In order to determine the value of our parameters, we conducted preliminary experiments on a small set of instances and report the selected parameter values for the exact and heuristic approaches in Table 1.

Table 1 Parameter settings.

Parameter	Description	Value
Φ_1	Time step in the lower-bounding procedure of PA	$0.05B_j$
Φ_2	No. of outgoing arcs in the reduced graph	5
Φ_3	Bound scaling multiplier in pulse algorithm	2.0
Φ_4	Dual price smoothing coefficient	0.5
Φ_5	Distance penalty for unreachable nodes	1.75

In our experiments, we terminate T$\triangleright$B algorithm whenever the relative MIP gap reaches 0.5% or less. This allows us to approximate the solution faster without sacrificing much of its quality. B$\triangleright$T and exact approaches are allowed to run until this gap closes. All algorithms are run in a time limit of 4 hours.

In order to find $\overline{Z}$ and $\overline{V}$ values defined in §2.2.1, one can use $\overline{Z} = \lceil \max_{j\in\mathcal{J}} Q_j / \sum_{k\in\mathcal{K}} D_k \rceil$ and $\overline{V} = \lceil Q'' / \sum_{k\in\mathcal{K}} D_k \rceil$, where $\lceil\cdot\rceil$ is the ceiling function. However, these bounds can be improved by solving small problems during the preprocessing stage. To calculate $\overline{Z}$, a bin-packing problem is solved where item sizes are customer demands and each bin represents a satellite with the given capacity. $\overline{V}$ is found similarly, except bin capacities are all equal to the capacity of second-echelon vehicles (Q'') and there can be as many bins as the number of customers. Since the number of customers is not very large in 2E-LRPTW instances, these bin-packing problems can be solved very efficiently with today's MIP solvers.

Using the above settings, we provide detailed numerical results of solving the test instances by the proposed BP and heuristic algorithms in the remainder of this section.

5.4. The Upper-Bounding Effect

BP and T$\triangleright$B approaches can use the solution of the 2E-FLPTW as a starting point (an upper-bound) to search for the final solution. In BP, no decision is fixed in the beginning, and the algorithm is allowed to investigate decisions other than the ones provided by the solution of the 2E-FLPTW. In T$\triangleright$B, however, we use the 2E-FLPTW solution to fix the locations of CDCs and satellites. Once these decisions are fixed, they will never change in the later steps. Therefore, if the locations are decided poorly, high overall cost might be incurred as a facility location affects the cost of vehicle route originated at that location. Since the facility location decisions in the 2E-FLPTW are made according to customer-to-satellite assignments without considering routing decisions in the second echelon, they can lead to undesirable results when establishing vehicle routes in the next step. This potential drawback can be avoided if our perception of the network design solution is improved. Here, we take the advantage of information available in the solution of B$\triangleright$T to make better location decisions in the T$\triangleright$B approach. Since the location decisions in the B$\triangleright$T approach are made according to nontrivial approximated second echelon routes, they are more reliable compared to the ones made in the 2E-FLPTW approach and can provide more precise information about the final solution.

In order to see the initialization effect on the T▷B solutions, we initialize this algorithm in two different ways: (i) fixing the facility locations based on the 2E-FLPTW solution, and (ii) based on the B▷T solution. If the latter case is used, we call the algorithm T*▷B. Figures 7 and 8 illustrate the effect of initial solution in T▷B. In Figure 7, facility locations are fixed according to the 2E-FLPTW solution. The 2E-FLPTW solution suggests opening CDC 0 and satellites 3, 4, and 5. However, even though its opening cost is incurred, satellite 5 is not used in any second echelon routes of the final solution. In Figure 8, facility locations are fixed according to the B▷T solution. This time, CDC 1 is used and satellite 3 is kept closed. Such decisions resulted in a dominating solution with much less objective function value. Here, starting T▷B with wrong location decisions led to a high objective function value, even though the routing decisions are optimal. On the other hand, using the location decisions provided by the B▷T solution led us to the optimal solution for this instance. Section §5.6 provides a more detailed comparison of T▷B and T*▷B performances.

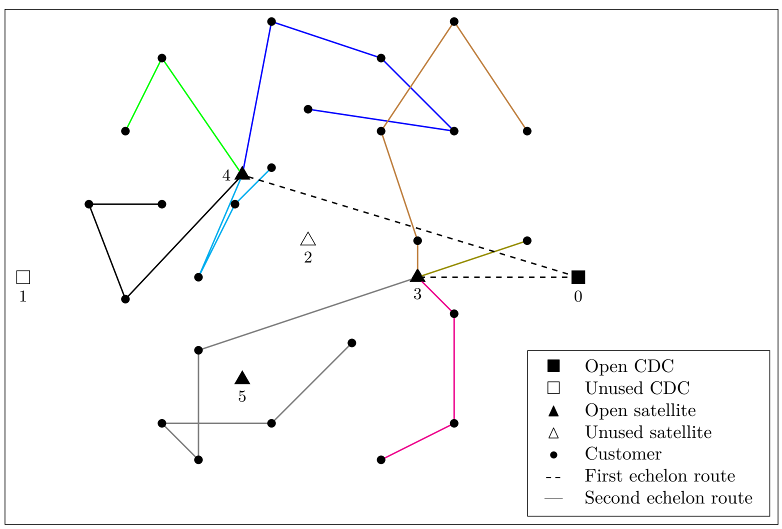


Figure 7 Illustration of the T▷B solution of the test instance R102 with size 2-4-25 (objective function value: 1649.2).

Note. In order to demonstrate direction of routes in the second echelon, the last customer-to-satellite arc of the second echelon routes are not depicted.

We can also use the B▷T as an initialization step for BP. Therefore, before solving any problem instance by BP, we first solve the problem by B▷T and then use the objective function value of the B▷T solution as an upper-bound and its final second-echelon vehicle routes as initial columns in BP (see the first step in Figure 3). Numerical results presented in §5.6 show that B▷T can provide efficient and high quality vehicle routes. Figure 9 shows the contribution of finding initial

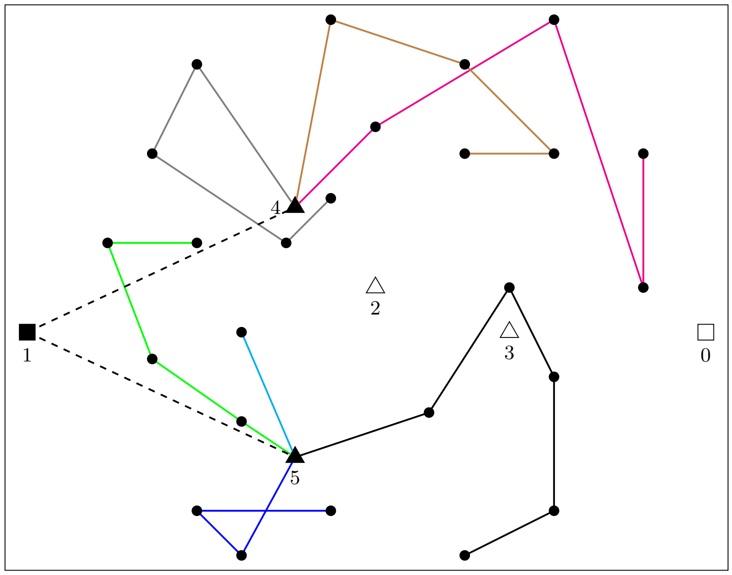


Figure 8 Illustration of the T$^*\triangleright$B solution of the test instance R102 with size 2-4-25 (objective function value: 1494.8). This solution is identical to the BP solution this problem.

columns in the exact approach in total computational time for a particular test instance. Here, finding initial columns corresponds to solving the problem with B$\triangleright$T and providing its solution as the starting point for the BP. As it is shown in Figure 9, this step is done very fast compared to the total computational time of BP.

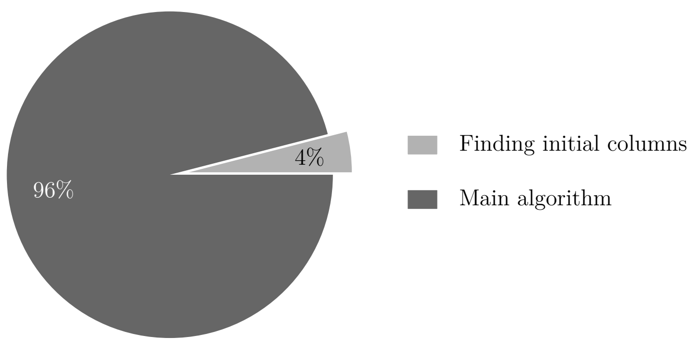


Figure 9 Amount of time spent in different stages of the exact approach for the test instance C102 with size 2-3-20 (Total time: 507.1s).

5.5. Numerical Results of the Exact Approach

This section presents the extensive computational results of implementing BP to solve our 2E-LRPTW problem test instances. We group the instances based on their type and size, and consider the following measures in evaluating the proposed BP for each group. *#opt* shows the number of optimal solutions found. *Av obj val* indicates the average objective function value. *Av subopt gap* is the average percentage MIP gap reported for the instances that are not solved to optimality during the given computing time (4 hours). *Av #pricer calls* denotes the average number of times the algorithm called the pricing problems. *Av #BB nodes* shows average number of nodes processed in the branch-and-bound tree. *Av #CDC-sat* denotes the average number of open CDCs and satellites in the final solution. *Av #veh1-veh2* is the average number of vehicles used in the fist and second echelon. *Av time* is the average solution time in seconds.

Table 2 shows the computational details of solving Set 1 instances by BP. The instances are grouped according to the customer distribution type (i.e. C, R, and RC). There are more than one instances having the same size in each group that are different in terms of customer time windows and/or demands. *#Inst* shows the number of instances in each size group. The summary of each type group is provided in the last row of the group.

In total, BP finds the optimal solution of 281 out of 348 test instances in Set 1. The average MIP gap for unsolved instances is 3.22%. When the number of candidate satellite locations and customers increase, the problem gets more difficult to solve. More branching nodes are explored and, consequently, the pricing problems are called more frequently. This leads to higher computational times. Among the three instance type groups, R is the most challenging one to solve. When customers are distributed randomly, more BB nodes are processed and more time is spent to solve the pricing problems. The reason is that the number of alternative second echelon routes increases for randomly distributed customers. Therefore, ensuring the optimal solution requires enumerating more BB nodes and vehicle route options. This is also shown by #veh2 value. The average #veh2 increases when moving from C instance group to RC, and from RC to R. On average, BP consumes 1 hour to solve an instance in Set 1. The optimality hit rate for this algorithm is around 80% for each type group. Customer distribution does not affect the performance of BP in searching for the optimal solution.

Numerical results for Set 2 instances are provided in Table 3. BP finds the exact solution of 195 instances out of 240 in Set 2. Considering the number of customers served, the exact algorithm is able to find the optimal solution for all instances with 15 and 30 customers in Set 2. Only two (out of 60) instances with 50 customers yield nonzero MIP gaps (around 0.35%). Among 60 instances with 100 customers, 17 are optimally solved by BP. For the instances that the optimal solution is not guaranteed during the time limit, the average MIP gap is 4.81%. The amount of optimaly solved instances for Group-a and Group-d is 88% and 86%, respectively, that is more than the other groups. Solution times are also smaller for Group-a and Group-d test instances. This shows that the problem is easier to solve when the customers have tight time windows and the demand distribution is more uniform. The most difficult group of Set 2 instances is Group-c, where wide customer time windows are considered. When time windows are wide, search space of PA increases and the algorithm needs more time to generate promising columns. BP is able to reach optimality for 73% of the instances in Group-*c*. The results in Table 3 indicate that while the overall instance size affects the performance of BP, the algorithm is more sensitive to the number of customers than to the number of candidate satellite locations, and more sensitive to the number of satellite locations than the number of candidate CDC locations. It is expected as most of the complications arises by the routing problem in the second echelon. On average, when the size of the second echelon network gets larger and/or customer time windows get tighter, more second echelon vehicles are used. On average, BP results in small MIP gap values and a computational time which is less than 1 hour.

Our experiments demonstrated the effect of the CG enhancements introduced in §3.3. We observed that the reduced graph and bound scaling steps of Algorithm 1 significantly contributed to CG by eliminating the need for running exact PA in many cases. Since the exact PA is more time-consuming than the other strategies, running it less frequently, can boost the performance of the algorithm. Figure 10 illustrates the contribution of different strategies in generating columns of the master problem for a particular test instance. The results indicate that implementing PA on reduced graph and on the complete graph with bound scaling can help to find a significant number of columns without a need to run exact PA on the complete graph.

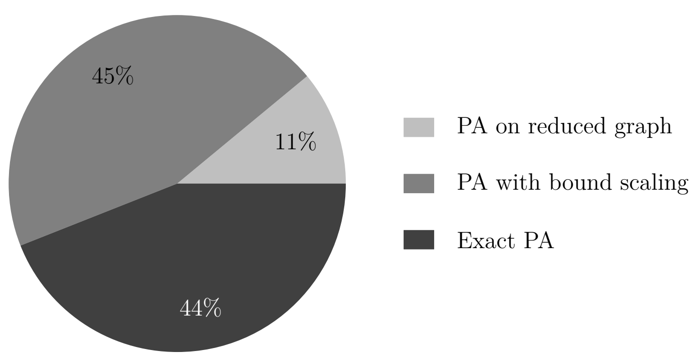


Figure 10 **Amount of columns found by different subproblem solvers for the test instance C102 with size 2-3-20 (Total columns: 5241).**

5.6. Numerical Results of the Heuristic Approaches

In this section, we run the proposed heuristic approaches to solve Set 1 and Set 2 problem instances and analyze the results. Table 4 provides numerical results of the three heuristics, namely T▷B, T*▷B, and B▷T, over Set 1 instances, and their comparisons to the exact (i.e. BP) approach. *#best* shows the total number of best solutions found by an algorithm considering the solution of BP or any of the heuristics. *Av %dev from best* indicates the average percentage deviation of the objective function value found by an algorithm from the best value reported by any of the algorithms.

Table 2 Numerical results of BP for Set 1 instances.

Inst type/size	#Inst (#opt)	Av obj val	Av subopt gap (%)	Av #pricer calls	Av #BB nodes	Av #CDC-sat	Av #veh1-veh2	Av time (s)
C 2-2-15	9 (9)	1452.63		365.7	3.8	1 - 2	2 - 2.0	29.2
C 2-2-20	9 (8)	1519.29	1.86	1230.4	53.3	1 - 2	2 - 2.6	3331.3
C 2-2-25	9 (7)	1577.08	4.7	1757.1	77.3	1 - 2	2 - 4.0	3815.0
C 2-2-30	9 (6)	1596.03	4.87	3920.8	150.2	1 - 2	2 - 4.0	6067.3
C 2-3-15	9 (9)	1397.21		567.2	4.4	1 - 2	2 - 2.0	101.6
C 2-3-20	9 (7)	1449.53	3.16	2007.8	132.3	1 - 2	2 - 2.4	3436.1
C 2-3-25	9 (6)	1450.92	3.49	3846.3	127.8	1 - 2	2 - 4.0	5231.7
C 2-3-30	9 (6)	1471.03	4.32	3919.7	98	1 - 2	2 - 4.0	6658.3
C 2-4-15	9 (9)	1396.08		728.8	17.4	1 - 2	2 - 2.0	197.3
C 2-4-20	9 (7)	1449.61	4.49	4585.3	366.2	1 - 2	2 - 2.7	4501.9
C 2-4-25	9 (7)	1451.33	5.39	3344.9	94.8	1 - 2	2 - 4.0	4892.5
C 2-4-30	9 (5)	1470.89	4.39	4170.6	65	1 - 2	2 - 4.0	7376.1
C	108 (86)	1473.47	4.22	2537.1	99.2	1 - 2	2 - 3.1	3803.2
R 2-2-15	12 (12)	1295.98		810.8	113.4	1 - 2	2 - 4.3	19.1
R 2-2-20	12 (12)	1362.17		7577.7	1502.8	1 - 2	2 - 4.8	859.6
R 2-2-25	12 (7)	1445.98	1.85	29061.8	7210.7	1 - 2	2 - 5.3	7899.0
R 2-2-30	12 (10)	1467.41	1.93	6123.8	339.6	1 - 2	2 - 6.4	3703.3
R 2-3-15	12 (12)	1287.28		1232.7	170.4	1 - 2	2 - 4.7	35.5
R 2-3-20	12 (12)	1343.62		10382.8	1339.9	1 - 2	2 - 5.3	1682.4
R 2-3-25	12 (5)	1438.88	1.85	31144.8	9742.3	1 - 2	2 - 6.0	9496.1
R 2-3-30	12 (7)	1465.22	1.47	16230.7	2970.3	1 - 2	2 - 6.5	6820.6
R 2-4-15	12 (12)	1283.42		2274.6	494.1	1 - 2	2 - 4.7	70.9
R 2-4-20	12 (12)	1339.74		6579.3	1389.9	1 - 2	2 - 5.3	600.1
R 2-4-25	12 (5)	1443.87	3.52	25950.4	5735.3	1 - 2	2 - 5.5	8554.9
R 2-4-30	12 (8)	1468.42	6.3	10935.9	653.9	1 - 2	2 - 6.3	7386.1
R	144 (114)	1386.83	2.78	12358.8	2638.5	1 - 2	2 - 5.4	3927.3
RC 2-2-15	8 (8)	1619.35		688.4	70.5	1 - 2	2 - 3.0	56.5
RC 2-2-20	8 (8)	1703.50		2932.4	225.8	1 - 2	2 - 4.0	340.6
RC 2-2-25	8 (6)	1738.61	0.56	14717.3	1866.1	1 - 2	2 - 4.1	4735.1
RC 2-2-30	8 (5)	1816.80	1.99	23247.4	6038.9	1 - 2	2 - 5.3	7752.3
RC 2-3-15	8 (8)	1500.08		2692	389	1 - 2	2 - 2.9	419.6
RC 2-3-20	8 (7)	1582.41	0.64	21247.6	5119.8	1 - 2	2 - 4.0	3835.5
RC 2-3-25	8 (7)	1606.95	0.51	10880.5	848	1 - 2	2 - 4.1	3522.7
RC 2-3-30	8 (7)	1754.28	2.47	10965.5	3145.6	1 - 2	2 - 5.1	5470.6
RC 2-4-15	8 (8)	1535.40		1298.8	163.5	1 - 2	2 - 2.9	237.0
RC 2-4-20	8 (7)	1586.90	1.22	16250.6	1828.9	1 - 2	2 - 4.0	3073.8
RC 2-4-25	8 (6)	1607.85	1.96	14707.6	939.5	1 - 2	2 - 4.1	4875.1
RC 2-4-30	8 (4)	1759.20	5.86	18778.3	3234.1	1 - 2	2 - 5.4	8403.6
RC	96 (81)	1650.94	2.62	11533.9	1989.1	1 - 2	2 - 4.1	3560.2
Grand total	348 (281)	1486.58	3.22	9083.1	1671.3	1 - 2	2 - 4.3	3787.5

Out of 348 instances in Set 1, BP returns 322 best solution hits and 281 optimality hits, which are the highest hit rates of all algorithms. Among the heuristics, T*▷B is the most successful one in finding best and optimal solutions (294 and 241, respectively). It also outperforms T▷B in terms of solution quality and computational time. On average, T*▷B gives 2% lower deviation from the best solutions, explores 42% less BB nodes, and uses 9% less computational time compared to T▷B. This shows the advantage of using high quality upper-bounds, good location decisions, and nontrivial candidate columns provided by B▷T in the initial step of T*▷B algorithm. Compared to

Table 3 Numerical results of BP for Set 2 instances.

Inst type/size	#Inst (#opt)	Av obj val	Av subopt gap (%)	Av #pricer calls	Av #BB nodes	Av #CDC-sat	Av #veh1-veh2	Av time (s)
a 2-3-15	5 (5)	1644.69		259.2	81.4	1 - 2.0	2.0 - 5.8	3.1
a 2-3-30	5 (5)	2160.38		1008.6	246.8	1 - 2.0	4.0 - 10.0	22.8
a 2-3-50	5 (5)	2608.32		3485.2	1438.8	1 - 2.0	4.0 - 15.8	284.4
a 2-3-100	5 (4)	4094.16	0.15	12035.6	7206.6	1 - 2.0	8.0 - 31.2	7213.2
a 3-5-15	5 (5)	1615.05		848.8	422.4	1 - 2.0	2.0 - 5.8	11.2
a 3-5-30	5 (5)	2134.52		1710.8	407.2	1 - 2.0	4.0 - 10.0	56.1
a 3-5-50	5 (5)	2519.66		2815	383	1 - 2.0	4.0 - 15.2	274.1
a 3-5-100	5 (2)	3980.19	6.21	11531.6	3903.4	1 - 2.0	8.0 - 30.8	11510.4
a 6-4-15	5 (5)	1591.79		759.2	408.6	1 - 2.0	2.0 - 5.0	9.4
a 6-4-30	5 (5)	2135.60		1735	568.6	1 - 2.0	4.0 - 10.0	49.7
a 6-4-50	5 (5)	2566.32		7257.4	3398	1 - 2.0	4.0 - 16.0	657.0
a 6-4-100	5 (2)	4026.49	9.41	21215.8	6790.2	1 - 2.0	8.0 - 30.4	13093.5
a	60 (53)	2589.76	6.72	5388.5	2104.6	1 - 2.0	4.5 - 15.5	2765.4
b 2-3-15	5 (5)	1653.71		752.6	441.8	1 - 2.0	2.0 - 5.8	8.2
b 2-3-30	5 (5)	2076.40		2641.2	1282.6	1 - 2.0	3.4 - 9.6	88.6
b 2-3-50	5 (5)	2598.79		5951.6	2298.8	1 - 2.0	4.0 - 15.0	524.8
b 2-3-100	5 (1)	4041.89	1.67	8980.6	1916.2	1 - 2.2	7.8 - 29.2	12114.4
b 3-5-15	5 (5)	1603.55		1932.8	1434.8	1 - 2.0	2.0 - 5.4	24.6
b 3-5-30	5 (5)	2155.54		2055	625.8	1 - 2.0	4.0 - 10.0	63.4
b 3-5-50	5 (4)	2553.12	0.39	26898.8	6165.4	1 - 2.0	4.0 - 15.6	3279.0
b 3-5-100	5 (0)	4099.85	5.17	9542.8	3547.8	1 - 2.0	8.0 - 32.0	14400.0
b 6-4-15	5 (5)	1623.17		745.4	437	1 - 2.0	2.0 - 5.8	9.3
b 6-4-30	5 (5)	2164.52		3252.4	1472.6	1 - 2.0	4.0 - 10.0	110.3
b 6-4-50	5 (5)	2590.62		10039.4	4897.2	1 - 2.0	4.0 - 16.0	1047.5
b 6-4-100	5 (1)	4090.26	4.39	12694	5147.4	1 - 2.2	7.8 - 30.4	14273.4
b	60 (46)	2604.29	3.6	7123.9	2472.3	1 - 2.0	4.4 - 15.4	3816.7
c 2-3-15	5 (5)	1617.39		463.2	124.4	1 - 2.0	2.0 - 5.6	6.3
c 2-3-30	5 (5)	2071.11		916	95.4	1 - 2.0	3.6 - 9.6	51.5
c 2-3-50	5 (5)	2545.32		5303.4	1320.6	1 - 2.0	4.0 - 15.2	1870.1
c 2-3-100	5 (0)	4070.00	3.28	4081.4	739	1 - 2.0	8.0 - 30.8	14400.0
c 3-5-15	5 (5)	1581.68		1241.4	547.2	1 - 2.0	2.0 - 5.2	20.7
c 3-5-30	5 (5)	2073.86		1828.4	293.2	1 - 2.0	3.6 - 9.8	128.6
c 3-5-50	5 (4)	2538.09	0.32	26601.6	18439.4	1 - 2.0	4.0 - 16.0	5792.7
c 3-5-100	5 (0)	4037.24	7.5	4411.2	388.8	1 - 2.0	8.0 - 31.8	14400.0
c 6-4-15	5 (5)	1601.88		922.4	491.4	1 - 2.0	2.0 - 5.6	12.2
c 6-4-30	5 (5)	2076.80		2018.6	506.2	1 - 2.0	3.8 - 9.8	114.1
c 6-4-50	5 (5)	2553.14		8483	3323.6	1 - 2.0	4.2 - 16.0	2397.2
c 6-4-100	5 (0)	4057.03	9.03	3250.6	104.8	1 - 2.2	7.8 - 30.8	14400.0
c	60 (44)	2568.63	6.21	4960.1	2197.8	1 - 2.0	4.4 - 15.5	4452.4
d 2-3-15	5 (5)	1629.08		286.2	86	1 - 2.0	2.0 - 5.2	2.5
d 2-3-30	5 (5)	2082.91		557.8	60.6	1 - 2.0	3.6 - 9.6	10.1
d 2-3-50	5 (5)	2611.14		2561.6	644.6	1 - 2.0	4.0 - 15.8	210.7
d 2-3-100	5 (3)	4052.20	3.12	9969.6	4503	1 - 2.0	8.0 - 30.2	8626.8
d 3-5-15	5 (5)	1607.05		806	396.4	1 - 2.0	2.0 - 5.6	10.1
d 3-5-30	5 (5)	2149.84		1725.6	414.8	1 - 2.0	4.0 - 10.2	46.8
d 3-5-50	5 (5)	2542.38		8696.8	4172.6	1 - 2.0	4.0 - 16.0	872.3
d 3-5-100	5 (1)	3986.58	3.18	12273.4	4440.6	1 - 2.0	8.0 - 30.0	13357.6
d 6-4-15	5 (5)	1607.54		801.2	440	1 - 2.0	2.0 - 5.6	8.3
d 6-4-30	5 (5)	2149.15		1572.6	476.2	1 - 2.0	4.0 - 10.2	48.0
d 6-4-50	5 (5)	2562.51		4344	1252.4	1 - 2.0	4.0 - 16.0	359.9
d 6-4-100	5 (3)	4017.73	0.32	13442.4	5670.2	1 - 2.2	7.8 - 30.8	10600.1
d	60 (52)	2583.18	2.45	4753.1	1879.8	1 - 2.0	4.5 - 15.4	2846.1
Grand total	240 (195)	2586.46	4.81	5556.4	2163.6	1 - 2.0	4.5 - 15.5	3470.2

BP (see Table 2), T*⊳B makes less computational effort, on average, as it explores less number of BB nodes (1459.9 versus 1671.3) and uses less time to find the solution (3156.1 sec. versus 3787.5 sec.). On average, T*⊳B deviates 0.04% from the best available solution. Note that around 82% of the best solutions of T*⊳B are proved to be optimal by BP. Therefore, one can conclude that T*⊳B produces solutions that are compatible with BP in terms of quality.

Considering the solution quality, B⊳T cannot find as many best or optimal solutions as other approaches, however it is able to find solutions that deviate from the best ones by only 0.43%. In terms of solution time, B⊳T is significantly faster than the other approaches. The average computational time of B⊳T is 4.5 seconds and almost all B⊳T solutions are found in the root node of the branch-and-bound tree.

Experimental results over Set 2 instances are given in Table 5. We report only T*⊳B over T⊳B as it outperforms T⊳B in most cases (see Table 4). Out of 240 instances in Set 2, BP finds the highest number of best and optimal solutions (207 and 195, respectively) and, on average, deviates 0.2% from the best solution. T*⊳B is able to find the best available solution for 139 instances. 75% of the best solutions found by this algorithm are optimal. Although the optimality hit rate of T*⊳B is less than that of BP, this algorithm is able to find a better solution for many cases where BP does not close the optimality gap in the given time. On average, T*⊳B solutions deviate only 0.09% from the best available solutions.

In terms of computational effort, T*⊳B proves to be more efficient than BP. It explores 80% less BB nodes, requires 60% less pricing calls, and reports 60% less solution times compared to BP. Table 5 also presents *Av root gap(%)* values to show the average percentage of MIP gap obtained after solving the root node by BP and T*⊳B. T*⊳B benefits from small initial MIP gaps by fixing binary location variables. Therefore, if good location decisions are provided to this algorithm, it can find a reliable solution fast.

B⊳T showed a promising performance by generating high quality solutions in short computational times. For small size instances, it was able to find high number of optimal solutions, and for larger ones, its solutions are not far from the best ones. Around 26% of the instances in Set 2 were solved to optimality by B⊳T. The overall *Av %dev from best* value for B⊳T is 1.09. Note that 81% of the best solutions are found to be optimal by BP. On average, B⊳T required 13.5 sec. to return a solution.

The above observations indicate: (i) BP is successful to find the optimal solution of around 80% of the instances in Set 1 and Set 2. For the instances for which the exact solution is not guaranteed, BP reports small MIP gaps, on average. (ii) T*⊳B shows a notable performance in terms of both solution quality and computational effort. (iii) B⊳T solutions are close to the optimal solutions or the best ones found by our algorithms. This algorithm uses significantly less computational time than the others. Therefore, one can rely on B▷T to find a fast but good approximation of the 2E-LRPTW solutions. B▷T solutions also provide a high quality information that can be used as a starting point for other approaches that seek more precise solutions. The advantage of using this information is demonstrated in T*▷B results.

Table 4 Numerical results of the proposed heuristics for Set 1 instances.

Inst	#best(#opt)				Av %dev from best				Av #pricer calls		Av #BB nodes			Av time (s)		
type/size	BP	T▷B	T*▷B	B▷T	BP	T▷B	T*▷B	B▷T	T▷B	T*▷B	T▷B	T*▷B	B▷T	T▷B	T*▷B	B▷T
C 2-2-15	9(9)	9(9)	8(8)	2(2)	0	0	0.02	0.52	171.8	368.2	1.0	1.0	1.0	12.9	30.7	1.3
C 2-2-20	9(8)	8(8)	9(8)	1(0)	0	0.16	0	1.02	1077.0	1179.6	76.4	45.3	1.0	1990.3	2213.9	3.4
C 2-2-25	9(7)	7(7)	8(6)	4(2)	0	0.95	0.02	0.25	1273.0	1570.1	72.1	44.9	1.0	3866.2	3734.9	5.9
C 2-2-30	9(6)	3(3)	8(5)	3(0)	0	13.25	0.03	0.41	11474.3	4370.0	101.9	142.9	1.0	6787.4	7018.0	6.8
C 2-3-15	9(9)	9(9)	8(8)	4(4)	0	0	0.02	0.24	167.1	413.1	1.0	1.0	1.0	11.1	37.6	1.8
C 2-3-20	9(7)	7(7)	8(7)	0(0)	0	4.11	0.03	1.23	1628.8	1891.9	131.6	100.1	1.0	3292.4	3321.3	9.5
C 2-3-25	9(6)	6(5)	6(4)	4(3)	0	6.14	0	0.11	2878.9	2930.4	125.9	68.3	1.0	4475.8	4303.5	6.5
C 2-3-30	9(6)	5(5)	7(4)	3(0)	0	4.36	0.02	0.54	2680.0	4458.4	103.1	96.3	1.0	5780.9	6851.4	6.8
C 2-4-15	9(9)	4(4)	7(7)	5(5)	0	0.06	0.02	0.09	153.1	417.0	1.0	1.0	1.0	13.6	38.9	2.5
C 2-4-20	8(7)	8(7)	7(6)	1(0)	0.07	0.02	0.03	0.37	3523.0	4791.7	316.4	406.7	1.0	4583.7	4611.0	5.4
C 2-4-25	8(7)	3(3)	6(4)	4(3)	0.02	4.55	0.05	0.15	885.3	2244.9	38.1	23.4	1.0	3857.3	3691.1	5.0
C 2-4-30	7(5)	7(5)	7(4)	2(0)	0.05	5.16	0.02	0.26	2133.1	4257.2	79.4	55.3	1.0	5302.3	6426.7	6.9
C	104(86)	76(72)	89(71)	33(19)	0.01	3.23	0.02	0.43	2337.1	2407.7	87.3	82.2	1.0	3331.2	3523.3	5.2
R 2-2-15	12(12)	12(12)	12(12)	11(11)	0	0	0	0.01	784.7	705.1	160.3	86.8	1.0	16.8	16.1	1.0
R 2-2-20	12(12)	8(8)	12(12)	5(5)	0	0.05	0	0.25	7240.3	7178.8	1710.3	1240.3	1.0	753.9	873.6	2.4
R 2-2-25	10(7)	8(7)	6(4)	1(1)	0.04	0.11	0.28	0.88	25681.8	26996.7	8247.4	7184.2	1.0	7145.9	8591.3	6.0
R 2-2-30	11(10)	7(6)	9(8)	0(0)	0.12	3.35	0.08	1.80	5295.2	5706.2	413.0	225.6	1.0	2826.0	3335.0	5.1
R 2-3-15	12(12)	0(0)	12(12)	8(8)	0	1.86	0	0.15	920.8	755.0	205.8	75.9	1.2	18.3	18.2	1.4
R 2-3-20	12(12)	1(1)	11(11)	4(4)	0	1.86	0.01	0.30	12056.1	6891.8	4106.3	1193.0	1.0	1116.5	1596.6	2.9
R 2-3-25	8(5)	0(0)	10(4)	2(1)	0.14	1.73	0.06	0.47	42606.6	19385.4	20700.3	5086.8	1.0	10617.6	5444.9	7.8
R 2-3-30	10(7)	0(0)	5(3)	0(0)	0.20	5.74	0.25	1.31	17645.9	13546.3	1888.5	2706.0	1.0	8499.9	5769.5	7.0
R 2-4-15	12(12)	3(3)	12(12)	9(9)	0	0.72	0	0.22	1608.9	1190.2	407.0	210.3	1.0	38.2	27.5	2.1
R 2-4-20	12(12)	7(7)	10(10)	7(7)	0	0.17	0.03	0.29	6298.1	3898.4	2399.0	932.1	1.0	340.5	239.3	3.4
R 2-4-25	9(5)	0(0)	10(4)	3(2)	0.15	10.63	0.03	0.40	27754.8	22453.9	8035.3	5478.8	1.0	8499.3	7282.8	15.2
R 2-4-30	9(8)	2(2)	8(4)	0(0)	0.17	0.90	0.12	1.39	12590.4	5303.8	1403.9	216.4	1.0	5205.9	3391.6	10.3
R	129(114)	48(46)	117(96)	50(48)	0.07	2.26	0.07	0.62	13373.6	9501.0	4139.8	2053.0	1.0	3756.6	3048.9	5.4
RC 2-2-15	8(8)	8(8)	7(7)	4(4)	0	0	0.01	0.09	362.5	549.0	31.4	39.9	1.0	25.9	43.8	1.0
RC 2-2-20	8(8)	6(6)	7(7)	4(4)	0	0.06	0.01	0.09	1479.3	1935.0	152.9	125.5	1.0	146.8	249.1	1.1
RC 2-2-25	8(6)	6(4)	7(5)	3(2)	0	0.05	0	0.15	8855.3	12889.0	1366.0	1948.4	1.0	1759.6	3798.7	1.5
RC 2-2-30	7(5)	4(3)	8(5)	3(2)	0	0.35	0	0.44	22927.9	23757.6	8533.4	7355.9	1.0	10066.6	8693.5	2.8
RC 2-3-15	8(8)	8(8)	8(8)	7(7)	0	0	0	0.01	2802.5	2943.9	497.1	414.8	1.0	319.3	426.4	1.7
RC 2-3-20	8(7)	8(7)	8(7)	7(7)	0	0	0	0.01	18146.1	24021.0	5557.4	5723.9	1.0	2623.2	3837.2	3.0
RC 2-3-25	8(7)	8(7)	8(7)	6(5)	0	0	0	0.10	4602.1	7005.5	361.1	434.4	1.0	1172.3	1943.4	1.6
RC 2-3-30	7(7)	0(0)	7(6)	5(5)	0	3.83	0.01	0.31	25175.4	6618.9	7324.5	1020.5	1.0	9193.7	3537.3	2.7
RC 2-4-15	8(8)	8(8)	6(6)	6(6)	0	0	0.05	0.08	554.0	690.4	51.3	46.8	1.0	54.9	72.9	2.2
RC 2-4-20	8(7)	7(6)	8(7)	6(5)	0	0.02	0	0.08	12148.3	10966.1	1861.8	1398.0	1.0	1568.6	1340.9	3.3
RC 2-4-25	7(6)	7(5)	8(6)	5(5)	0.05	0.01	0	0.08	13869.0	14114.1	1645.5	1048.1	1.0	3002.8	2998.2	3.1
RC 2-4-30	4(4)	6(3)	6(3)	3(3)	0.28	0.32	0.03	0.29	21868.4	23390.6	5607.6	5885.5	1.0	8702.1	7907.2	5.5
RC	89(81)	76(65)	88(74)	59(55)	0.03	0.39	0.01	0.14	11065.9	10740.1	2749.2	2120.1	1.0	3219.6	2904.1	2.5
Grand total	322(281)	200(183)	294(241)	142(122)	0.04	2.04	0.04	0.43	9311.9	7641.4	2498.5	1459.9	1.0	3476.4	3156.1	4.5

5.6.1. Bottom-to-Top Approach for the LRPTW. Our computational experiments show that B▷T is an efficient algorithm that can find acceptable solutions in very short times. This algorithm is also flexible and can be used to solve the routing problems for which a path-based set-partitioning formulation is available. If needed, small modifications can be made to adopt different route feasibility conditions and constraints in the clustering step and implement B▷T to tackle a wide range of LRP and VRPs. In this section, we implement this algorithm to solve the LRPTW introduced in Farham, Süral, and Iyigun (2018), and investigate its performance.

Table 5 Numerical results of the proposed heuristics for Set 2 instances.

Inst	#best(#opt)			Av %dev from best			Av #pricer calls	Av #BB nodes		Av root gap (%)		Av time (s)	
type/size	BP	T*▷B	B▷T	BP	T*▷B	B▷T	T*▷B	T*▷B	B▷T	BP	T*▷B	T*▷B	B▷T
a 2-3-15	5(5)	5(5)	5(5)	0	0	0	94.8	12.0	1.0	3.32	1.52	1.6	1.1
a 2-3-30	5(5)	3(3)	0(0)	0	0.06	0.27	530.4	92.6	1.0	11.43	2.11	11.7	3.3
a 2-3-50	5(5)	1(1)	0(0)	0	0.11	1.23	1672.6	350.6	1.0	5.12	1.70	124.9	3.1
a 2-3-100	5(4)	0(0)	0(0)	0	0.20	2.86	2506.0	334.2	1.0	8.01	3.33	1070.5	10.3
a 3-5-15	5(5)	5(5)	5(5)	0	0	0	145.0	21.4	1.0	5.67	1.79	3.6	2.7
a 3-5-30	5(5)	3(3)	0(0)	0	0.05	0.21	395.0	47.0	10.0	15.55	2.82	17.5	11.3
a 3-5-50	5(5)	2(2)	0(0)	0	0.07	1.27	734.8	30.8	5.6	11.34	1.74	57.1	13.8
a 3-5-100	3(2)	2(0)	0(0)	0.50	0.20	2.32	4118.6	1230.0	3.2	14.65	3.06	2575.8	23.8
a 6-4-15	5(5)	4(4)	4(4)	0	0.03	0.03	103.2	7.6	1.0	4.81	1.21	2.0	1.3
a 6-4-30	5(5)	4(4)	1(1)	0	0.06	0.18	301.2	25.2	16.8	14.98	2.11	14.1	9.6
a 6-4-50	5(5)	2(2)	0(0)	0	0.06	1.01	1685.2	246.6	1.0	10.47	2.13	131.9	11.9
a 6-4-100	2(2)	3(0)	0(0)	0.85	0.15	2.36	5300.8	1500.0	5.6	14.80	3.01	3803.4	21.7
a	55(53)	34(29)	15(15)	0.11	0.08	0.98	1465.6	324.8	4.0	10.01	2.21	651.2	9.5
b 2-3-15	5(5)	5(5)	5(5)	0	0	0	518.6	298.0	1.0	3.49	2.27	5.8	1.9
b 2-3-30	5(5)	5(5)	0(0)	0	0	0.69	2419.2	1135.2	1.8	9.87	2.23	89.7	32.4
b 2-3-50	5(5)	0(0)	0(0)	0	0.27	1.86	2663.2	678.2	2.6	6.49	2.81	217.9	16.2
b 2-3-100	1(1)	4(0)	0(0)	0.54	0.05	3.47	8485.4	2488.2	4.8	10.79	4.95	8731.0	29.5
b 3-5-15	5(5)	5(5)	5(5)	0	0	0	636.4	426.0	1.0	5.12	1.72	8.2	2.7
b 3-5-30	5(5)	2(2)	0(0)	0	0.17	0.53	632.4	147.6	10.2	16.72	3.14	22.7	11.5
b 3-5-50	5(4)	2(2)	0(0)	0	0.17	1.90	4292.2	1541.2	1.0	12.67	2.74	412.7	38.7
b 3-5-100	0(0)	5(0)	0(0)	2.00	0	4.13	6871.6	2498.4	6.6	14.73	5.19	4539.5	35.2
b 6-4-15	5(5)	5(5)	5(5)	0	0	0	133.6	22.8	6.6	5.97	1.54	2.9	2.0
b 6-4-30	5(5)	0(0)	0(0)	0	0.18	0.46	775.8	172.8	17.0	16.45	2.56	48.2	33.6
b 6-4-50	5(5)	1(1)	0(0)	0	0.07	1.34	1522.8	295.4	3.8	10.04	2.43	99.9	11.8
b 6-4-100	1(1)	4(0)	0(0)	1.27	0.04	3.88	7714.4	2256.0	10.0	16.11	4.74	8250.2	54.7
b	47(46)	38(25)	15(15)	0.32	0.08	1.52	3055.5	996.7	5.5	10.70	3.03	1869.1	22.5
c 2-3-15	5(5)	5(5)	4(4)	0	0	0.02	162.8	18.6	1.0	4.00	2.35	3.3	2.2
c 2-3-30	5(5)	3(3)	1(1)	0	0.09	0.25	322.8	6.4	2.2	6.77	1.45	15.8	3.2
c 2-3-50	5(5)	0(0)	0(0)	0	0.11	1.13	2474.0	332.8	1.0	5.43	1.89	904.3	7.7
c 2-3-100	1(0)	4(0)	0(0)	0.96	0.04	2.38	2604.8	282.6	1.0	7.68	2.94	6158.6	14.1
c 3-5-15	5(5)	5(5)	5(5)	0	0	0	124.0	7.8	1.0	6.45	1.80	3.8	2.8
c 3-5-30	5(5)	2(2)	1(1)	0	0.10	0.19	420.4	29.2	1.0	13.02	1.65	24.7	9.4
c 3-5-50	5(4)	1(1)	0(0)	0	0.16	0.97	3618.8	649.6	1.0	12.30	2.18	816.6	10.8
c 3-5-100	0(0)	5(0)	0(0)	1.62	0	2.27	3021.8	344.0	1.0	11.45	3.13	6971.9	29.2
c 6-4-15	5(5)	5(5)	4(4)	0	0	0	243.4	63.2	6.2	6.41	1.70	4.1	2.3
c 6-4-30	5(5)	2(2)	1(1)	0	0.03	0.20	480.6	56.4	23.0	14.14	2.24	33.8	17.6
c 6-4-50	5(5)	0(0)	0(0)	0	0.16	0.83	2097.0	250.8	7.4	9.44	1.80	486.2	15.5
c 6-4-100	2(0)	5(0)	2(0)	1.12	0	1.12	3658.4	369.4	3.4	9.97	2.67	11757.5	32.0
c	48(44)	37(23)	18(16)	0.31	0.06	0.78	1602.4	200.9	4.1	8.92	2.15	2265.0	12.2
d 2-3-15	5(5)	5(5)	4(4)	0	0	0.05	139.0	20.0	1.0	3.23	1.69	1.4	0.6
d 2-3-30	5(5)	3(3)	0(0)	0	0.07	0.29	268.8	9.8	2.4	8.79	1.41	4.8	1.7
d 2-3-50	5(5)	1(1)	0(0)	0	0.32	1.11	17445.4	49.2	1.0	5.17	1.70	2736.1	10.2
d 2-3-100	4(3)	1(0)	0(0)	0.32	0.12	2.83	2925.0	486.6	1.0	8.36	3.23	1545.0	8.7
d 3-5-15	5(5)	5(5)	5(5)	0	0	0	105.4	15.4	1.0	6.55	2.38	4.9	4.4
d 3-5-30	5(5)	4(4)	1(1)	0	0.05	0.29	537.2	72.8	6.8	14.76	2.02	16.8	9.0
d 3-5-50	5(5)	1(1)	0(0)	0	0.36	1.29	2299.6	516.2	1.0	12.16	2.15	169.2	10.2
d 3-5-100	3(1)	2(0)	0(0)	0.46	0.10	3.33	4035.2	917.8	1.0	12.81	3.89	2441.7	28.4
d 6-4-15	5(5)	5(5)	5(5)	0	0	0	130.4	22.2	1.4	6.44	1.75	3.4	2.7
d 6-4-30	5(5)	2(2)	1(1)	0	0.12	0.30	464.2	52.6	23.0	14.58	1.73	16.8	9.4
d 6-4-50	5(5)	1(1)	0(0)	0	0.09	1.17	916.8	55.2	5.0	10.60	2.34	70.8	18.0
d 6-4-100	5(3)	0(0)	0(0)	0	0.37	2.27	3309.8	365.8	6.2	13.76	2.73	1946.9	15.4
d	57(52)	30(27)	16(16)	0.07	0.13	1.08	2714.7	215.3	4.2	9.77	2.25	746.5	9.9
Grand total	207(195)	139(104)	64(62)	0.20	0.09	1.09	2209.6	434.4	4.5	9.85	2.41	1382.9	13.5

Since the LRPTW only contains a single echelon, the master problem has to be modified in order to contain satellite facilities and not CDCs. Therefore, the variable and constraints regarding the first echelon are removed. However, the procedure to find candidate variables in $\mathcal{P}_j, \forall j \in \mathcal{J}$, remains the same. B▷T approach for the LRPTW uses the clustering method to find vehicle routes and then, solves the master problem of the LRPTW (see Farham, Süral, and Iyigun 2018) over the generated routes.

The LRPTW test instances in Farham, Süral, and Iyigun (2018) consist of four sets. The first set includes 36 test instances containing up to 3 candidate depot locations and 40 randomly distributed customers. The second set has larger instances with up to 5 candidate depot locations and 50 randomly distributed customers. The other two sets contain at most 5 candidate depots and 50 clustered customers. All instances in Farham, Süral, and Iyigun (2018) have nonzero facility location and vehicle fixed costs. We call the exact solution algorithm of Farham, Süral, and Iyigun (2018) as FSI. For each instance set, Table 6 shows (i) *Obj val* (*Min*, *Max* and *Av %dev*): the minimum, maximum, and average percent deviation of the objective function value found by B▷T from FSI, (ii) *Total #depots* and *Total #vehicles*: the total number of open depots and the total number of vehicle routes found by each algorithm, and (iii) *Av time*: the average computational time reported by FSI and B▷T (in seconds). FSI was provided 6 hours of time limit and was run on the same computer as ours.

Table 6 Numerical results of running B▷T on LRPTW test instances.

Inst set	#	Obj val			Total #depots		Total #vehicles		Av time (s)	
(type)	Insts	Min %dev	Max %dev	Av %dev	FSI	B▷T	FSI	B▷T	FSI	B▷T
1 (R)	36	0.14	5.51	1.59	72	72	162	172	4625.0	7.5
2 (R)	48	-0.63	11.29	5.62	127	132	351	405	11151.1	24.9
3 (C)	27	-13.55	3.93	-1.07	84	81	123	127	6910.6	15.4
4 (C)	27	-13.63	5.43	-0.38	83	81	124	128	5165.9	17.3
Grand Total	138			2.09					7448.2	17.0

B▷T yields an average deviation of 2.09% from the FSI solutions. Since facility locations are considered as strategic decisions that make up a large portion of the total cost in the LRPTW instances, it is important to make a correct decision about the number and the location of open facilities. Table 6 shows that B▷T is successful in keeping the number of open depots close to the ones reported in the literature, while it uses 0.52 more vehicles per instance (72 more vehicles are used in total of 138 instances). Negative deviation values indicate that our algorithm is able to improve the solutions of FSI for some instances. When the performance of B▷T is analyzed according to the instance types, it finds high quality solutions for C type instances, some of which dominate the ones reported by FSI. The most challenging set for B▷T to solve is the one corresponding to the second row of Table 6. It contains instances with many randomly distributed customers. For 48 test instances in this set, the average deviation is returned as 5.62%. Higher deviations are expected for a clustering-based heuristic when both locations and time windows of customers are randomly generated. However, if the computational times matter, B▷T can provide efficient solutions in any instance type. The results indicate that B▷T takes only 0.23% of the FSI reported times to find the solution. The average time to solve an LRPTW test instance in Table 6 is 17 sec.

We also list the solution details for which B▷T found a better result than FSI in Table 7.

Table 7 **Improved solutions for the LRPTW test instances.**

Inst set	Inst	Inst size	Obj val	#depots	#vehicles	time (s)
2	R110	0-5-50	8238.0	4	7	5.5
3	C102	0-5-50	7615.2	4	6	21.1
3	C103	0-5-50	7617.0	4	6	47.3
3	C104	0-5-50	7622.4	4	6	97.9
3	C109	0-5-50	7606.9	4	6	45.2
4	C103	0-5-50	7636.6	4	6	58.8
4	C104	0-5-50	7642.0	4	6	94.7
4	C109	0-5-50	7614.4	4	6	66.5

6. Conclusion

In this study, we consider the two-echelon location-routing problem under capacity and time window constraint (2E-LRPTW) to address main strategic and tactical-level decisions in urban freight transportation systems from an operations research perspective. Despite the importance of this problem, a very limited amount of research has been done that provide closed-form formulation and suggest effective solution approaches. We present a path-based formulation for the problem that is solved by an exact branch-and-price-based algorithm. Different enhancement techniques are proposed and the optimal solutions are found for the instances with up to 3 candidate CDC locations, 5 candidate satellite locations, and 100 customers. Although the exact approach is successful in solving small and medium-size problem instances, it struggles to find the optimal solution for larger problems in a reasonable time. Therefore, we present two heuristics based on the mathematical structure of the problem, which reduce the original problem in different ways. One way is to make the facility location decisions first, and solve for routing decisions next. However, as problems with routing decisions are shown to be difficult combinatorial problems, this stage becomes computationally expensive. Another way is to estimate routing decisions first, and find the optimal facility location and select the best routes in the next step. Once a candidate set of the most detailed decisions (i.e. vehicle routes) is determined, solving the problem becomes more straightforward. We show that customers can be clustered into vehicle routes based on not only their spatial characteristic but also their temporal attribute (i.e. time window). The experimental results indicate that the latter approach is highly successful in solving problem instances with different size and characteristics. Therefore, we highlight the importance of taking tactical-level decisions into account while making strategic decisions in such a complex system. The effective clustering method proposed in this study can be used to find an initial solution and/or a tight upper bound for the problem in a relatively short computational time.

One of the most important applications of the 2E-LRPTW is freight transportation network design in city logistics. In this area, air-pollution and energy consumption concerns, parking restrictions in densely inhabitant areas of the city, and multi-trip vehicle routes option are among the most interesting factors to consider. We believe that the current study can be a starting point to formulate richer two-echelon city logistics problems and develop practical solution approaches.

Acknowledgments

This study is a part of research conducted in CONCOORD project of JPI Urban Europe. The authors thank The Scientific and Technological Research Council of Turkey (TÜBİTAK) for their financial support (project no. 113M121).

References

Ambrosino D, Scutellà MG, 2005 *Distribution network design: New problems and related models. European Journal of Operational Research* 165(3):610–624.

Anderluh A, Hemmelmayr VC, Nolz PC, 2017 *Synchronizing vans and cargo bikes in a city distribution network. Central European Journal of Operations Research* 25(2):345–376.

Bala K, Brcanov D, Gvozdenović N, 2017 *Two-echelon location routing synchronized with production schedules and time windows. Central European Journal of Operations Research* 25(3):525–543.

Baldacci R, Mingozzi A, Roberti R, 2012 *Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. European Journal of Operational Research* 218(1):1–6.

Baldacci R, Mingozzi A, Roberti R, Calvo RW, 2013 *An exact algorithm for the two-echelon capacitated vehicle routing problem. Operations Research* 61(2):298–314.

Boccia M, Crainic TG, Sforza A, Sterle C, 2010 *A metaheuristic for a two echelon location-routing problem.* Festa P, ed., *Experimental Algorithms*, 288–301 (Berlin: Springer).

Bräysy O, Gendreau M, 2005 *Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. Transportation Science* 39(1):104–118.

Breunig U, Schmid V, Hartl R, Vidal T, 2016 *A large neighbourhood based heuristic for two-echelon routing problems. Computers & Operations Research* 76:208–225.

Contardo C, Desaulniers G, Lessard F, 2015 *Reaching the elementary lower bound in the vehicle routing problem with time windows. Networks* 65(1):88–99.

Contardo C, Hemmelmayr V, Crainic TG, 2012 *Lower and upper bounds for the two-echelon capacitated location-routing problem. Computers & Operations Research* 39(12):3185–3199.

Crainic TG, Mancini S, Perboli G, Tadei R, 2011 *Multi-start heuristics for the two-echelon vehicle routing problem.* Merz P, Hao JK, eds., *Evolutionary Computation in Combinatorial Optimization*, volume 6622, 179–190 (Berlin: Springer).

Crainic TG, Mancini S, Perboli G, Tadei R, 2013 *GRASP with path relinking for the two-echelon vehicle routing problem.* Di Gaspero L, Schaerf A, Sttzle T, eds., *Advances in Metaheuristics*, 113–125 (New York: Springer).

Crainic TG, Perboli G, Mancini S, Tadei R, 2010 *Two-echelon vehicle routing problem: A satellite location analysis. Procedia - Social and Behavioral Sciences* 2(3):5944–5955, 6th International Conference on City Logistics.

Crainic TG, Ricciardi N, Storchi G, 2004 *Advanced freight transportation systems for congested urban areas. Transportation Research Part C: Emerging Technologies* 12(2):119–137.

Crainic TG, Ricciardi N, Storchi G, 2009 *Models for evaluating and planning city logistics systems. Transportation Science* 43(4):432–454.

Cuda R, Guastaroba G, Speranza M, 2015 *A survey on two-echelon routing problems. Computers & Operations Research* 55:185–199.

Dabia S, Ropke S, Van Woensel T, De Kok T, 2013 *Branch and price for the time-dependent vehicle routing problem with time windows. Transportation Science* 47(3):380–396.

Dellaert N, Dashty Saridarq F, Van Woensel T, Crainic TG, 2019 *Branch-and-price–based algorithms for the two-echelon vehicle routing problem with time windows. Transportation Science* 53(2):463–479.

Desaulniers G, Lessard F, Hadjar A, 2008 *Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transportation Science* 42(3):387–404.

Desaulniers G, Solomon MM, Desrosiers J, 2005 *Column Generation* (Springer).

Desrochers M, Desrosiers J, Solomon M, 1992 *A new optimization algorithm for the vehicle routing problem with time windows. Operations Research* 40(2):342–354.

Dondo R, Cerdá J, 2007 *A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. European Journal of Operational Research* 176(3):1478–1507.

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

Farahani RZ, Hekmatfar M, Fahimnia B, Kazemzadeh N, 2014 *Hierarchical facility location problem: Models, classifications, techniques, and applications. Computers & Industrial Engineering* 68:104–117.

Farham MS, Süral H, Iyigun C, 2018 *A column generation approach for the location-routing problem with time windows. Computers & Operations Research* 90:249–263.

Feillet D, Dejax P, Gendreau M, Gueguen C, 2004 *An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks* 44(3):216–229.

Gleixner A, Bastubbe M, Eifler L, Gally T, Gamrath G, Gottwald RL, Hendel G, Hojny C, Koch T, Lübbecke ME, Maher SJ, Miltenberger M, Müller B, Pfetsch ME, Puchert C, Rehfeldt D, Schlösser F, Schubert C, Serrano F, Shinano Y, Viernickel JM, Walter M, Wegscheider F, Witt JT, Witzig J, 2018 *The SCIP Optimization Suite 6.0.* Technical report, Optimization Online, URL `http://www.optimization-online.org/DB_HTML/2018/07/6692.html`.

Grangier P, Gendreau M, Lehud F, Rousseau LM, 2016 *An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. European Journal of Operational Research* 254(1):80–91.

Gündüz HI, 2015 *Optimization of a two-stage distribution network with route planning and time restrictions. Proceedings of the 48th Hawaii International Conference on System Sciences*, 1088–1097, HICSS 2015 (Piscataway, NJA: IEEE Computer Society).

Hemmelmayr VC, Cordeau JF, Crainic TG, 2012 *An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Computers & Operations Research* 39(12):3215–3228.

IBM, 2018 *CPLEX Optimization Studio V12.8.* `http://www-01.ibm.com/support/docview.wss?uid=swg27050618`, date accessed: July 1, 2018.

Irnich S, Desaulniers G, 2005 *Shortest path problems with resource constraints*, 33–65 (Boston, MA: Springer US).

Jepsen M, Spoorendonk S, Ropke S, 2013 *A branch-and-cut algorithm for the symmetric two-echelon capacitated vehicle routing problem. Transportation Science* 47(1):23–37.

Klose A, Drexl A, 2005 *Facility location models for distribution system design. European Journal of Operational Research* 162(1):4–29.

Laporte G, Semet F, 2002 *Classical heuristics for the capacitated VRP*. Toth P, Vigo D, eds., *The vehicle routing problem*, 109–128 (Philadelphia, PA, USA: SIAM).

Li H, Zhang L, Lv T, Chang X, 2016 *The two-echelon time-constrained vehicle routing problem in linehaul-delivery systems. Transportation Research Part B: Methodological* 94:169–188.

Lozano L, Duque D, Medaglia AL, 2016 *An exact algorithm for the elementary shortest path problem with resource constraints. Transportation Science* 50(1):348–357.

Lübbecke ME, Desrosiers J, 2005 *Selected topics in column generation. Operations Research* 53(6):1007–1023.

Neame PJ, 1999 *Nonsmooth dual methods in integer programming*. Ph.D. thesis, University of Melbourne, Department of Mathematics and Statistics, Australia.

Nguyen VP, Prins C, Prodhon C, 2010 *A multi-start evolutionary local search for the two-echelon location routing problem*. Blesa M, Blum C, Raidl G, Roli A, Sampels M, eds., *Hybrid Metaheuristics*, 88–102 (Berlin: Springer).

Nguyen VP, Prins C, Prodhon C, 2012a *A multi-start iterated local search with tabu list and path relinking for the two-echelon location-routing problem. Engineering Applications of Artificial Intelligence* 25(1):56–71.

Nguyen VP, Prins C, Prodhon C, 2012b *Solving the two-echelon location routing problem by a GRASP reinforced by a learning process and path relinking. European Journal of Operational Research* 216(1):113–126.

Pecin D, Contardo C, Desaulniers G, Uchoa E, 2017 *New enhancements for the exact solution of the vehicle routing problem with time windows. INFORMS Journal on Computing* 29(3):489–502.

Perboli G, Tadei R, Vigo D, 2011 *The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. Transportation Science* 45(3):364–380.

Pessoa A, Sadykov R, Uchoa E, Vanderbeck F, 2013 *In-out separation and column generation stabilization by dual price smoothing*. Bonifaci V, Demetrescu C, Marchetti-Spaccamela A, eds., *Experimental Algorithms*, 354–365 (Berlin, Heidelberg: Springer).

Prodhon C, Prins C, 2014 *A survey of recent research on location-routing problems. European Journal of Operational Research* 238(1):1–17.

Pugacs S, 2014 *A clustering approach for vehicle routing problems with hard time windows*. Master's thesis, Faculdade de Ciencias e Tecnologia, Lisbon.

Qi M, Lin WH, Li N, Miao L, 2012 *A spatiotemporal partitioning approach for large-scale vehicle routing problems with time windows. Transportation Research Part E: Logistics and Transportation Review* 48(1):248–257.

Rahmani Y, Cherif-Khettaf WR, Oulamara A, 2016 *The two-echelon multi-products location-routing problem with pickup and delivery: Formulation and heuristic approaches. International Journal of Production Research* 54(4):999–1019.

Ryan DM, Hjorring C, Glover F, 1993 *Extensions of the petal method for vehicle routing. Journal of the Operational Research Society* 44(3):289–296.

Santos FA, Mateus GR, da Cunha AS, 2015 *A branch-and-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. Transportation Science* 49(2):355–368.

Schwengerer M, Pirkwieser S, Raidl G, 2012 *A variable neighborhood search approach for the two-echelon location-routing problem*. Hao JK, Middendorf M, eds., *Evolutionary Computation in Combinatorial Optimization*, 13–24 (Berlin: Springer).

Solomon MM, 1987 *Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research* 35(2):254–265.

Taniguchi E, Thompson RG, 2002 *Modeling city logistics. Transportation Research Record: Journal of the Transportation Research Board* 1790:45–51.

Wang Y, Assogba K, Liu Y, Ma X, Xu M, Wang Y, 2018 *Two-echelon location-routing optimization with time windows based on customer clustering. Expert Systems with Applications* 104:244–260.

Zhou L, Baldacci R, Vigo D, Wang X, 2018 *A multi-depot two-echelon vehicle routing problem with delivery options arising in the last mile distribution. European Journal of Operational Research* 265(2):765–778.