INFORMS JOURNAL ON COMPUTING
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0899-1499 | EISSN 1526-5528 | 00 | 0000 | 0001

INFORMS
DOI 10.1287/xxxx.0000.0000

AILS-II: An Adaptive Iterated Local Search Heuristic for the Large-scale Capacitated Vehicle Routing Problem

Vinícius R. Máximo
Instituto de Ciência e Tecnologia, Universidade Federal de São Paulo (UNIFESP)
Av. Cesare M. G. Lattes, 1201, Eugênio de Mello, São José dos Campos-SP, CEP: 12247-014, Brazil, vinymax10@gmail.com,

Jean-François Cordeau
HEC Montréal and GERAD, 3000 chemin de la Côte-Sainte-Catherine, Montréal, H3T 2A7, Canada, jean-francois.cordeau@hec.ca,

Mariá C. V. Nascimento
Divisão de Ciência da Computação (IEC), Instituto Tecnológico de Aeronáutica (ITA), Praça Marechal Eduardo Gomes, 50, Vila das Acácias, São José dos Campos-SP, CEP 12228-900, Brazil, mariah@ita.br,

A recent study on the classical Capacitated Vehicle Routing Problem (CVRP) introduced an adaptive version of the widely used Iterated Local Search (ILS) paradigm, hybridized with a path-relinking (PR) strategy. The solution method, called AILS-PR, outperformed existing meta-heuristics for the CVRP on benchmark instances. However, tests on large-scale instances suggest that PR is too slow, making AILS-PR less advantageous in this case. To overcome this challenge, this paper presents an Adaptive Iterated Local Search (AILS) combined with mechanisms to handle large CVRP instances, called AILS-II. The computational cost of this implementation is reduced while the algorithm also searches the solution space more efficiently. AILS-II is very competitive on smaller instances, outperforming the other methods from the literature with respect to the average gap to the best known solutions. Moreover, AILS-II consistently outperforms the state of the art on larger instances with up to 30,000 vertices.

Key words: Combinatorial Optimization, Capacitated Vehicle Routing Problem (CVRP), Adaptive Iterated Local Search (AILS), Learning in Metaheuristics

1. Introduction

The Capacitated Vehicle Routing Problem (CVRP) is a widely studied combinatorial optimization problem first introduced by Dantzig and Ramser (1959). It consists in finding a set of routes that minimizes the cost of making deliveries to a set of customers with a homogeneous vehicle fleet based at a central depot. There is a great interest in providing efficient solutions to this problem, given its important role in many supply chains. Toth and Vigo (2001) carried out a study that reported potential savings of around 5 to 20% in total transportation costs with the use of computer systems in real routing problems. Most of these problems are difficult to solve in reasonable time for large instances with hundreds or thousands of customers. In these cases, heuristic methods are often preferred as they provide high-quality feasible solutions in reasonable computing time.

A number of successful heuristic methods have been employed to tackle the CVRP, among which we highlight the recently proposed Hybrid Genetic Search (HGS) (Vidal 2022), Slack Induction by String Removals (SISRs) (Christiaens and Vanden Berghe 2020), Fast ILS Localized Optimization (FILO) (Accorsi and Vigo 2021) and Adaptive Iterated Local Search with Path-Relinking (AILS-PR) (Máximo and Nascimento 2021). In particular, AILS-PR outperformed the state-of-the-art metaheuristics when considering small- and medium-sized instances, with up to 1000 vertices, proposed by Uchoa et al. (2017), Christofides et al. (1979) and Golden et al. (1998). However, recent experiments indicate that AILS-PR is too slow on large-scale instances. More specifically, the path-relinking (PR) strategy shows poor performance in these cases. For large-scale CVRP instances, we highlight that FILO has achieved outstanding results.

This paper introduces a new version of AILS, called AILS-II, specially designed to efficiently solve large-scale CVRP instances. Several components were modified from the previous version of AILS to allow the processing of a large amount of data. Computational experiments with small- and medium-size instances show that AILS-II is very competitive with FILO and HGS. In addition, the experiments on large-scale instances with more than 336 customers indicate that AILS-II is consistently better than the other algorithms.

The rest of the paper is organized as follows. Section 2 presents the description of the CVRP and the main notations used throughout the paper. Section 3 introduces AILS-II and presents different approaches for controlling the perturbation degree and acceptance criteria. Section 4 shows a summary of the code organization and presents all parameters required to run the algorithm. Section 5 presents an example of how to solve a CVRP instance. Section 6 reports the results of computational experiments comparing AILS-II with other algorithms from the literature. Finally, in Section 7, we draw some conclusions and suggest future research directions.

2. Problem description

The Capacitated Vehicle Routing Problem (CVRP) can be defined on an undirected graph $G = (V, E)$, where $V = \{0, 1, \ldots, n\}$ is the set of $n + 1$ vertices and E the set of edges. The depot is represented by the vertex 0 and the other vertices in the subset $V_c = V \setminus \{0\}$ represent the n customers. Each edge $\{i, j\} \in E$ has a non-negative weight d_{ij} that represents the cost associated with a vehicle moving between vertices i and j.

Each customer $i \in V_c$ has a non-negative demand q_i that must be satisfied (the depot has demand $q_0 = 0$). To meet the demands of the customers, m identical vehicles with a capacity of $\bar{q}$ are used. To ensure the feasibility of the problem, the demand of each customer is assumed to be smaller than or equal to the vehicle capacity, that is, $q_i \leq \bar{q}\ \forall i \in V_c$. In the CVRP, each route takes the form of a closed loop with no node repetition. A closed loop is represented by a cyclic sequence of vertices where a pair of vertices is adjacent if they are consecutive in the sequence and not adjacent otherwise. The CVRP, therefore, consists in finding a set of m routes that minimize the sum of the edge weights. The routes of a solution s are described by $\mathscr{R} = \{R_1^s, R_2^s, \ldots, R_{m_s}^s\}$, where m_s represents the number of routes in the solution s, $R_i^s = \{v_0^i, v_1^i, \ldots, v_{t_i}^i\}$, $t_i + 1$ is the length of route R_i^s, $v_0^i = v_{t_i}^i = 0$, $R_i^s \cap R_j^s = \{0\}$, for $i \neq j$, and $\cup_{i=1}^{m_s} R_i^s = V$.

The CVRP is $\mathbb{NP}$-Hard (Lenstra and Rinnooy Kan 1981) since it generalizes the Traveling Salesman Problem (TSP), which seeks to minimize the length of a Hamiltonian tour.

3. Overview of Adaptive Iterated Local Search (AILS)

AILS was initially proposed by Máximo and Nascimento (2021) to solve the CVRP, and has recently been adapted for the Heterogeneous Fleet Vehicle Routing Problem (HVRP) by Máximo et al. (2022). AILS is an adaptive metaheuristic that has two main steps: perturbation and local search. These two steps are performed iteratively until a stopping criterion is reached. At each iteration, a reference solution is perturbed to generate a potentially different solution. A local search step is then applied to improve the quality of the resulting solution through an exploration of the neighborhood formed by vertex and edge movements. Then, this solution is evaluated by the acceptance criterion. If it is accepted, it becomes the new reference solution. The solution obtained after the perturbation step may be infeasible. For this reason, we apply an algorithm that uses the same neighborhood as the local search but will guarantee the feasibility of the solution. The main difference between the local search and feasibility strategies is that the movements in the latter are chosen by prioritizing the reduction of capacity constraint violations. The adaptive behavior of AILS can be observed in the definition of the perturbation degree and in the acceptance criterion, described in more detail in Sections 3.1 and 3.2, respectively. These two conditions are responsible for the diversity control of the method, which means that controlling them is of utmost importance.

3.1. Perturbation Degree

The perturbation degree establishes the number of changes applied to the reference solution s^r to obtain a different solution. The greater the perturbation degree, the larger the distance between these solutions. Thus, the control of the perturbation degree enables the method to manage the diversity of the search algorithm. For this reason, the perturbation degree control can be seen as a mechanism of great relevance for metaheuristics, considering that adequate control can allow the algorithm to escape from local optima. High diversity means that the algorithm will be able to escape from a local optimum more easily, but it might be costly to find the best solution in a given neighborhood (Lourenço et al. 2010). On the other hand, low diversity yields a higher chance that the algorithm will get stuck in a local optimum. To achieve an adequate balance between these two goals, the proposed AILS considers a mechanism similar to the one presented by Máximo and Nascimento (2021). However, we introduce a convergent behavior to control the degree of perturbation. The mechanism proposed in Máximo and Nascimento (2021) uses a fixed parameter, called d_β, that establishes the ideal distance between the solution s obtained by the local search and the reference solution s^r. In the convergent mechanism used here, the value of d_β starts with a value of d_{max} and decreases throughout the execution of the algorithm until it reaches a value of d_{min}.

At each iteration of the algorithm, the value of d_β is adjusted as $d_\beta \leftarrow d_\beta \left(\frac{d_{min}}{d_{max}}\right)^{\frac{1}{it_{max}}}$, where it_{max} is the estimated maximum number of iterations performed by the algorithm.

3.2. Acceptance Criterion

The acceptance criterion establishes whether the current solution should become the reference solution and be used in the following iterations of the algorithm. The AILS introduced in this paper uses a convergent acceptance criterion with a more relaxed criterion at the beginning of the search and a more restrictive condition as it approaches the end of the search. This criterion was inspired by the Threshold Acceptance (TA) algorithm proposed by Dueck and Scheuer (1990). In line with this, the employed acceptance criterion restricts the quality of the accepted solutions to the threshold $\bar{b} = \underline{f} + \eta(\bar{f} - \underline{f})$. The value of η establishes a percentage between the average quality of the solutions obtained by the local search, called $\bar{f}$, and the best solution found in the last γ iterations, referred to as $\underline{f}$. This threshold was proposed by Máximo and Nascimento (2021) and the η value was adjusted according to the flow of accepted solutions. In the proposed AILS, we propose a convergent variation for the value of η. Therefore, η starts at 1 and decreases until a minimum value $\epsilon = 0.01$. Thus, at each iteration of the algorithm $\eta \leftarrow \eta\epsilon^{\frac{1}{it_{max}}}$.

3.3. Other differences between AILS and AILS-II

Besides the convergent criteria in the diversity control mechanisms employed in AILS-II, this version of AILS presents substantial changes from its previous versions. These were necessary to achieve a good performance on large instances. We enumerate next the key differences between AILS and AILS-II:

- **Reduced neighborhood in SWAP movement:** Besides replacing SWAP by SWAP* (Vidal 2022) in AILS-II, we restricted the neighborhood in the SWAP* by considering for each vertex v_i its φ closest vertices and not the entire set of vertices that belong to its route. The motivation is that the larger the route size, the greater the computational cost. For large instances, the computational cost becomes prohibitive.

- **Feasibility algorithm:** AILS-PR uses two different criteria to assign values to rank the moves depending on whether the performed move will provide a better or worse quality solution. AILS-II considers the same criterion for both cases and ranks the moves according to the ratio of the difference in cost between the original solution and the solution after the move and the feasibility gain. The feasibility gain is the value of the infeasibility reduction considering the violated constraints.

- **Different perturbation heuristics:** The perturbation heuristics of AILS are composed of a set of addition and removal heuristics. Removal heuristics remove vertices from a given solution whereas addition heuristics add them back in positions that depend on the criterion of the method. Both AILS-PR and AILS-II follow the batch approach to remove and add them back to the solutions. This means that they first remove ω vertices from the solution through a removal heuristic and, after that, the addition heuristic inserts the vertices back in the solution. The AILS for the HVRP follows an alternate call between removal and addition heuristics, where one of the ω vertices is removed and subsequently added to a position of the solution. The process repeats ω times, to allow all ω vertices to change position. On the one hand, AILS does allow the vertices to return to the same position. On the other hand, with the addition heuristics in AILS-PR and AILS-II, vertices can be inserted in the same position. Regarding the employed removal heuristics, all of them use the concentric and sequence removal strategies. However, AILS-PR considers two other removal heuristics, whereas AILS for the HVRP also employs the random removal. AILS-II considers two addition heuristics. The only addition heuristic common to AILS for the HVRP, AILS-PR and AILS-II is the one called insertion by cost. Besides, AILS-II uses the insertion by distance, also considered by AILS for the HVRP. Table 1 presents the main heuristics used in the perturbation step of the three different AILS versions as well as the employed approaches.

The AILS-II does not employ the so-called path-relinking (PR), present in AILS-PR (Máximo and Nascimento 2021). The reason for it is that the computational cost required by the hybridized version on large-scale CVRP instances is too high.

Table 1 Main characteristics of the perturbation step in the AILS-PR (Máximo and Nascimento 2021), AILS (Máximo et al. 2022) and AILS-II.

	Algorithm		
Perturbation approach/method	AILS-PR (CVRP)	AILS (HVRP)	AILS-II (This paper)
Batch approach	✓		✓
Alternate approach		✓	
Concentric removal	✓	✓	✓
Random removal		✓	
Sequence removal	✓	✓	✓
Proximity removal	✓		
Insertion by proximity	✓		
Insertion by cost	✓	✓	✓
Insertion by distance		✓	✓

4. Code Organization

AILS-II was developed in Java and the code is available at `https://github.com/vinymax10/AILS-CVRP`. The code is divided into 8 packages containing 35 files and roughly 6000 lines. The main class is called AILS-II, which is inside the `SearchMethod` package. The main method is presented in Source Code 1. The user parameter values are input provided by the user. An instance of the problem is constructed and used to create the `ailsII` object from the `AILSII` class. This class receives in the constructor an instance of the problem and the configuration of the algorithm. The beginning of the search occurs with the call to the search method of the `AILSII` class.

```java
public static void main(String[] args)
{
    InputParameters reader=new InputParameters();
    reader.readingInput(args);
    Instance instance=new Instance(reader);
    AILSII ailsII=new AILSII(instance, reader);
    ailsII.search();
}
```

A summary of the contents of each package is presented next.

- `Auxiliary`: This package contains the method that calculates the distance between two solutions and the dynamic average used in the diversity control.

- `Evaluators`: This package contains the cost and feasibility evaluators of the movements, and the execution process of the movements used by the local search and feasibility.
- `DiversityControl`: This package contains the source codes of the acceptance criterion and the control of the perturbation degree.
- `Data`: This package contains codes for reading instance data.
- `Improvement`: This package contains the implementation of the local search and feasibility methods.
- `SearchMethod`: This package contains the implementation of AILS-II.
- `Perturbation`: This package contains the addition and removal heuristics used in the perturbation step.
- `Solution`: This package contains the data structures of the solution, routes and graph vertices.

5. Example of Usage

To run the AILS class it is necessary to define the following parameters:

- `-file`: the file name of the problem instance.
- `-rounded`: A flag that indicates whether the instance has rounded distances or not. The options are: [false, true]. The default value is true.
- `-stoppingCriterion`: It is possible to use two different stopping criteria:
  — `Time`: The algorithm stops when a given time in seconds has elapsed.
  — `Iteration`: The algorithm stops when the given number of iterations has been reached.
- `-limit`: Refers to the value that will be used in the stopping criterion. If the stopping criterion is a time limit, this parameter is the timeout in seconds. Otherwise, this parameter indicates the number of iterations. The default value is the maximum limit for a double precision number in the JAVA language.
- `-best`: Indicates the value of the optimal solution. The default value is 0.
- `-varphi`: Parameter of the feasibility and local search methods that refers to the maximum cardinality of $\delta(v)$ – nearest neighbors of v. The default value is 40. The larger it is, the greater the number of movements under consideration in the methods.
- `-gamma`: Number of iterations for AILS-II to perform a new adjustment of variable ω. The default value is 20.
- `-dMax`: Initial reference distance between the reference solution and the solution obtained after the local search. The default value is 30.

- `-dMin`: Final Reference distance between the reference solution and the solution obtained after the local search. The default value is 15.

An example of how to execute AILS-II would be:

```
java -jar AILS.jar -file Instances/X-n214-k11.vrp -rounded true
-best 10856 -limit 100 -stoppingCriterion Time
```

This command will run AILS-II to solve an instance whose parameters are in file “Instances/X-n214-k11.vrp”, considering that the distance between the vertices is rounded to an integer value. We also indicate that the optimal value for this instance is 10856, therefore, if the algorithm finds a solution with this cost, it halts. The stopping criterion is a time limit of 100 seconds.

6. Computational Experiments

The computational experiments were performed on a cluster with 104 nodes, each of them with 2 Intel Xeon E5-2680v2 processors running at 2.8 GHz, 10 cores, and 128 GB DDR3 of 1866 MHz RAM. In the first experiment, we investigate the performance of the proposed algorithm using the set of benchmark instances presented in Uchoa et al. (2017). This set contains 100 CVRP instances whose size ranges from 100 to 1000 vertices. The second experiment assesses the performance of AILS-II on larger instances. We consider the 10 instances proposed in Arnold et al. (2019) which have between 3000 and 30,000 vertices. All experiments were carried out using as stopping criterion of $3n$ seconds, where n is the number of vertices in the instance.

6.1. First Experiment

This experiment contrasts the performance of AILS-II to HGS (Vidal 2022) and FILO (Accorsi and Vigo 2021) on the 100 instances proposed by Uchoa et al. (2017). We run 50 times each of them and report the average gap, defined as:

$$gap = 100 \times \frac{(Avg - \mathrm{BKS})}{\mathrm{BKS}}, \tag{1}$$

where BKS is the objective function value of the best known solution of a given instance and Avg is the average objective function value of the solutions obtained in the independent executions.

Figure 1 shows the average gap of 10 classes of instances that were grouped by size. Considering instances with up to 331 vertices, we observed the following order of best performance: HGS, AILS-II and FILO. In instances with more than 336 vertices, AILS-II achieves the best results and for instances with more than 655 vertices, FILO has a better result than HGS. AILS-II presented a lower computational cost. In these experiments, AILS-II found a better best known solution for the ten following instances: X-n536-k96 = 94828, X-n701-k44=81919, X-n716-k35=43330, X-n766-k71=114416, X-n783-k48=72381, X-n837-k142=193734, X-n895-k37=53848, X-n916-k207=329178, X-n957-k87=85464, X-n979-k58=118913.

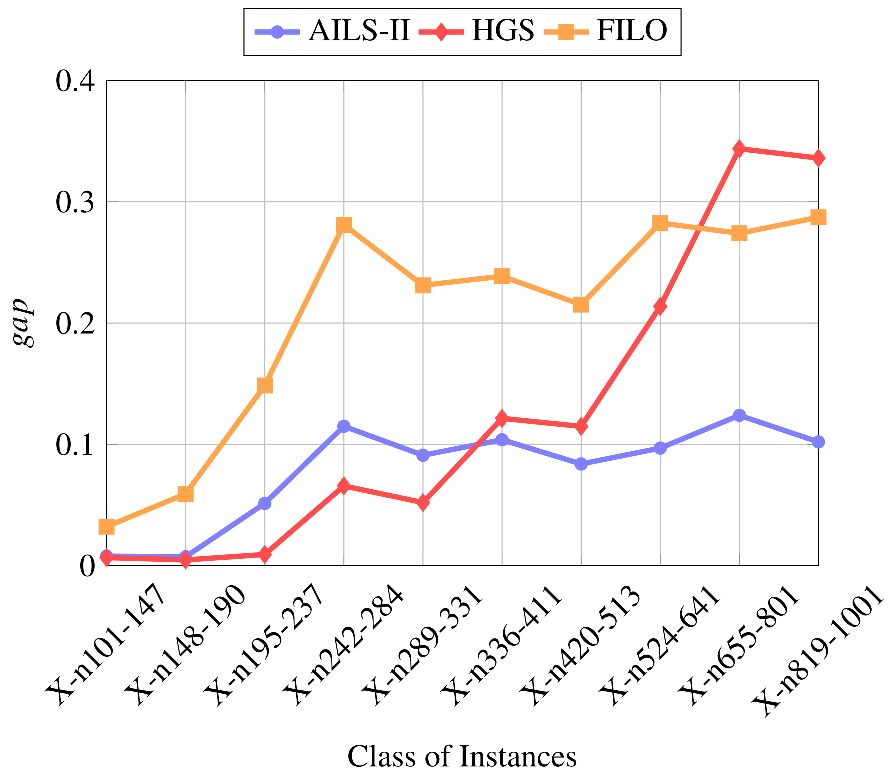

Figure 1 Comparison of the average gaps obtained by the HGS, FILO and AILS-II on the 100 instances proposed by Uchoa et al. (2017).

6.2. Second Experiment

In this experiment, 10 independent executions of AILS-II and FILO were performed on the instances proposed by Arnold et al. (2019). HGS was not considered in this experiment because its author suggested not to use it (Vidal 2022). The reason is that these instances have very different characteristics from those for which the algorithm was designed.

Table 2 shows, for each instance, the instance name, the number of nodes (n), the best-known solution, followed by the results achieved by FILO and AILS-II. The results compiled in the table are the average solution value of the independent executions (Avg), the average gap (gap), the objective function value of the best solution obtained in the runs (Best); and the average time in

Table 2 Results achieved by AILS-II and FILO on the instances introduced by Arnold et al. (2019).

			FILO (Accorsi and Vigo 2021)			AILS-II		
Instance	*n*	BKS	Avg (*gap*)	Best	T (min)	*Avg* (*gap*)	Best	T (min)
Antwerp1	6000	477277	478002.8 (0.1521)	477854	298.3	**477526.5 (0.0523)**	**477466**	**289.0**
Antwerp2	7000	291350	291776.6 (0.1464)	**291511**	348.0	**291715.8 (0.1256)**	291587	**338.4**
Brussels1	15000	501584*	502349.1 (0.1525)	502175	746.5	**501883.5 (0.0597)**	**501735**	**737.8**
Brussels2	16000	345057*	345940.4 (0.2560)	345691	798.3	**345419.0 (0.1049)**	**345253**	**792.3**
Flanders1	20000	7238970*	7249408.7 (0.1442)	7248487	996.3	**7240768.8 (0.0248)**	**7239443**	**982.6**
Flanders2	30000	4367291*	4384803.7 (0.4010)	4382783	1494.8	**4370437.6 (0.0720)**	**4369014**	**1477.3**
Ghent1	10000	469483*	469991.9 (0.1084)	469830	496.5	**469684.0 (0.0428)**	**469476**	**488.5**
Ghent2	11000	257571*	258173.0 (0.2337)	258052	546.9	**257926.1 (0.1379)**	**257670**	**543.1**
Leuven1	3000	192848	193119.1 (0.1406)	193014	146.0	**193013.1 (0.0856)**	**192923**	**143.4**
Leuven2	4000	111391	111706.2 (0.2830)	111618	197.7	**111703.0 (0.2801)**	**111512**	**192.0**
Average			(0.2018)		606.9	(0.0986)		598.4

minutes that the algorithm took to find the best solution in each run (T (min)). BKSs highlighted with '*' were found by AILS-II and improve previous BKSs.

AILS-II achieved the smallest average gap in all 10 instances. For one instance we did not find the best solution. These results confirm the better performance of AILS-II in large instances with respect to the state-of-the-art large scale heuristic method.

7. Final Remarks and Future Work

AILS is an adaptive ILS metaheuristic introduced by Máximo and Nascimento (2021) to solve the CVRP. The authors proposed a hybridization of the path-relinking strategy with AILS, called AILS-PR, to better explore the solution space of such a problem. As a result, AILS-PR achieved outstanding results on the tested instances, which consisted mostly of small- and medium-size instances. However, preliminary experiments indicated that neither AILS-PR nor AILS performed well on large-scale instances. On the one hand, AILS does not intensify the search much. On the other hand, AILS-PR was too slow on large instances.

Therefore, the main goal of this study was to present a more efficient approach to intensification for large instances of the CVRP. In the proposed AILS, called AILS-II, we employed a convergent criterion for both the diversity control strategy and in the acceptance criterion. The results show that the adequate control of these two stages in the adaptive process is of great relevance to ensure good performance.

The computational experiments were performed using a benchmark dataset proposed by Uchoa et al. (2017) that contains 100 instances whose size ranges from 100 to 1000 vertices. We compared

AILS-II with the HGS and FILO. Considering this dataset, AILS-II achieved the best average gap in all instances with more than 336 vertices. To evaluate the performance of AILS for larger instances, we carried out experiments with a dataset proposed by Arnold et al. (2019) that contains 10 large instances of CVRP. These instances have between 3000 and 30,000 vertices. The results of this experiment showed that AILS-II outperformed FILO on all instances.

As future work, we intend to continue investigating more efficient mechanisms for controlling diversity. We also want to explore other search strategies that are robust and simpler. Other areas of interest involve machine learning to optimize the performance of metaheuristics.

Acknowledgments

The authors are grateful for the financial support provided by CNPq and FAPESP (2016/01860-1, 2019/22067-6); the Laboratory for Simulation and High Performance Computing (LaSCADo), funded by (FAPESP) through project 2010/50646-6; and the Center for Mathematical Sciences Applied to Industry (CeMEAI), funded by FAPESP by through the 2013/07375-0 project, for the availability of computer resources.

References

Accorsi L, Vigo D (2021) A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems. *Transportation Science* 55(4):832–856, URL http://dx.doi.org/10.1287/trsc.2021.1059.

Arnold F, Gendreau M, Sörensen K (2019) Efficiently solving very large-scale routing problems. *Computers & Operations Research* 107:32–42, ISSN 0305-0548, URL http://dx.doi.org/10.1016/j.cor.2019.03.006.

Christiaens J, Vanden Berghe G (2020) Slack induction by string removals for vehicle routing problems. *Transportation Science* 54(2):417–433.

Christofides N, Mingozzi A, Toth P (1979) The vehicle routing problem. Christofides N, Mingozzi A, Toth P, Sandi C, eds., *Combinatorial Optimization*, 315–338 (Chichester: Wiley).

Dantzig GB, Ramser JH (1959) The truck dispatching problem. *Management Science* 6(1):80–91.

Dueck G, Scheuer T (1990) Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics* 90(1):161–175, ISSN 0021-9991, URL http://dx.doi.org/10.1016/0021-9991(90)90201-B.

Golden BL, Wasil EA, Kelly JP, Chao IM (1998) The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. *Fleet Management and Logistics*, 33–56 (Springer).

Lenstra JK, Rinnooy Kan AHG (1981) Complexity of vehicle routing and scheduling problems. *Networks* 11(2):221–227.

Lourenço HR, Martin OC, Stützle T (2010) *Iterated Local Search: Framework and Applications*, 363–397 (Boston, MA: Springer US), ISBN 978-1-4419-1665-5, URL http://dx.doi.org/10.1007/978-1-4419-1665-5_12.

Máximo VR, Cordeau JF, Nascimento MC (2022) An adaptive iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research* 148:105954.

Máximo VR, Nascimento MCV (2021) A hybrid adaptive iterated local search with diversification control to the capacitated vehicle routing problem. *European Journal of Operational Research* 294(3):1108–1119, ISSN 0377-2217, URL http://dx.doi.org/10.1016/j.ejor.2021.02.024.

Toth P, Vigo D (2001) An overview of vehicle routing problems. Toth P, Vigo D, eds., *The Vehicle Routing Problem*, 1–26 (Philadelphia, PA, USA: Society for Industrial and Applied Mathematics), URL http://dx.doi.org/10.1137/1.9780898718515.ch1.

Uchoa E, Pecin D, Pessoa A, Poggi M, Vidal T, Subramanian A (2017) New benchmark instances for the capacitated vehicle routing problem. *European Journal of Operational Research* 257(3):845–858, ISSN 0377-2217.

Vidal T (2022) Hybrid genetic search for the CVRP: Open-source implementation and swap* neighborhood. *Computers & Operations Research* 140:105643, ISSN 0305-0548, URL http://dx.doi.org/10.1016/j.cor.2021.105643.