A Neighborhood Search and Set Cover Hybrid Heuristic for the Two-Echelon Vehicle Routing Problem

Youcef Amarouche[1]
Sorbonne universités, Université de technologie de Compiègne, CNRS, Heudiasyc UMR 7253
CS 60 319, 60 203 Compiègne cedex, France
youcef.amarouche@hds.utc.fr

Rym N. Guibadj
LISIC, Laboratoire d'Informatique Signal et Image de la Côte d'Opale, ULCO, Université Lille Nord-de-France, France
rym.guibadj@univ-littoral.fr

Aziz Moukrim
Sorbonne universités, Université de technologie de Compiègne, CNRS, Heudiasyc UMR 7253
CS 60 319, 60 203 Compiègne cedex, France
aziz.moukrim@hds.utc.fr

Abstract

The Two-Echelon Vehicle Routing Problem (2E-VRP) is a variant of the classical vehicle routing problem arising in the context of city logistics. In the 2E-VRP, freight from a main depot is delivered to final customers using intermediate facilities, called satellites. In this paper, we propose a new hybrid heuristic method for solving the 2E-VRP that relies on two components. The first component effectively explores the search space in order to discover a set of interesting routes. The second recombines the discovered routes into high-quality solutions. Experimentations on benchmark instances show the performance of our approach: our algorithm achieves high-quality solutions in short computational times and improves the current best known solutions for several large scale instances.

2012 ACM Subject Classification Mathematics of computing → Optimization with randomized search heuristics, Applied computing → Transportation

Keywords and phrases Two-Echelon Vehicle Routing Problem, City Logistics, hybrid method, integer programming

Digital Object Identifier 10.4230/OASIcs.ATMOS.2018.11

Funding This work is supported by the French Environment and Energy Management Agency (ADEME), the Hauts-de-France region and the European Regional Development Fund (ERDF). It was carried out within the framework of the Labex MS2T, funded by the French Government (Reference ANR-11-IDEX-0004-02). It is also partially supported by TCDU project (Collaborative Transportation in Urban Distribution, ANR-14-CE22-0017).

[1] Agence de l'environnement et de la Maîtrise de l'Energie 20, avenue du Grésillé - BP 90406, 49004, Angers Cedex 01, France

1 Introduction

Freight transportation is a key factor underpinning economic growth. However, it is also a major nuisance, especially in urban areas where congestion and environmental effects disturb people's well-being. As demand for freight transportation increases, new transport policies and better traffic management become essential to limit its effects. The concept of city logistics is one approach to solving the problem. It aims to optimize freight transportation within city areas while considering traffic congestion and environmental issues as well as costs and benefits to the freight shippers [18]. Some of the most used models in city logistics are multi-echelon distribution systems, especially two-echelon systems.

In a two-echelon distribution system, delivery from one or more depots to the customers is managed by shipping and consolidating freight through intermediate depots called *satellites*. Freight is first moved from the depots to the satellites using large trucks. Then, freight is delivered from the satellites to the customers using smaller vehicles. Proceeding like this allows to shape more conveniently the fleet of vehicles to be used, as larger trucks are more cost efficient whereas smaller ones are preferable in city centers. Because the flow of freight in each echelon depends on that in the other echelon, routing problems arising in two-echelon distribution systems must be studied as a whole; they cannot be merely decomposed into two separate sub-problems. The problem that studies how to efficiently route freight in such systems is known as the *Two-Echelon Vehicle Routing Problem (2E-VRP)*.

In this paper, we consider the basic version of the 2E-VRP. It is characterized by a single depot and a set of satellites. A fleet of homogeneous vehicles of known size is available at each echelon. Vehicle capacities are limited. Only one type of product is to be shipped and split deliveries are only allowed at the first level. The objective is to minimize the total routing cost in both levels.

To address this problem, we propose a hybrid heuristic that relies on two components embedded in an iterative framework. The first component aims to generate a set of promising routes using destroy and repair operators combined with an efficient local search procedure. The second component recombines the generated routes by solving a set covering problem to obtain a high quality solution. Computational experiments conducted on the test instances of the literature show the performances of our approach, as it reached high quality solutions in short computing times, and was able to improve the current best known solution for several large instances.

The remainder of this paper is organized as follows. In Section 2, an overview of the related literature is given. The problem is described in Section 3, and the proposed approach is explained in Section 4. Section 5 presents computational results and compares them to the best known solutions of the literature. Finally, Section 6 concludes and discusses possible directions for future research.

2 Related work

The first definition of the Two-Echelon Vehicle Routing Problem (2E-VRP) was introduced by Perboli et al. [11, 12] who proposed a flow-based formulation, and solved the problem using a Branch-&-Cut algorithm (B&C). Since then, several exact methods have been developed to solve the 2E-VRP. Jepsen et al. [9] presented a different model for the 2E-VRP and solved it using a B&C that relies on a MILP relaxation of the problem, a feasibility test, and a specialized branching scheme to branch on infeasible solutions. Santos et al. [15] developed two Branch-&-Price (B&P) algorithms to solve the 2E-VRP. The first considers routes that satisfy elementary constraints while the second relaxes such conditions when pricing. Later,

Santos et al. [16] implemented a Branch-&-Cut-&-Price algorithm by incorporating valid inequalities into their B&P. Currently, the best exact method for the 2E-VRP was introduced in [1] and solves the problem by decomposing it into a set of Multi-Depot VRP with side constraints. It relies on a new integer linear programming (ILP) formulation, a bounding procedure based on dynamic programming, and a dual ascent method.

Various heuristics were also proposed to solve the 2E-VRP. Crainic et al. [3, 4] addressed the 2E-VRP by separating the first and the second echelon into two sub-problems, and then solving them sequentially. Components of these heuristics were later used in a hybrid GRASP with path re-linking in [5]. Hemmelmayr et al. [8] implemented an Adaptive Large Neighborhood Search (ALNS) that uses various repair and destroy operators specifically designed to solve the 2E-VRP. They also introduced new large instances on which they tested their algorithm. Zeng et al. [20] presented a hybrid two phase heuristic composed of a GRASP and Variable Neighborhood Descent (VND). Breunig et al. [2] developed a Large Neighborhood Search (LNS) for the 2E-VRP that was able to improve the best known solutions for several instances of the literature. More recently, Wang et al. [19] implemented a hybrid algorithm for the *2E-VRP with Environmental Considerations (2E-CVRP-E)*. Their algorithm comprises of a Variable Neighborhood Search (VNS) followed by a post optimization step based on the resolution of a linear program. Their algorithm further improves some best known solutions for 2E-VRP instances.

Related work may include other variants of the 2E-VRP (see [7, 17, 21]), and similar problems like the Two-Echelon Location Routing Problem (2E-LRP) and the Truck and Trailer Routing Problem (TTRP). For a more detailed survey on the subject, we invite the reader to refer to Cuda et al. [6].

3 Problem definition

The 2E-VRP is defined on a weighted undirected graph $G = (V, A)$, where V is the set of nodes and A the set of arcs. Set V is partitioned as $V = \{v_0\} \cup V_{sat} \cup V_{cust}$. Node v_0 represents the depot, subset V_{sat} contains n_{sat} satellites and subset V_{cust} contains n_{cust} customers. Set $A = A_1 \cup A_2$ is divided into two subsets. $A_1 = \{(i, j) : i, j \in \{v_0\} \cup V_{sat}, i \neq j\}$ contains the arcs that can be taken by first level vehicles: trips between the depot and the satellites and trips between pairs of satellites. $A_2 = \{(i, j) : i, j \in V_{sat} \cup V_{cust}, (i, j) \notin V_{sat} \times V_{sat}, i \neq j\}$ contains the arcs that can be taken by second level vehicles: trips between customers and satellites and trips between pairs of customers. A travel cost c_{ij}, $(i, j) \in A$, is associated with each arc. We assume that the matrix (c_{ij}) satisfies the triangle inequality.

Each customer $i \in V_{cust}$ demands d_i units of freight to be delivered. The demand of a customer cannot be split among several vehicles, that is, a customer must be served exactly once. Moreover, customer demands cannot be delivered by direct shipping from the depot and must be consolidated at a satellite. Satellite demands are not explicitly given but considered to be the sum of all the customer demands that are served trough the satellite. We assume that it can exceed vehicle capacity and thus, we allow for it to be split among different vehicles e.i. a satellite can be served by more than one vehicle. A satellite may also have a demand equal to zero and, in this case, not be visited by any vehicle. Consolidating shipments at satellite $s \in V_{sat}$ incurs handling costs equal to h_s times the quantity of handled goods.

A fleet f_1 of m_1 identical vehicles of capacity Q_1 is located at the depot v_0 and is used to deliver goods to the satellites. Additionally, a fleet f_2 of m_2 identical vehicles of capacity Q_2 is available for serving the customers. Each of the m_2 vehicles can be located at any satellite $s \in V_{sat}$ as long as the number of vehicles at one satellite does not exceed a limit k_s.

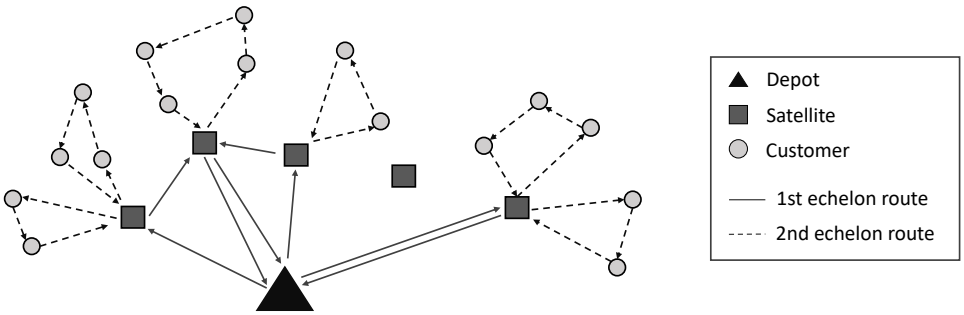

Figure 1 Example of a 2E-VRP solution.

We define a first-level route as a route performed by a first-level vehicle that starts at the depot, visits one or several satellites then returns to the depot. In a same way, we define a second-level route as a route run by a second-level vehicle that starts at satellite $s \in V_{sat}$, visits a subset of customers before returning to s. Routes must respect vehicle capacities, that is, the sum of deliveries made by a first-level route to the satellites it visits must not exceed Q_1 and the total demand of the customers visited by a second-level route must not exceed Q_2. Each vehicle performs only one tour, and each route has a cost equal to the sum of the costs of the arcs used.

The objective of the 2E-VRP is to find a set of routes at both levels such that each costumer is visited exactly once, the capacity constraints are respected, the quantity delivered to costumers from each satellite is equal to the quantity received from the depot, and the total routing and handling costs are minimized. Figure 1 shows a solution example for the 2E-VRP.

4 Solution method

We propose a hybrid heuristic that relies on a neighborhood search to generate good feasible solutions, and a integer programming (IP) method to recombine the routes from those solutions into a better one. Algorithm 1 summarizes the steps of our method.

At each iteration of the algorithm, the route generation heuristic takes an initial solution S and tries to improve it while exploring the solution space and storing new routes in the pool. After that, the recombination component uses the discovered routes to construct a better solution by solving a Set Cover based formulation of the 2E-VRP. If the recombination fails to produce a better solution, the algorithm constructs a different one from scratch and uses it as initial solution for the route generation heuristic during the next iteration. The idea of the approach is to use the integer program as a mean to find better quality solutions missed by the route generation heuristic while guiding the search process towards different regions of the solution space.

Exact models are usually used as post-optimization techniques after the heuristic resolution process as was done in [14, 19]. This is due to the exponential worst case performance of the model. What is new in our proposal is that we iteratively apply a Set Cover (SC) based formulation of the problem as a refinement technique rather than focusing on the local search results. We show that is possible to combine efficiently heuristic and exact algorithms to explore the search space within short runtimes.

Algorithm 1 Neighborhood Search and Set Cover hybrid heuristic for the 2E-VRP.

```
S_best :=  BestInsertionHeuristic();
S := S_best;
pool := {};
While (!Stopping criteria) Do
Begin
    S := RouteGenerationHeuristic(S,pool);
    If (cost(S) < cost(S_best)) Then
    S_best := S;

    S := RouteRecombination(pool);
    If (cost(S) < cost(S_best)) Then
        S_best := S;
    Else
        S := GreedyInsertionHeuristic();    /* Restart */
    End;
Return S_best;
```

4.1 Route generation heuristic

The neighborhood search we use to explore the solution space is based on the destroy-and-repair principle. At each iteration, a part of the solution is destroyed by removing a limited number of customers using a *destroy operator*. The removed customers are then re-inserted into the solution with a *repair operator*. The structure of this heuristic is described in Algorithm 2.

Starting from an initial solution, a random number $\eta \in [1, \tau]$ of customers is removed from the second echelon. The maximum number of customers to be removed τ is first initialized to τ_{min} and then increased after each non-improving iteration until it reaches τ_{max}. As soon as an improvement is found, τ is reset to τ_{min}. Slowly varying the value of τ during the execution allows to intensify the search around promising solutions and then to slowly increase diversification as the search converges toward a local optima. Once the solution is destroyed, the removed customers are reinserted using a *repair operator* and the obtained solution is passed to a *local search* to improve the second echelon routes. After that, the satellite demands are computed and the first echelon routes are constructed to obtain a complete solution. If the new solution has a better objective value than S, it is accepted as the new incumbent. Moreover, after i_{max} consecutive iterations without improving the incumbent solution, the best-known solution is updated and the configuration of the available satellites is modified using $perturb(S)$ to allow the search procedure to explore a different region of the solution space. The solution obtained after the perturbation becomes the new incumbent. The algorithm ends after $iter_{repeat}$ consecutive iterations have been performed without improving the best-found solution.

4.1.1 Destruction

The destroy procedure only considers the second level routes. At each iteration, it randomly chooses one of the following operators and removes a random number of customers η in $[1, \tau]$.

a. **Random removal operator:** removes η randomly chosen customers from the solution.
b. **Worst removal operator:** removes the customers with the highest increase in solution cost. More precisely, it calculates for each customer k located between i and j a saving

Algorithm 2 Route generation heuristic.

```
S := S_0;  /* Initial solution */
S_best := S;
tau := tau_min;  iter := 0;
Repeat
    i := 1;
    While(i < i_max) Do
    Begin
        S_tmp := destroy(S,tau);
        S_tmp := localSearch(repair(S_tmp));
        S_tmp := firstLevelReconstruction(S_tmp);
        pool := update(pool,S_tmp); /* Add routes to pool */
        If(cost(S_tmp) < cost(S)) Then
        Begin
            S:= S_tmp;
            i := 1;
            tau := tau_min;
        End;
        Else
        Begin
            i := i + 1;
            increment(tau);
        End;
    End;
    If(cost(S) < cost(S_best)) Then
    Begin
        S_best := S;
        iter := 0;
    End;
    Else iter := iter + 1;

    S := perturb(S);

    Until (iter >= iter_repeat);
Return  S_best;
```

value $c_{ik} + c_{kj} - c_{ij}$. Savings are then normalized by the average cost of the incident arcs of the corresponding customer and altered by a random factor between 0.8 and 1.2 as in [8]. Finally, customers are sorted in decreasing order of their normalized savings and the η first customers are removed from the solution. Normalizing the savings serves to avoid repeatedly removing the customers that are isolated from the others.

c. **Sequence removal operator:** removes a sequence of η consecutive customers from a randomly chosen route. If η is larger than the chosen route, the whole route is destroyed and the remaining number of customers is removed from a second route.

4.1.2 Repair and first level reconstruction

Repair is performed by using two heuristics : *Best Insertion Heuristic* (BIH) and *Greedy Insertion Heuristic* (GIH). When repairing an incomplete solution, we first use BIH. This constructive heuristic identifies among all the unrouted customers the one that increases the least the total solution cost and inserts it at its best position. It repeats the process until

all customers are routed. If one or more customers remain unrouted because their demands are higher than the largest remaining capacity of any vehicle, the repair process is restarted using GIH. The *Greedy Insertion Heuristic* inserts customers in a random order one after the other at their cheapest possible position in the solution. If the GIH fails, the customers are randomly reordered and the heuristic restarts. We observed that proceeding this way is sufficient to achieve feasible solutions after a small number of tries. These repair heuristics consider feasible insertions in already existing routes. If the maximum number of vehicles is not yet reached, the creation of new empty routes from open satellites is also tested.

The construction of the first-echelon routes is achieved by means of a heuristic similar to GIH. The heuristic starts by creating for each satellite with a demand greater than Q_1 enough back-and-forth trips so that its remaining demand becomes smaller than Q_1. Once it is done, the heuristic proceeds to insert of the remaining demands the same way as GIH.

4.1.3 Local search

The local search procedure consists of the following operators : $2-opt$, $2-opt^*$, $Relocate(\lambda)$, and $Swap(\lambda_1, \lambda_2)$ with $\lambda, \lambda_1, \lambda_2 \in \{1, 2\}$. The $2-opt$ operator [10] removes arcs $(i, i+1)$ and $(j, j+1)$ from the same route and reconnects arcs (i, j) and $(i+1, j+1)$. The $2-opt^*$ operator [13] is performed on each pair of routes u and v originating from the same satellite. It replaces arcs $(i, i+1)$ from u and $(j, j+1)$ from v by arcs $(i, j+1)$ and $(j, i+1)$ or by arcs (i, j) and $(i+1, j+1)$. *Relocate* moves sequences of λ customers to their best positions in the solution. Finally, *Swap* exchanges the positions of two sequences of λ_1 and λ_2 customers from the same route or from two different routes. At each iteration, the local search procedure randomly applies one of the above operators. If the chosen operator does not improve the solution, it is discarded, otherwise the set of operators is reset. The process continues until all operators have been discarded. Moves from each operator are performed in a first-improvement manner until no improving move can be found in the neighborhood.

4.1.4 Perturbation

In order to explore different regions of the search space, we temporarily close satellites and reopen them using the *Close Satellites* and *Open Satellites* operators.

a. **Close Satellites:** randomly chooses one satellite among the open ones having at least one route originating from them and closes it. The routes of the chosen satellite are reassigned to an open satellite that keeps their cost to a minimum. When the number of open satellites becomes less than the minimum required to serve all customers, the operator chooses a random satellite among the closed ones and opens it.
b. **Open Satellites:** chooses a random number of satellites among those that are closed and opens them. In order to allow the number of open satellites to decrease, especially at the beginning when most of them are open, the number of satellites to be opened can be nil.

4.2 Recombination method

The route recombination component uses a pool of routes collected during the search process and recombines them to obtain a high-quality solution by solving a set cover based formulation of the problem. In the following, we introduce the notations used in the IP model.

Let $\mathcal{M}$ be the set of all the possible first level routes, and $\mathcal{M}_s \subseteq \mathcal{M}$ the subset of first-level routes that serve satellite $s \in V_{sat}$. We note g_r the cost of route $r \in \mathcal{M}$. Let $\mathcal{R}$ be the set of

all the possible second-level routes, and $\mathcal{R}_s$ the subset or routes passing through $s \in V_{sat}$, thus $\mathcal{R} = \bigcup_{s \in V_{sat}} \mathcal{R}_s$. We associate to each route $r \in \mathcal{R}$ a cost c_r, and a load $w_r = \sum_{c \in r} d_c$ equal to the total demand of customers visited in route r. The binary parameter δ_{ri} is equal to 1 if and only if route $r \in \mathcal{R}$ visits customer $i \in V_{cust}$, and 0 otherwise. The second-level routes having been extracted from valid solutions, they all satisfy the vehicle capacity constraints.

Let $y_r \in \{0,1\}$ be a binary decision variable equal to 1 if and only if first-level route $r \in \mathcal{M}$ is in the solution, $x_r \in \{0,1\}$ a binary decision variable equal to 1 if and only if second-level route $r \in \mathcal{R}$ is in the solution, and q_{sr} a non-negative variable representing the amount of goods delivered by route $r \in \mathcal{M}$ to satellite $s \in V_{sat}$. We assume that $q_{sr} = 0$ if satellite s is not visited in route r. Parameter h_s represents handling costs at satellite $s \in V_{sat}$. The route recombination model can be formulated as follows:

$$\min \ z = \sum_{r \in \mathcal{R}} c_r \cdot x_r + \sum_{r \in \mathcal{M}} g_r \cdot y_r + \sum_{s \in V_{sat}} \sum_{r \in \mathcal{M}_s} h_s \cdot q_{sr} \tag{1}$$

$$\text{s.t.} \sum_{r \in \mathcal{R}} \delta_{ri} \cdot x_r \geq 1 \quad , \quad \forall i \in V_{cust} \tag{2}$$

$$\sum_{r \in \mathcal{R}_s} x_r \leq k_s \quad , \quad \forall s \in V_{sat} \tag{3}$$

$$\sum_{r \in \mathcal{R}} x_r \leq m_2 \tag{4}$$

$$\sum_{r \in \mathcal{M}} y_r \leq m_1 \tag{5}$$

$$\sum_{r \in \mathcal{M}_s} q_{sr} = \sum_{r \in \mathcal{R}_s} w_r \cdot x_r \quad , \quad \forall s \in V_{sat} \tag{6}$$

$$\sum_{s \in V_{sat}} q_{sr} \leq Q_1 \cdot y_r \quad , \quad \forall r \in \mathcal{M} \tag{7}$$

$$x_r \in \{0,1\}, \quad r \in \mathcal{R} \tag{8}$$

$$y_r \in \{0,1\}, \quad r \in \mathcal{M} \tag{9}$$

$$q_{sr} \in \mathbb{R}_+, \quad s \in V_{sat}, \ r \in \mathcal{M} \tag{10}$$

The objective function (1) states to minimize routing costs on both levels plus handling costs at each satellite. Constraints (2) ensure that each customer is visited at least once. Constraints (3) limit the number of second-level vehicles per satellite. Constraints (4) and (5) impose upper bounds on the number of vehicles used to implement first and second level routes. Balance between the quantity delivered by first-level routes to a satellite and the customer demands supplied from said satellite is imposed by constraints (6). Constraints (7) ensure that the capacity of first-level vehicles in not exceeded. Because the total amount of goods that need to be supplied to each satellite is not known beforehand, we cannot assume that capacity constraints are respected by first-level routes like we did for second-level routes. We need to explicitly state them in the formulation. Finally, constraints (8), (9), and (10) define the values domain for the decision variables.

Note that the model we use in our recombination component is a relaxation of the 2E-VRP. Constraints (2) require that each customer is visited at least once, instead of exactly once. However, since the distance matrix satisfies the triangle inequality, the two formulations remain equivalent as the resolution process will naturally lean towards solutions with the least possible amount of visits to a same customer. If the pool contains all the possible routes, solving the formulation with the relaxed model will still result in an optimal solution

where each customer is visited exactly once. The idea of relaxing the problem stems from the fact that the recombination pool only contains a limited subset of routes, thus the solutions it finds may be few. To increase the number of combinations that can be made, we choose to allow combining routes that share common customers, as it can lead to better objective values. Even though the resulting combination may not be a valid solution to the 2E-VRP, removing the extra visits to each customer makes it feasible while producing new routes and further lowering the objective value.

4.2.1 Pool management and initialization

The performance of the route recombination component strongly depends on the size of the pool of routes. A larger size increases the chances of finding high-quality solutions but also induces higher computation times, whereas a small size reduces computation times but makes finding improved solutions less likely. Thus, pool size must be fixed in order to offer a good trade-off between solution quality and computation efforts. Furthermore, to account for the lesser number of available routes, it is better to keep inside the pool only routes that are more likely to be in high-quality solutions. To this end, we assign each route a priority based on the cost of the solution it was extracted from, thus favoring routes that belong to the best found solutions. When the pool capacity is reached, routes with lower priority are removed and replaced by the new ones. If a route already exists inside the pool, its priority is updated if it is extracted from a better solution.

The pool is initialized with the routes of x different solutions generated by the *Greedy Insertion Heuristic* described in Section 4.1.2 and improved with the local search procedure described in Section 4.1.3. Furthermore, for each satellite s we add m^1 copies of round trip routes to s from the depot to account for the possibility of it being served more than once.

4.2.2 Correcting heuristic

When the route recombination model is solved, some customers might be visited more than once. In this case, we use a correcting heuristic to remove the extra visits and produce a valid solution. The algorithm starts by establishing the set V_{cm} of customers that are visited more than once. It then computes for each visit v of each customer $i \in V_{cm}$ its removal gain δ_{iv}, removes the visit with the highest gain and updates the gains for the remaining ones. When the number of visits to a customer drops to one, it is removed from V_{cm}. The procedure is repeated until V_{cm} becomes empty. During our tests, we observed that only a few customers tend to be visited multiple times. Thus, this simple heuristic proves to be enough to provide good results with limited computational effort.

5 Computational results

Our algorithm was coded in C++ using the Standard Template Library (STL) for data structures, and IBM ILOG CPLEX 12.6.3 to solve the IP. The algorithm is compiled with the GNU GCC compiler in a Linux environment and tested on an Intel Xeon E5-2670v2 CPU at 2.50GHz with similar performance to the ones used in the literature.

We conducted extensive computational experiments on the benchmark instances for the 2E-VRP. There are currently six instance sets available. The size of the instances ranges from 12 customers and 2 satellites, to 200 customers and 10 satellites. The main characteristics of the benchmark instances are listed in Table 5 of Appendix A. Note that the small instances of *Set 1* are no longer used for testing, thus they are not included. For our tests we used the files provided by Breunig et al. [2].

Table 1 Parameter settings.

Parameter	Description	Value
i_{max}	max. nb. of non-improving iterations before perturbing the solution	$0.2n$
$iter_{repeat}$	max. nb. of non-improving iterations for the route generation	10
τ_{min}, τ_{max}	max. nb. of customers to be removed	$0.15n, 0.45n$
S_{pool}	size of the pool	$\frac{\sum d_c}{m^2} * 15$
N_{algo}	max. nb. of non-improving iterations in the global algorithm	n

5.1 Parameter tuning

The proposed approach has six parameters: (1) i_{max}, $iter_{repeat}$, τ_{min} and τ_{max} in the route generation heuristic; (2) the size of the pool (S_{pool}) in the route recombination component; and (3) the stopping criterion of the iterative framework. We carried out a series of preliminary experiments to set the parameter values: we tested our algorithm on a subset of instances while varying parameter values, and kept those that offered the best trade-off between solution quality and runtime. The stopping criteria is set according to previous literature. Breunig et al. [2] set the maximum runtime of their algorithm to 60s for small instances and 900s for larger ones. Wang et al. [19] use the maximum runtime and the maximum number of iterations N_{algo} without improving the best found solution as stopping rules. They set them so that the maximum runtime of their algorithm does not exceed 1500s. To show the performance of our method we restrict our runtime to 60s and 900s as do Breunig et al. [2]. The remaining parameter settings are given in Table 1.

5.2 Comparison with the literature

In order to investigate the effectiveness of the proposed algorithm, we compare its performance, when applicable, with that of the ALNS by Hemmelmayr et al. [8], the LNS by Breunig et al. [2] and the VNS by Wang et al. [19] as well as the current best-known solution for each instance from the literature. All the results were obtained through five independent runs of the algorithm and are summarized in Table 2. The results of our Neighborhood Search and Set Cover Hybrid Heuristic are listed in column "NS-SC". The columns "ALNS", "LNS", and "VNS" show the results of the methods proposed by Hemmelmayr et al. [8], Breunig et al. [2], and Wang et al. [19], respectively. The average and the best objective value of the five runs are given in columns "Avg. 5" and "Best 5", respectively. Column "CPU" shows the average runtime of the algorithm in seconds. The column "BKS" refers to the best-known solution of that set of instances. As was observed by Breunig et al. [2], there exist some small differences in objective values that can be explained by a different rounding convention or the small optimality gap of CPLEX. Table 3 summarizes the gaps obtained by each algorithm on each benchmark. Columns "Avg. %" and "Best %" show the average and best gap, respectively, expressed as a percentage. The overall gap is calculated by considering the number of instances in each benchmark.

Instances in Sets 2 and 3, are relatively easy to solve and all algorithms are able to find the best known solutions at least one time out of five. Instances in Set 4, while not bigger than some instances of Sets 2 and 3, are more difficult to solve due to customer distribution [2]. Our "NS-SC" only misses four of the current best known solutions, and still achieves high quality solutions with gaps less than 0.04%. Instances of Set 5 are the largest of the literature and those where the gaps and the runtimes are more important. On these instances, all the algorithms fail to achieve the best known solutions for several instances,

Table 2 Computational results for 2E-VRP instances.

Instances		$\|V_{cust}\|$	ALNS			LNS			VNS			NS-SC			BKS
			Avg 5	Best 5	CPU	Avg 5	Best 5	CPU	Avg 5	Best 5	CPU	Avg	Best 5	CPU	
Set 2	a	21	410.69	410.69	35	410.69	410.69	60	410.69	410.69	1	410.69	410.69	1	410.69
		32	744.49	744.49	69	744.49	744.49	60	744.49	744.49	4	744.49	744.49	6	744.49
	b	50	559.20	559.20	147	559.20	559.20	60	559.20	559.20	18	559.20	559.20	22	559.20
		50	530.10	530.10	150	530.10	530.10	60	530.10	530.10	13	530.10	530.10	19	530.10
	c	50				628.65	628.65	60	628.65	628.65	42	628.65	628.65	24	628.65
		50				566.52	566.52	60	566.52	566.52	25	566.52	566.52	22	566.52
Set 3	a	21	512.61	512.61	40	512.61	512.61	60	512.61	512.61	1	512.61	512.61	1	512.61
		32	669.99	669.99	78	669.99	669.99	60	669.99	669.99	6	669.99	669.99	5	669.99
	b	50	714.75	714.75	161	714.75	714.75	60	714.75	714.75	37	714.75	714.75	28	714.75
	c	50				668.40	668.40	60	668.93	668.40	70	668.40	668.40	27	668.40
Set 4	a	50				1527.13	1526.86	60	1527.80	1527.55	38	1527.23	1526.87	25	1526.86
		50				1415.66	1415.66	60	1416.69	1415.85	67	1416.12	1415.70	23	1415.65
		50				1317.36	1317.34	60	1318.34	1317.48	85	1318.16	1317.34	31	1317.33
	b	50	1524.85	1524.70	248	1516.01	1515.61	60	1516.70	1516.27	36	1516.18	1515.61	33	1515.60
		50	1378.16	1377.85	139	1375.68	1375.68	60	1376.37	1375.79	36	1375.82	1375.68	29	1375.67
		50	1301.17	1300.33	121	1299.93	1299.88	60	1300.23	1299.96	63	1300.28	1299.91	37	1299.87
Set 5	5_1	100	1067.11	1058.27	373	1058.94	1057.58	900	1059.37	1058.26	953	1059.32	1056.75	582	1056.74
	5_2	100	960.87	950.01	421	961.41	956.64	900	951.70	949.47	1452	955.21	951.45	557	946.82
	5_3	200	1386.45	1357.14	974	1375.70	1357.79	900	1343.60	1340.87	1500	1355.42	1348.62	900	1338.35
Set 6	A_50	50				639.64	639.64	60	640.29	640.06	155	639.89	639.64	35	639.64
	A_75	75				947.85	946.35	900	946.54	946.32	730	946.86	946.86	235	946.32
	A_100	100				1150.05	1147.34	900	1147.05	1146.18	1216	1147.31	1146.32	560	1146.18
	B_50	50				826.48	826.48	60	826.59	826.48	141	826.59	826.59	33	826.48
	B_75	75				1461.94	1461.54	900	1461.33	1461.29	578	1461.53	1461.29	363	1461.29
	B_100	100				1741.67	1738.57	900	1736.80	1734.02	1322	1736.53	1733.92	712	1733.36
Avg.			904.65	900.78	227	984.83	983.53	295	983.01	982.45	344	983.51	982.72	172	982.06

Table 3 Summary of average and best gaps on 2E-VRP benchmarks.

	ALNS		LNS		VNS		NS-SC	
	Avg. %	Best %	Avg. %	Best %	Avg. %	Best %	Avg. %	Best %
Set 2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Set 3	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00
Set 4a			0.01	0.00	0.07	0.02	0.04	0.00
Set 4b	0.30	0.26	0.01	0.00	0.05	0.02	0.03	0.00
Set 5	2.00	0.63	1.51	0.86	0.39	0.20	0.80	0.42
Set 6 A			0.16	0.04	0.06	0.02	0.07	0.02
Set 6 B			0.17	0.11	0.07	0.01	0.11	0.03
Overall	1.27	1.20	0.18	0.09	0.08	0.03	0.10	0.04

Table 4 New best known solution values found by NS-SC.

	Instance	$\|V_{cust}\|$	$\|V_{sat}\|$	m_1	m_2	Former best known	New best known
Set 5	100-5-1b	100	5	5	15	1108.62	1103.55
	100-10-1b	100	10	5	18	916.25	911.8
	100-10-3b	100	10	5	17	850.92	849.73
	200-10-1	200	10	5	62	1539.29	1538.35
	200-10-1b	200	10	5	30	1186.78	1175.81
	200-10-3	200	10	5	63	1780.67	1779.68
	200-10-3b	200	10	5	30	1197.9	1196.93

mainly due to the bigger numbers of customers and satellites that constitute the instances. The ALNS and the LNS can achieve an average relative gap of 2.00% and 1.51%, respectively. The VNS achieves an average relative gap of 0.39%, but is slower than the other algorithms. Our algorithm, on the other hand, offers good compromise between solutions quality and runtime, as it achieves an average relative gap of 0.80% while being significantly faster than both the LNS and the VNS. It was also able to improve the current best known solutions for a total of seven instances from Set 5 during our experiments. Only Breunig et al. [2] and Wang et al. [19] report results on the instances of Set 6. The LNS, the VNS, and our NS-SC are all able to obtain very low average relative gaps on both Set 6a and Set 6b, but once again our algorithm has a smaller runtime.

Overall, our algorithm is able to achieve the current best known solutions for 216 out of 234 instances with an overall average relative gap of 0.10% and running times smaller than those of the literature. During our experiments, NS-SC found seven new best known solution values for the instances of Hemmelmayr et al. [8]. The values of these newly found solutions are reported in Table 4. The Set designation and the names of the instances are displayed in the first two columns. Column $|V_{cust}|$ represents the number of customers in the instance, $|V_{sat}|$ is the number of satellites, and m_1 and m_2 indicate the number of available first-level and second-level vehicles, respectively. Based on the above results, our approach is very effective in solving the 2E-VRP.

6 Conclusions

In this paper, we presented a hybrid heuristic for the 2E-VRP. The algorithm uses an effective neighborhood search to explore the solution space and discover high quality solutions. By keeping trace of the exploration steps, the heuristic generates a set of routes which are then

recombined using an integer programming model. Solving this model serves as way to find better solutions that were missed by the neighborhood search procedure and to faster lead the algorithm towards promising regions of the solution space. Computational experiments on the standard benchmark instances demonstrate the competitiveness of our approach. Our algorithm consistently achieves high quality solutions with an overall average relative gap of 0.10%, while requiring less running time than other algorithms, and improves the current best known solutions for seven instances for which no optimal solution is known.

In summary, the results presented in this paper are encouraging for the application of our approach to optimize other two-echelon routing problems. Its components can be adapted and additional ones can be integrated to account for different constraints. Future work will primarily focus on the extension of the algorithm to variants of the 2E-VRP and similar routing problems, mainly to accommodate more practical constraints and more realistic cost structures.

References

1. Roberto Baldacci, Aristide Mingozzi, Roberto Roberti, and Roberto Wolfler Calvo. An exact algorithm for the two-echelon capacitated vehicle routing problem. *Operations research*, 61(2):298–314, 2013.
2. U. Breunig, V. Schmid, R. F. Hartl, and T. Vidal. A large neighbourhood based heuristic for two-echelon routing problems. *Computers and Operations Research*, 76:208–225, 2016.
3. Teodor Gabriel Crainic, Simona Mancini, Guido Perboli, and Roberto Tadei. Clustering-based heuristics for the two-echelon vehicle routing problem. *CIRRELT-2008-46*, 2008.
4. Teodor Gabriel Crainic, Simona Mancini, Guido Perboli, and Roberto Tadei. Multi-start heuristics for the two-echelon vehicle routing problem. In Peter Merz and Jin-Kao Hao, editors, *Evolutionary Computation in Combinatorial Optimization*, pages 179–190, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
5. Teodor Gabriel Crainic, Simona Mancini, Guido Perboli, and Roberto Tadei. *GRASP with Path Relinking for the Two-Echelon Vehicle Routing Problem*, pages 113–125. Springer New York, New York, NY, 2013.
6. R Cuda, G Guastaroba, and M G Speranza. A survey on two-echelon routing problems. *Computers & Operations Research*, 55:185–199, 2015.
7. Philippe Grangier, Michel Gendreau, Fabien Lehuédé, and Louis-Martin Rousseau. 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, 2016.
8. Vera C Hemmelmayr, Jean-François Cordeau, and Teodor Gabriel Crainic. An adaptive large neighborhood search heuristic for Two-Echelon Vehicle Routing Problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228, 2012.
9. Mads Jepsen, Simon Spoorendonk, and Stefan Ropke. A branch-and-cut algorithm for the symmetric two-echelon capacitated vehicle routing problem. *Transportation Science*, 47(1):23–37, 2013.
10. S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.
11. Guido Perboli and Roberto Tadei. New families of valid inequalities for the two-echelon vehicle routing problem. *Electronic notes in discrete mathematics*, 36:639–646, 2010.
12. Guido Perboli, Roberto Tadei, and Daniele Vigo. The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transportation Science*, 45(3):364–380, 2011.

13 Jean-Yves Potvin and Jean-Marc Rousseau. An exchange heuristic for routing problems with time windows. *Journal of the Operational Research Society*, 46(12):1433–1446, 1995.

14 Yves Rochat and Éric D Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1):147–167, 1995.

15 Fernando Afonso Santos, Alexandre Salles da Cunha, and Geraldo Robson Mateus. Branch-and-price algorithms for the two-echelon capacitated vehicle routing problem. *Optimization Letters*, 7(7):1537–1547, 2013.

16 Fernando Afonso Santos, Geraldo Robson Mateus, and Alexandre Salles da Cunha. A branch-and-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. *Transportation Science*, 49(2):355–368, 2014.

17 Mehmet Soysal, Jacqueline M. Bloemhof-Ruwaard, and Tolga Bektaş. The time-dependent two-echelon capacitated vehicle routing problem with environmental considerations. *International Journal of Production Economics*, 164:366–378, jun 2015.

18 Eiichi Taniguchi, Russell G. Thompson, Tadashi Yamada, and J.H.R. van Duin. *City logistics –Network modelling and intelligent transport systems*. Pergamon, Oxford, elsevier edition, 2001.

19 Kangzhou Wang, Yeming Shao, and Weihua Zhou. Matheuristic for a two-echelon capacitated vehicle routing problem with environmental considerations in city logistics service. *Transportation Research Part D*, 57(October):262–276, 2017.

20 Zheng Yang Zeng, Wei Sheng Xu, Zhi Yu Xu, and Wei Hui Shao. A Hybrid GRASP+VND heuristic for the two-echelon vehicle routing problem arising in city logistics. *Mathematical Problems in Engineering*, 2014, 2014.

21 Lin Zhou, Roberto Baldacci, Daniele Vigo, and Xu Wang. 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, 2018.

A Benchmark instances for the 2E-VRP

There are six sets of benchmark instances for the Two-Echelon Vehicle Routing Problem (2E-VRP). The size of the instances ranges from 12 customers and 2 satellites, to 200 customers and 10 satellites. For our tests, we used the instances provided by [2]. Table 5 displays the main characteristics of the different sets. Column *Nb.* represents the number of instances of the set, $|V_{cust}|$ is the number of customers, $|V_{sat}|$ is the number of satellites, m_1 and m_2 the number of available first-level and second-level vehicles, respectively, and k_s represents the maximum number of second-level routes per satellite. Note that the small instances of *set 1* are no longer used for testing, thus they are not included in the table.

Table 5 Characteristics of the benchmark instances for the 2E-VRP.

Set	Subset	Nb.	$\|V_{cust}\|$	$\|V_{sat}\|$	m_1	m_2	k_s
Set 2	a	6	21	2	3	4	-
		6	32	2	3	4	-
	b	6	50	2	3	5	-
		3	50	4	4	5	-
	c	6	50	2	3	5	-
		3	50	4	4	5	-
Set 3	a	6	21	2	3	4	-
		6	32	2	3	4	-
	b	6	50	2	3	5	-
	c	6	50	2	3	5	-
Set 4	a	18	50	2	3	6	4
		18	50	3	3	6	3
		18	50	5	3	6	2
	b	18	50	2	3	6	-
		18	50	3	3	6	-
		18	50	5	3	6	-
Set 5		6	100	5	5	$[15, 32]$	-
	-	6	100	10	5	$[17, 35]$	-
		6	200	10	5	$[30, 63]$	-
Set 6	a	9	50	$[4, 6]$	2	50	-
		9	75	$[4, 6]$	3	75	-
		9	100	$[4, 6]$	4	100	-
	b	9	50	$[4, 6]$	2	50	-
		9	75	$[4, 6]$	3	75	-
		9	100	$[4, 6]$	4	100	-