Impact of Iterated Local Search Heuristic Hybridization on Vehicle Routing Problems: Application to the Capacitated Profitable Tour Problem

Hayet Chentli[1(✉)], Rachid Ouafi[1], and Wahiba Ramdane Cherif-Khettaf[2]

[1] Department of Operations Research, USTHB, P.O. Box 32 El Alia, 16111 Bab Ezzouar, Algiers, Algeria
chentli.hayet@gmail.com, hchentli@usthb.dz

[2] LORIA, UMR 7503, Lorraine University, Mines Nancy, Nancy, France

Abstract. The present paper highlights the impact of heuristic hybridization on *Vehicle Routing Problems* (VRPs). More specifically, we focus on the hybridization of the *Iterated Local Search heuristic* (ILS). We propose different hybridization levels for ILS with two other heuristics, namely a *Variable Neighborhood Descent with Random neighborhood ordering* (RVND) and a *Large Neighborhood Search heuristic* (LNS). To evaluate the proposed approaches, we test them on a variant of VRPs called the *Capacitated Profitable Tour Problem* (CPTP). In a CPTP, the visit of all customers is no longer required and the visit of each customer generates a specific profit. The available fleet of vehicle is limited and capacitated. The aim of the CPTP is to choose which set of customers to visit and in which order to maximize the difference between collected profits and routing costs. Our experiments show that the more ILS is hybridized the better are the results. To bring out the effectiveness of the proposed hybrid approach combining ILS, RVND and LNS, a comparison is made between that proposed approach and three local search heuristics from the literature of the CPTP. The obtained results are competitive.

Keywords: Heuristics · Hybridization · Vehicle Routing Problem · Iterative local search

1 Introduction

In recent years, considerable attention has been paid to logistic problems in general and to *Vehicle Routing Problems* (VRPs) in particular. Different methodologies have been adopted to "solve" that kind of problems. Among the proposed methodologies, heuristic algorithms are particularly much studied. Researchers in the fields of combinatorial optimization are trying their best to improve the solution quality and the computing time of previously proposed heuristics, especially when it comes to solve difficult problems as VRPs.

G. H. Parlier et al. (Eds.): ICORES 2018, CCIS 966, pp. 80–101, 2019.
https://doi.org/10.1007/978-3-030-16035-7_5

The present work aims at analyzing the impact of heuristic hybridization on VRPs. More specifically, hybridizations of the *Iterated Local Search heuristic* (ILS) with other single-solution based heuristics are considered.

We recall that ILS principle is to improve a given initial solution by alternating *local search* (LS) and *perturbation procedures*. The role of a LS is to improve a given solution by performing a set of small modifications (or moves) to the studied solution. One can say that the LS visits the neighborhood of the studied solution and selects the best neighboring solution according to some criterion. After some iterations, the LS is no longer able produce better quality solutions using the same set of moves. We say that the heuristic is trapped in a local optimum.

To help LS to escape local optima, ILS provides a perturbation procedure. The latter procedure performs some changes to the current local optimum, producing thereby a new starting solution for the LS. The quality of that new starting solution is generally not as good as the quality of the local optimum. That decrease in the solution quality induced by the perturbation procedure allows the LS to visit a larger search space area.

In the present paper, we attempt to improve a simple ILS heuristic by modifying its local search and perturbation procedures. Several ILS hybridizations are implemented based on several simple LS heuristics, a *Variable Neighborhood Descent with Random neighborhood ordering* (RVND) and a *Large Neighborhood Search heuristic* (LNS).

To assess the performance of the studied heuristics, the latter are tested on a VRP variant called *Capacitated Profitable Tour Problem* (CPTP).

The CPTP has been introduced by Archetti et al. [1] in order to deal with empty returns, that trucks are facing after performing delivery operations (see [1] for more details). The main difference between the CPTP and the classical VRP is the relaxation of the constraints imposing a visit for each customer. In addition, in a CPTP a profit is assigned to each customer and the objective is to maximize the difference between collected profits and routing costs. The number of available vehicles in a CPTP is supposed to be finite. These vehicles are homogeneous with a fixed capacity bound. Each customer in a CPTP has a given pickup demand that must be entirely fulffiled if the customer is visited. Furthermore, if a customer is included in the solution, its demand has to be satisfied by performing a single visit.

The rest of the paper is organized as follows. Section 2 presents some previous works from the literature dealing with the application of ILS to combinatorial optimization problems and VRPs. Hybridizations of ILS with other heuristics are highlighted. A CPTP literature review is also given in that Section. Section 3 describes the proposed approaches. Section 4 discusses the computational results. Finally, a conclusion is given in Sect. 5.

2 Literature Review

The present section is divided into two subsections. In the first subsection, we provide the literature review of the Iterated Local Search (ILS) heuristic. While the second subsection is devoted to the CPTP heuristic approaches proposed in the literature.

2.1 Iterated Local Search Heuristic

According to Lourenço et al. [2], ILS is an efficient heuristic that has several desirable features of a metaheuristic. The main features are the simplicity, the high effectiveness, the robustness and the ease and the malleability of implementation (several implementation choices are left to the developer). The authors also state that ILS effectiveness depends on the choice of the used modules: local search, perturbation procedure and acceptance criterion.

In the literature, several researchers attempted the resolution of combinatorial optimization problems using hybridization of ILS with other heuristics. For instance, Martins et al. [3] developed a *Variable Neighborhood Descent* (VND) combined with an ILS for the Routing and Wavelength Assignment problem. In that paper, VND plays the role of the local search procedure and uses three neighborhood structures. When VND is blocked, ILS perturbs the so far obtained solution and the process iterates until a stopping criterion is met.

Martins et al. [4] implemented a hybrid ILS and RVND heuristic for the Cell Formation Problem. The proposed RVND uses three neighborhood structures. In addition, three perturbation procedures are used in ILS.

Many researchers successfully applied hybrid ILS heuristics to VRP variants. For example, Chen et al. [5] developed a hybridization of ILS with VND for the Capacitated Vehicle Routing Problem. VND uses two inter- and two intra route(s) operators consisting of intra-route relocation, 2-opt, inter-routes swap and 2-opt*. The perturbation phase is performed using the cross-exchange operator.

Subramanian et al. [6] proposed a parallel algorithm combining an ILS with a RVND for solving the Vehicle Routing Problem with Simultaneous Pick-up and Delivery services. Five intra- and seven inter-route(s) neighborhood structures are given together with three perturbation mechanisms.

Subramanian et al. [7] implemented a hybrid algorithm combining an exact method with ILS and RVND for a class of VRPs with heterogeneous fleet. The ILS and RVND heuristics are based on those presented in [6].

Assis et al. [8] presented a hybrid ILS using RVND in the local search phase. The proposed RVND uses six inter- and six intra-route(s) neighborhood structures. That hybrid approach is tested on the multiobjective vehicle routing problem with fixed delivery and optional collections.

Another hybridization of ILS is implemented by Subramanian and Battarra [9] to solve the Travelling Salesman Problem with Pickups and Deliveries. The authors hybridized ILS with RVND. In RVND, four neighborhood structures are given.

Hernández-Pérez et al. [10] studied a hybridization of ILS with VND. The approach is applied to the multi-commodity Pickup-and-Delivery Traveling Salesman Problem. The approach is tested with up to six neighborhood structures and a combination of three shaking procedures.

Todosijević et al. [11] developed another hybrid approach using both ILS and VND for the Swap-Body Vehicle Routing Problem. The used neighborhood structures are 1-Opt, 2-Opt, Or-Opt, relocate and exchange. The shaking procedure is based on customer relocation.

The ILS and the VND heuristics also provide good quality solutions for other variants of VRPs see [12–14]. In addition, the two heuristics perform well on some *Vehicle Routing Problems with Profits* (see [15]). Furthermore, several versions of VND are used to solve different variants of transportation problems (see [16–18]).

As one can see from the literature review, several papers use combinations of ILS and RVND for solving VRP variants. However, to the best of our knowledge, only one work has been addressed using a hybridization of ILS with LNS [19]. ILS and LNS heuristics are nevertheless quite effective in solving VRP variants as well as other transportation problems. For ILS, we refer the reader to the papers [20–23], and for LNS, we refer the reader to the papers [24–28].

2.2 Capacitated Profitable Tour Problem

Despite its importance, the CPTP has not received a lot of attention from researcher. Archetti et al. [1] introduced the CPTP and proposed three methodologies to solve that problem. The proposed methodologies are the Tabu Feasible (TF), the Tabu Admissible (TA) and the Variable Neighborhood Search (VNS) heuristics. Both TF and TA algorithms use two inter-route operators. The first operator is called 1-move. 1-move either relocates a given customer in a different route or deletes that customer completely from the solution. The second movement is called swap-move. Swap-move either exchanges the positions of two given customers from two different routes or deletes a customer and replaces it by an unrouted one.

In order to deal with infeasible solutions obtained by the TA algorithm, Archetti et al. proposed a repair heuristic based on series of 1-move. In addition, the authors evaluated the solutions according to several criteria including the difference between total profit and total distance, the number of routes, the route duration and the maximum constraint violation.

The VNS algorithm uses the TF algorithm with a small iteration number, which allows the visit of a larger area within the search space.

Some researchers proposed exact methods for the CPTP. As the present work deals with heuristic approaches, we do not describe those exact methods.

3 The Proposed Methodology

In the present section, we describe the implemented construction heuristic and the studied approaches: ILS, LNS and RVND. We also describe the tested hybridizations.

Note that a preliminary work dealing with ILS hybridization has been presented in [29]. The present work extends the one proposed in [29] by providing: (*i*) other heuristic approaches combining ILS with other neighborhood operators, (*ii*) a hybridization of ILS with both LNS and a neighborhood operator, (*iii*) a detailed comparison between the use of the *basic greedy* heuristic and a *random insertion* heuristic within the perturbation procedure, (*iv*) detailed results of the hybrid heuristic combining ILS, RVND and LNS compared with Archetti et al. [1] results.

3.1 Construction Heuristic

The implemented construction heuristic is a sequential heuristic based on the I1 heuristic of Solomon [30]. I1 was first developed for the Vehicle Routing Problem with Time Window. The pseudo-code of the construction heuristic is displayed in Algorithm 1.

Algorithm 1. Construction heuristic for the CPTP.

Inputs:
A CPTP instance
A list L_{unr} containing all the unrouted customers
A number $nbRoutes = 0$ of the current solution routes
Outputs:
A feasible solution
while $nbRoutes <$ vehicle number **do**
  Generate a new route;
  $nbRoutes++$;
  Add a seed customer to the new route;
  while $\exists u \in L_{unr}$ whose insertion leads to a feasible solution **do**
    Evaluate the insertion of each unrouted customer $u \in L_{unr}$ into the studied route r_{stu};
    Choose the best insertion position for each u using the criterion $cr_1(i, u, j)$;
    Choose the customer u^* that has the best value of $cr_1(i, u, j)$;
    Insert the customer u^* in its best insertion position within r_{stu};
    Delete u^* from L_{unr};
  end while
end while

An empty route is considered in the first iteration of the construction heuristic. That empty route is first filled with a seed customer which is randomly chosen. After that, the heuristic evaluates the insertion of the remaining unrouted customers into the route. The best insertion position of each unrouted customer u between customers i and j is selected. This is done according to a given criterion denoted $cr_1(i, u, j)$. Among all the best insertion positions, the construction heuristic chooses the one that optimizes the given criterion. That process iterates until no customer can be inserted into the current route. If some customers are still unrouted, a new route is generated and the process is repeated. The heuristic stops either if there are no more unrouted customers or if the number of generated routes exceeds the vehicle number.

To describe $cr_1(i,u,j)$, let us consider $(i_0, i_1, \ldots, i_h)$ as the current route where i_ρ stands for the ρ^{th} position in the route if $\rho \notin \{0,h\}$, i_ρ stands for the depot otherwise ($\rho \in \{0,h\}$). The best insertion position of customer u within the current route is selected according to Expressions (1)–(4) (Source [29]). In these Equations c_{ij} refers to the distance between customers i and j, pr_u refers to customer u profit, $\alpha_1, \alpha_2 \geq 0$ with $\alpha_1 + \alpha_2 = 1$ are two parameters set by the user.

$$cr_1(i(u),u,j(u)) = \max\{cr_1(i_{\rho-1},u,i_\rho), \rho = 1,\ldots,h\}; \text{ (Source [29])} \quad (1)$$

$$cr_1(i,u,j) = \alpha_1 \cdot cr_{11}(i,u,j) - \alpha_2 \cdot cr_{12}(i,u,j); \text{ (Source [29])} \quad (2)$$

$$cr_{11}(i,u,j) = pr_u; \text{ (Source [29])} \quad (3)$$

$$cr_{12}(i,u,j) = c_{iu} + c_{uj} - c_{ij}. \text{ (Source [29])} \quad (4)$$

The Eqs. (1)–(4) have been already presented in [29] on page 117, Sect. 2.1.

Note that different values of parameters α_1 et α_2 can lead to different solutions for the CPTP.

3.2 Iterated Local Search

ILS principle is described in Sect. 1. The pseudo-code of the ILS heuristic used in the present paper is given in Algorithm 2. ILS starts from an initial solution given by the construction heuristic presented in Sect. 3.1. Then a local search procedure is executed for a given number of iterations. If the obtained solution is better than the current best one, the best solution is replaced by the obtained one. The solution is, after that, perturbed using a perturbation procedure. The process is repeated until a stopping criterion is met. In the present work, the stopping criterion stands for the completion of a given number of iterations without improvement.

Algorithm 2. Iterated Local Search.

1: **Inputs:**
   A CPTP instance
2: **Outputs:**
   The best solution found
3: Generate an initial solution;
4: **while** stopping criterion is not met **do**
5: Execute a local search procedure;
6: Update the best solution;
7: Perturb the obtained solution;
8: **end while**

Local Search. As mentioned by Talbi [31], any single-solution based metaheuristic can be used in the local search phase of an ILS.

In the present work, seven basic ILS versions are first tested. In each ILS basic version, a different neighborhood structure (or neighborhood operator) is used. The implemented neighborhood operators consist of four intra- and three inter-route(s) operators. The latter are described in what follows. Examples of neighborhood movements for each operator are given in Fig. 1. Figure 1 was first given in [29] on page 118.

2-Opt introduces two new arcs and deletes two other arcs in a given route by connecting two customers k and l and reversing the path between those customers. In Fig. 1, the arcs $(1, 4)$ and $(2, 5)$ are deleted, the arcs $(1, 2)$ and $(4, 5)$ are added, customers 1 and 2 are connected and the path $(4-3-2)$ is reversed. For maintaining the route connectivity, customers 4 and 5 are connected.

*2-Opt** divides two given routes into four segments: initial and final segments. Then, the operator connects each first segment from a route with a second segment from the other route. In Fig. 1, the first route (0–1–2–3–0) is disconnected into a first segment $(0-1)$ and a second segment $(2-3-0)$. The second route $(0-4-5-6-0)$ is disconnected into a first segment $(0-4-5)$ and a second segment $(6-0)$. After that, $(0-1)$ is connected to $(6-0)$ and $(0-4-5)$ is connected to $(2-3-0)$.

Intra-route 1-0 Exchange relocates a customer l into a position k within a same route. In Fig. 1, customer 2, which is in the 5^{th} position of the route, is relocated in the second position within the same route.

Inter-routes 1-0 Exchange relocates a customer l into a position k in a different route. In Fig. 1, customer 3, which is in the third position of the first route, is relocated in the 4^{th} position of the second route.

Intra-route 1-1 Exchange exchanges the positions of two customers within a same route. In Fig. 1, customer 2 is relocated at the position of customer 5, and customer 5 is relocated at the position of customer 2. No path is reversed.

Inter-routes 1-1 Exchange exchanges the positions of two customers from two different routes. In Fig. 1, customer 3 is relocated at the position of customer 6, and customer 6 is relocated at the position of customer 3.

Or-Opt relocates two consecutive customers (or an arc) in a different position within a same route. In Fig. 1, the arc $(1, 2)$ is relocated between the depot and customer 3.

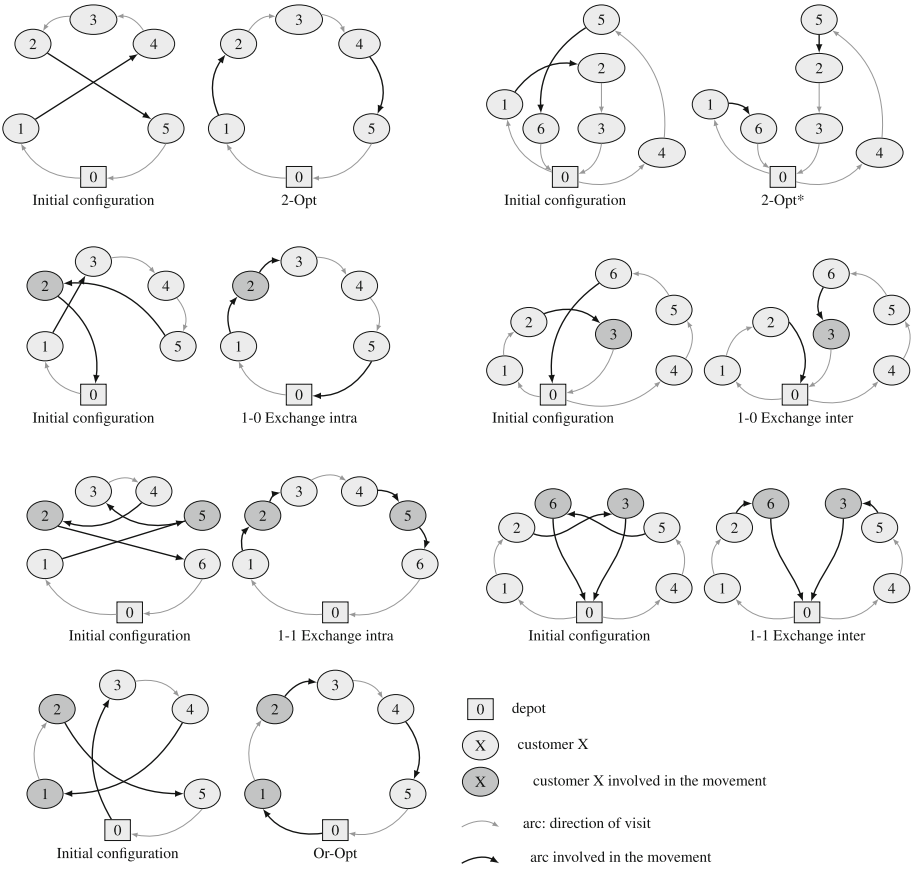

Fig. 1. Illustration of neighborhood movements in the RVND heuristic (Source [29]).

Each combination of ILS with a neighborhood operator is denoted by ILS_NeiOp where NeiOp stands for the used neighborhood operator. Thus, we have: ILS_2-Opt, ILS_2-Opt*, ILS_1-0 Exchange-intra, ILS_1-0 Exchange-inter, ILS_1-1 Exchange-intra, ILS_1-1 Exchange-inter and ILS_Or-Opt.

Perturbation Mechanism. The perturbation procedure destroys the solution obtained by the local search procedure to escape local optima. To do so, the *random removal* operator described by Pisinger and Ropke [32] is used. Before re-applying the local search procedure, some customers may be added to the obtained solution using the *basic greedy* heuristic described by the same authors [32].

3.3 Hybrid Iterated Local Search Heuristic

In the present work, several hybridization of ILS are proposed. They are described in what follows.

Hybrid ILS_LNS Heuristic. In [29], several versions of the LNS are tested. The latter versions use a unique removal and a unique insertion operator. All the removal/insertion operators proposed by Pisinger and Ropke [32] are tested using the CPTP constraints and objective function. Each implemented LNS version consists of a different couple of removal/insertion operators.

Each LNS version starts from a solution generated by the construction heuristic. At each iteration, LNS deletes a given number of customers using its removal operator. Then, LNS inserts a set of customers using its insertion operator. LNS stops when a given number of iterations without improvement is reached.

After an experimental study, we choose the *related removal* and the *regret heuristic* with a regret number equal to 4 as the couple of removal/insertion operators used in LNS.

For more information regarding the selection of the removal/insertion couple, we refer the reader to [29].

The hybrid ILS_LNS heuristic is a multi-start heuristic that executes, at each iteration, an ILS heuristic using LNS as a local search procedure. The initial solutions are obtained by iteratively modifying parameter values of the construction heuristic described in Sect. 3.1. All the possible combinations of α_1 and α_2 parameter values are considered. ILS_LNS uses the same perturbation procedure as the one described in Sect. 3.2.

As defined in Sect. 3.2, ILS stopping criterion consists in the completion of a given number of iterations without improvement.

ILS_LNS stops when all combinations of α_1 and α_2 parameter values are tested.

Hybrid ILS_RVND Heuristic. The RVND heuristic uses all the seven operators described in Sect. 3.2. At each iteration of RVND, the neighborhood operators are chosen in a random way. Actually, all the operators are put in a list of available operators, and each time an operator is used, the heuristic deletes that operator from the list. RVND stops when the list of available operators is empty. Hence, each operator is executed only once.

The hybrid ILS_RVND heuristic differs from the ILS_LNS heuristic (presented above) in the use of RVND instead of LNS in the local search phase.

Hybrid LNS_RVND Heuristic. LNS_RVND is not a multi-start heuristic. This hybrid heuristic begins with an initial solution obtained by the construction heuristic given in Sect. 3.1 using random values for parameters α_1 and α_2. Then, the LNS heuristic is run until reaching a given number of iterations. The obtained solution is possibly improved using the RVND heuristic with some probability. After that, the LNS_RVND heuristic goes back again to LNS and that process iterates until reaching a given number of iterations without improvement.

Hybrid ILS_NeiOp_LNS Heuristic. ILS_NeiOp_LNS is a multi-start heuristic that hybridizes ILS_NeiOp with LNS. NeiOp is defined in a similar manner as described in Sect. 3.2.

The neighborhood operator used in this heuristic is the one that gives the best results with respect to other neighborhood operators when combined to ILS.

In ILS_NeiOp_LNS, the construction heuristic is first run to generate different initial solutions at each iteration. Then, for each initial solution, an ILS heuristic is run until reaching a given number of iterations without improvement. In ILS, a LNS heuristic is executed followed by the selected neighborhood operator. The combination of LNS with the selected neighborhood operator plays the role of the local search procedure in ILS. This combination of LNS with the neighborhood operator is repeated for a given number of iterations. When the local search stops, the perturbation procedure is run. The latter procedure is the same as the one presented in Sect. 3.2. The pseudo-code of ILS_NeiOp_LNS is given in Algorithm 3.

Algorithm 3. ILS_NeiOp_LNS.

```
1: Inputs:
       A CPTP instance
2: Outputs:
       The best encountered solution
3: while stopping criterion is not met do
4:     Generate an initial solution;
5:     while ILS stopping criterion is not met do
6:         while stopping criterion is not met do
7:             Run LNS;
8:             Run NeiOp;
9:         end while
10:        Update the best encountered solution;
11:        Perturb the current solution;
12:    end while
13: end while
```

Hybrid ILS_RVND_LNS Heuristic. This hybrid heuristic is quite similar to the one presented in Sect. 3.3. The only difference is that ILS_RVND_LNS uses RVND instead of the neighborhood operator after LNS.

The pseudo-code of ILS_RVND_LNS is given in Algorithm 4.

Algorithm 4. ILS_RVND_LNS.

```
Inputs:
    A CPTP instance
Outputs:
    The best encountered solution
while stopping criterion is not met do
    Generate an initial solution;
    while ILS stopping criterion is not met do
        while stopping criterion is not met do
            Run LNS;
            Run RVND;
        end while
        Update the best encountered solution;
        Perturb the current solution;
    end while
end while
```

4 Computational Results

In the present Section, we begin by describing the CPTP instances proposed in the literature and studied in the present work. After that, we analyze the results of each proposed approach and we evaluate the impact of the hybridization. Finally, the approach that provides the best results is compared to some CPTP heuristics from the literature.

In order to quickly determine the best approach among the proposed ones, each approach is executed using a relatively small number of iterations. However, more iterations are used for the comparison between the best approach with the literature ones.

Note that some of the experimental results/heuristic tuning details are not given in the present work as they have already been published in the conference paper [29]. Those experimental results/heuristic tuning details concern the couple of operators chosen for LNS, the comparison between LNS and ILS_LNS and the tuning of both ILS_RVND and LNS_RVND.

The new experimentations are implemented in C and performed on a personal laptop with an Intel(R) Core (TM) i5-4210U CPU @ 1.70 GHz with 6.00 Gb RAM and 64-bit operating system.

Due to the random aspect of the approaches, they all are executed 3 times for each instance. We report the best encountered solutions in terms of percentage deviation from the best solutions presented by Archetti et al. [1]. A percentage deviation (gap) of a heuristic a from a heuristic b is computed according to the following Expression

$$gap = 100 \cdot \frac{z_b - z_a}{z_b}$$

where z_a and z_b are the objective function values obtained by heuristics a and b respectively.

4.1 CPTP Instances

The CPTP instances studied in the present work were proposed by Archetti et al. [1]. The authors modified the Capacitated Vehicle Routing Problem instances (CVRP) described by Christofides et al. [33] with 50 to 199 customers. Archetti et al. [1] generated three set of instances from the CVRP instances by varying the capacity bounds and the number of vehicles.

The first set of CPTP instances consists of the original 10 CVRP instances in which each customer i has a profit pr_i computed following the Expression $pr_i = (0.5 + h) \cdot d_i$, where d_i is the demand of i and h is randomly chosen from $[0, 1]$.

The second set of CPTP instances consists of 90 different instances obtained by modifying the first set of instances. Actually, Archetti et al. [1] consider the cases $Q = 50$, $Q = 75$ and $Q = 100$, where Q stands for to the capacity bound. For each case, three instances are generated using different vehicle numbers. The latter numbers are chosen from the set $\{2, 3, 4\}$. In the second set of CPTP instances, the profits are computed in the same manner as described for the first set.

The third set of CPTP instances consists of 30 different instances obtained by modifying the first set of instances. Archetti et al. [1] maintain the same capacity bounds as those presented for the CVRP. However, they consider three cases for the vehicle numbers. The latter are chosen from the set $\{2, 3, 4\}$.

A total of 130 CPTP instances are thus proposed by Archetti et al. [1]. As instance types $p03$ and $p08$ of Archetti et al. [1] are exactly the same, we do not consider instances of type $p03$. Hence we obtain a total of 117 CPTP instances.

4.2 Study of Basic ILS Heuristics

As said in Sect. 3.2, seven basic version of ILS are tested. Each version differs from the others in the used neighborhood operator. The tested versions are ILS_2-Opt, ILS_2-Opt*, ILS_1-0 Exchange-intra, ILS_1-0 Exchange-inter, ILS_1-1 Exchange-intra, ILS_1-1 Exchange-inter and ILS_Or-Opt.

As these basic ILS heuristics are quite fast, we decide to fix their number of iterations without improvement in ILS to 500 instead of 50.

Table 1 displays the obtained results in terms of average gap (among all instances) from Archetti et al. [1] results. That Table also displays the average computing time (among all instances) in seconds (CPU).

Table 1. Comparison between basic ILS heuristics.

	ILS_2-Opt	ILS_2-Opt*	ILS_1-0 Exchange-intra	ILS_1-0 Exchange-Inter	ILS_Or-Opt	ILS_1-1 Exchange-intra	ILS_1-1 Exchange_inter
Gap	5.41	4.94	5.78	5.42	5.76	5.52	5.07
CPU	15.70	14.98	15.72	16.42	14.58	15.15	15.92

The results of Table 1 show that the 2-Opt* operator performs better than the other operators in terms of solution quality and computing time.

Hence, in the reminder of the present work, the basic ILS heuristic will refer to the basic ILS heuristic using the 2-Opt* operator. That heuristic will be compared to the other implemented heuristics.

4.3 Study of ILS_LNS Heuristic

Several tests were performed in [29] to determine the best removal/insertion couple of operators of LNS. In addition, LNS was compared with ILS_LNS. ILS_LNS was able to reach better quality solutions and was faster than LNS.

In the present Section, we compare the results of ILS_LNS using the selected couple of removal/insertion operators (Source [29]) with those of the basic ILS heuristic presented in Sect. 4.2.

ILS_LNS is run until 50 iterations without improvement are reached. We remarked that, with only 50 iterations, ILS_LNS is more time consuming than the above studied heuristics using 500 iterations. Hence, we maintain ILS_LNS iteration number to 50.

As one can see from Table 2, the average gap of ILS_2-Opt* is slightly better that the average gap of ILS_LNS. Regarding the average computing time, ILS_LNS appears to be slower than ILS_2-Opt*.

Table 2. Comparison between ILS_LNS and ILS_2-Opt*.

	ILS_LNS	ILS_2-Opt*
Gap	5.07	4.94
CPU	36.35	14.98

4.4 Study of ILS_RVND Heuristic

As the basic ILS, ILS_RVND is very fast in comparison with the ILS_LNS heuristic. Hence, we decide to fix the number of iterations without improvement in the ILS embedded in ILS_RVND to 500. Results of ILS_RVND using 50 iterations are provided in [29].

Table 3 compares the results of ILS_RVND (Source [29]) with both ILS_LNS (Source [29]) and ILS_2-Opt*. We remark that the average gaps of ILS_2-Opt* and ILS_RVND are quite similar. In addition, the average computing time of ILS_RVND is slightly better than the average computing time of ILS_2-Opt*. We think that difference between the two heuristics can be more evident if the iteration number increases and/or if the heuristics are hybridized with other ones.

Table 3. Comparison between ILS_LNS, ILS_2-Opt* and ILS_RVND.

	ILS_LNS	ILS_2-Opt*	ILS_RVND
Gap	5.07	4.94	4.95
CPU	36.35	14.98	10.57

4.5 Study of LNS_RVND Heuristic

LNS_RVND is described in Sect. 3.3. The results related to this hybrid heuristic are presented in [29].

We remark that the results of LNS_RVND (Source [29]) are quite "disappointing" in comparison with other approaches. Indeed, the heuristic obtains the worst average gap and computing time. That can be seen in Table 4.

From the results displayed in Table 4, we conclude that the multi-start ILS heuristic has a considerable impact on the solution quality and the speed of finding solutions.

Table 4. Comparison between ILS_LNS, ILS_2-Opt*, ILS_RVND and LNS_RVND.

	ILS_LNS	ILS_2-Opt*	ILS_RVND	LNS_RVND
Gap	5.07	4.94	4.95	11.94
CPU	36.35	14.98	10.57	67.10

4.6 Study of ILS_2-Opt*_LNS Heuristic

ILS_2-Opt*_LNS is described in Sect. 3.3. In order to have a good balance between solution quality and computing time, the 2-Opt*_LNS heuristic is repeated 7 times at each iteration of ILS. The number of iterations without improvement of the embedded ILS heuristic $maxOcc$ is set to 200. While the number of iterations without improvement of LNS $maxOcc_{LNS}$ is fixed to 20.

Table 5 presents the results of ILS_2-Opt*_LNS compared with those of the previously studied heuristics. We remark that ILS_2-Opt*_LNS is able to reach the best average gap in comparison with the other heuristics. ILS_2-Opt*_LNS seems to be relatively time consuming. However, the computing time of this heuristic is still reasonable, especially if we take the solution quality into account.

Table 5. Comparison between ILS_2-Opt*_LNS and the previously studied heuristics.

	ILS_LNS	ILS_2-Opt*	ILS_RVND	LNS_RVND	ILS_2-Opt*_LNS
Gap	5.07	4.94	4.95	11.94	1.72
CPU	36.35	14.98	10.57	67.10	36.64

4.7 Study of ILS_RVND_LNS Heuristic

ILS_RVND_LNS is described in Sect. 3.3. As in ILS_2-Opt*_LNS, the local search phase of ILS_RVND_LNS, which consists in the RVND_LNS heuristic, is repeated 7 times at each iteration of ILS. The numbers of iterations without improvement in the ILS heuristic $maxOcc$ and in the LNS heuristic $maxOcc_{LNS}$ are set to 200 and 20 respectively.

Note that, contrary to the LNS_RVND studied in Sect. 3.3, the combination of LNS and RVND involved in ILS_RVND_LNS uses the RVND heuristic with a probability equal to 1. That modification is motivated by the fact that ILS_RVND gives good quality solutions in comparison with both ILS_LNS and LNS_RVND.

Table 6 compares the results of ILS_RVND_LNS (Source [29]) with those of the other approaches presented in the present work. From that Table, we remark that ILS_RVND_LNS provides the best average gap without being too time consuming. When comparing ILS_RVND_LNS with ILS_2-Opt*_LNS, we can see that ILS_RVND_LNS is also better in terms of average results (gaps and computing time). That confims our assumption that the use of RVND instead of a neighborhood operator in the ILS heuristic can lead to better results when using more iterations and/or when ILS is hybridized with other heuristics (more than two heuristics are involved).

We also remark that better average gaps are obtained when the heuristics are hybridized. However, that hybridizations can lead to an increase of the computing time. We think that the hybridized heuristics are more time consuming because they are first trapped in local optima then they extract themselves from these optima. That process is repeated several times. On the other hand, the basic heuristic (with a basic level of hybridization or no hybridization at all) are quickly trapped in local optima.

Table 6. Comparison between ILS_RVND_LNS and the other proposed heuristics.

	ILS_LNS	ILS_2-Opt*	ILS_RVND	LNS_RVND	ILS_2-Opt*_LNS	ILS_RVND_LNS
Gap	5.07	4.94	4.95	11.94	1.72	1.57
CPU	36.35	14.98	10.57	67.10	36.64	28.39

4.8 Study of the Perturbation Procedure

In all the studied approaches described so far, we use the *random removal* and the *basic greedy* heuristic (both described in [32]).

Initially, we wanted to use the *random removal* combined to a random insertion heuristic in order to change the characteristics of the solution after a local search is performed. We thought that the two random heuristics can lead to a more diversified search and thus, to better results.

In practice, we found that the use of the *random removal* combined to a random insertion introduces too much diversification. That diversification could not be correctly handled by the local search procedure. Hence, we decided to use the *random removal* combined with the *basic greedy* heuristic instead.

Figure 2 describes the gaps obtained by both versions of the perturbation procedure: with *basic greedy* and the random insertion. The tests are performed using ILS_RVND_LNS.

In Fig. 2, the gap for each instance is shown. From that Figure, we can see that the perturbation using the *basic greedy* generally provides better gaps. Actually the random insertion outperforms *basic greedy* in only 4 cases. The average computing time reached when using *basic greedy* is slightly worse that the average computing time reached when using the random insertion. Indeed, the farmer is equal to 37.85 seconds while the latter is equal to 29.52 seconds.

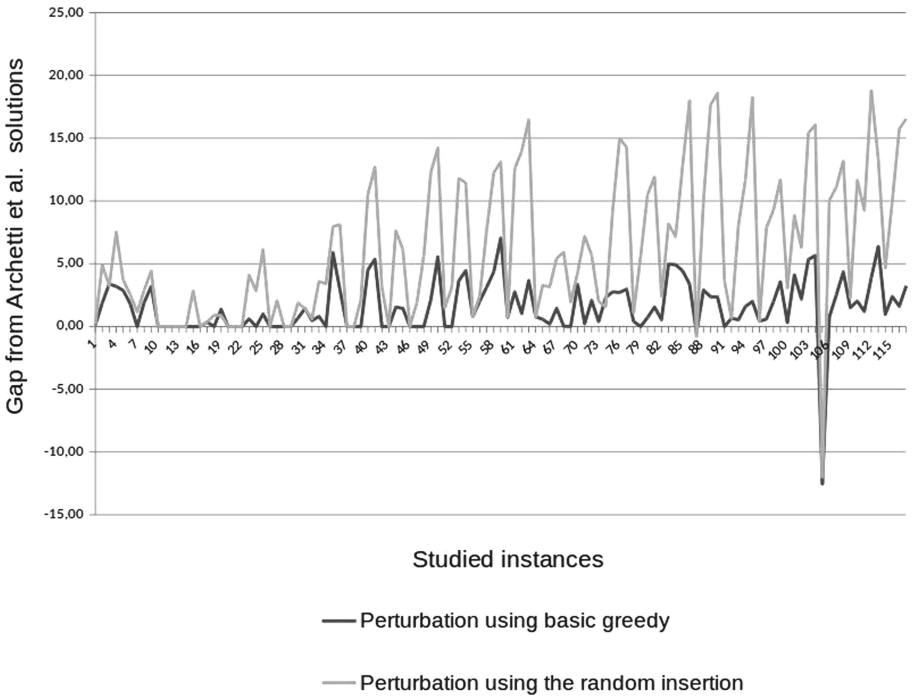

Fig. 2. Comparison between the use of the basic greedy operator and the random insertion for the perturbation procedure.

4.9 Comparison of ILS_RVND_LNS with Other Approaches from the Literature

ILS_RVND_LNS gives the best results among all the implemented approaches. Therefore, it is compared with other approaches from the literature.

ILS_RVND_LNS is compared with TF, TA and VNS heuristics proposed by Archetti et al. [1].

In order to have relatively comparable results for the compared approaches (in terms of solution quality and computing time), we execute ILS_RVND_LNS 20 times at each run.

Table 7 provides the detailed results. In that Table, *ins* stands for the instance name, where an instance $pXY-m-Q$ refers to the original instance pXY of the CVRP in which the vehicle number is fixed to m and the capacity bound is set to Q. n stands for the customer number. VNS, TF, TA and ILS_RVND_LNS refer to the objective values obtained by VNS, TF, TA and ILS_RVND_LNS respectively. CPU is the computing time of ILS_RVND_LNS is seconds. $CPU(min)$ stands for the average computing time in minutes of all the heuristics among all the instances.

Note that the detailed computing time of VNS, TF and TA can be found in [1].

From Table 7, we remark that ILS_RVND_LNS provides 6 new best solutions. In addition, the heuristic reaches the literature solutions in 55 cases. ILS_RVND_LNS is not able to reach the literature solutions in 56 cases. However, the average gap of ILS_RVND_LNS is relatively small. Indeed, it is equal to 0.66%.

The computing time of ILS_RVND_LNS is reasonable.

Table 7. Detailed results for CPTP instances.

Ins	n	VNS	TF	TA	ILS_RVND_LNS	CPU
p06-10-160	50	258,97	258,97	255,38	**259,12**	356,73
p07-20-140	75	**534,81**	525,06	527,90	524,39	1006,98
p08-15-200	100	**663,98**	657,31	656,32	649,67	2330,63
p09-10-200	150	1189,33	**1192,68**	1143,65	1162,55	4194,31
p10-20-200	199	**1773,65**	1761,37	1759,81	1741,43	10736,14
p13-15-200	120	284,71	269,74	274,28	**289,59**	2417,94
p14-10-200	100	**890,44**	886,78	888,18	**890,44**	1283,71
p15-15-200	150	**1168,63**	1156,01	1134,17	1157,38	4945,23
p16-20-200	199	**1791,78**	1764,15	1776,41	1747,06	8435,07
p06-2-50	50	**33,88**	**33,88**	**33,88**	**33,88**	38,71
p07-2-50	75	**49,18**	**49,18**	**49,18**	**49,18**	51,73
p08-2-50	100	**57,75**	**57,75**	**57,75**	**57,75**	76,92
p09-2-50	150	**65,03**	63,89	**65,03**	**65,03**	97,6
p10-2-50	199	**70,87**	**70,87**	**70,87**	**70,87**	129,57
p13-2-50	120	**64,12**	**64,12**	**64,12**	**64,12**	129,66
p14-2-50	100	**43,26**	**43,26**	**43,26**	**43,26**	76,34
p15-2-50	150	**64,98**	**64,98**	**64,98**	**64,98**	107,39
p16-2-50	199	**66,81**	**66,81**	**66,81**	66,39	142,77
p06-3-50	50	**40,95**	**40,95**	**40,95**	**40,95**	56,78
p07-3-50	75	**69,94**	**69,94**	**69,94**	**69,94**	74,12
p08-3-50	100	**80,82**	**80,82**	**80,82**	**80,82**	121,37

Table 7. (*continued*)

p09-3-50	150	**96,16**	**96,16**	**96,16**	**96,16**	162,38
p10-3-50	199	**103,79**	**103,79**	**103,79**	**103,79**	209,51
p13-3-50	120	**87,25**	**87,25**	**87,25**	**87,25**	207,5
p14-3-50	100	**59,43**	**59,43**	**59,43**	**59,43**	74,87
p15-3-50	150	**96,42**	**96,42**	**96,42**	**96,42**	173,42
p16-3-50	199	**99,7**	**99,7**	**99,7**	**99,7**	228,97
p06-4-50	50	**45,43**	**45,43**	**45,43**	**45,43**	77,98
p07-4-50	75	**90,65**	**90,65**	**90,65**	**90,65**	99,74
p08-4-50	100	**100,36**	98,47	**100,36**	99,76	174,71
p09-4-50	150	**121,35**	**121,35**	**121,35**	**121,35**	247,87
p10-4-50	199	**134,81**	**134,81**	**134,81**	**134,81**	297,66
p13-4-50	120	**104,18**	103,73	103,72	103,34	243,39
p14-4-50	100	**68,63**	**68,63**	**68,63**	**68,63**	133,17
p15-4-50	150	**124,02**	**124,02**	**124,02**	119,52	252,23
p16-4-50	199	**131,37**	**131,37**	**131,37**	**131,37**	327,39
p06-2-75	50	**72,28**	**72,28**	**72,28**	**72,28**	49,4
p07-2-75	75	**92,44**	**92,44**	**92,44**	**92,44**	69,34
p08-2-75	100	**106,15**	**106,15**	**106,15**	**106,15**	143,26
p09-2-75	150	**117,66**	**117,66**	**117,66**	**117,66**	136,25
p10-2-75	199	**124,85**	**124,85**	**124,85**	**124,85**	167,55
p13-2-75	120	**110,12**	**110,12**	**110,12**	**110,12**	155,25
p14-2-75	100	**77,09**	**77,09**	**77,09**	**77,09**	70,38
p15-2-75	150	**120,93**	**120,93**	**120,93**	**120,93**	144,84
p16-2-75	199	**123,38**	**123,38**	**123,38**	**123,38**	179,58
p06-3-75	50	**92,32**	**92,32**	**92,32**	**92,32**	75,7
p07-3-75	75	**131,12**	**131,12**	**131,12**	**131,12**	106,59
p08-3-75	100	**147,55**	**147,55**	145,87	**147,55**	238,69
p09-3-75	150	**160,96**	**160,96**	**160,96**	160,66	238,63
p10-3-75	199	**177,9**	**177,9**	176,50	176,22	263,4
p13-3-75	120	**139,37**	137,95	137,45	**139,37**	375,34
p14-3-75	100	**112,56**	112,51	**112,56**	**112,56**	104,9
p15-3-75	150	**174,58**	**174,58**	**174,58**	**174,58**	225,78
p16-3-75	199	**179,55**	**179,55**	179,23	177,35	279,69
p06-4-75	50	**99,37**	**99,37**	**99,37**	**99,37**	100,6
p07-4-75	75	**158,11**	**158,11**	**158,11**	**158,11**	147,84
p08-4-75	100	**185,27**	**185,27**	**185,27**	181,42	400,73
p09-4-75	150	**204,25**	203,24	203,24	201,47	335,87
p10-4-75	199	**229,27**	**229,27**	**229,27**	225,22	381,49

Table 7. (*continued*)

p13-4-75	120	**161,62**	160,68	157,98	161,59	548,71
p14-4-75	100	**139,88**	139,67	139,83	**139,88**	187,11
p15-4-75	150	**219,22**	**219,22**	216,61	**219,22**	318,75
p16-4-75	199	**235,03**	**235,03**	**235,03**	228,49	392,83
p06-2-100	50	**100,27**	99,50	99,50	**100,27**	63,84
p07-2-100	75	**132,7**	**132,7**	**132,7**	**132,7**	103,65
p08-2-100	100	**158,21**	**158,21**	**158,21**	**158,21**	134,63
p09-2-100	150	**161,23**	**161,23**	**161,23**	161,15	176,16
p10-2-100	199	**171,24**	**171,24**	**171,24**	171,19	189,8
p13-2-100	120	**145,75**	145,67	145,67	**145,75**	248,41
p14-2-100	100	**125,29**	**125,29**	**125,29**	**125,29**	123,27
p15-2-100	150	**169,71**	**169,71**	**169,71**	**169,71**	181,41
p16-2-100	199	**177,23**	**177,23**	175,57	173,56	217,8
p06-3-100	50	**134,72**	**134,72**	**134,72**	**134,72**	93,74
p07-3-100	75	**185,25**	184,88	**185,25**	184,88	167,88
p08-3-100	100	**218,63**	**218,63**	218,33	218,43	284,91
p09-3-100	150	**230,49**	229,61	229,61	229,58	284,75
p10-3-100	199	**250,18**	246,56	246,95	246,56	316,77
p13-3-100	120	**181,63**	177,76	180,04	180,79	625,97
p14-3-100	100	**182,31**	179,48	**182,31**	**182,31**	195,82
p15-3-100	150	**244,08**	241,84	**244,08**	243,89	279,08
p16-3-100	199	**258,07**	257,10	252,44	255,38	333,9
p06-4-100	50	**153,3**	**153,3**	152,97	152,97	120,51
p07-4-100	75	**233,4**	**233,4**	232,05	226,61	225,8
p08-4-100	100	**268,34**	266,23	266,08	259,2	504,64
p09-4-100	150	**290,54**	**290,54**	290,15	285,3	433,15
p10-4-100	199	**324,02**	321,17	321,03	320,07	508,78
p13-4-100	120	200,62	178,82	183,66	**202,21**	1208,47
p14-4-100	100	**237,68**	236,50	**237,68**	**237,68**	251,91
p15-4-100	150	**308,07**	305,30	304,81	302,78	437,42
p16-4-100	199	**336,24**	328,20	329,53	328,29	515,7
p06-2-9	50	**168,6**	**168,6**	**168,6**	**168,6**	93,97
p07-2-9	75	**199,97**	**199,97**	**199,97**	**199,97**	131,79
p08-2-9	100	**330,14**	319,28	319,28	328,37	242,88
p09-2-9	150	**347,9**	347,43	**347,9**	343,72	299,39
p10-2-9	199	**382,41**	378,32	379,81	376,35	362,3
p13-2-9	120	**239,57**	238,58	230,59	238,58	1225,12
p14-2-9	100	**303,17**	302,94	**303,17**	303,14	201,63

Table 7. (*continued*)

p15-2-9	150	**378,09**	**378,09**	**378,09**	376,16	322
p16-2-9	199	**394,05**	390,47	391,71	389,21	372,33
p06-3-9	50	**219,36**	218,96	218,96	218,67	153,06
p07-3-9	75	**274,8**	**274,8**	**274,8**	273,27	207,18
p08-3-9	100	**447,15**	444,82	433,38	444,87	538,03
p09-3-9	150	**500,17**	496,84	500,12	488,79	521,54
p10-3-9	199	**559,8**	549,83	551,44	533,15	608,92
p13-3-9	120	250,69	234,99	244,96	**283,15**	1955,02
p14-3-9	100	418,28	416,32	417,32	**419,63**	331,84
p15-3-9	150	**519,39**	517,18	512,83	513,09	534,16
p16-3-9	199	**567,24**	558,61	558,10	556,53	608,02
p06-4-9	50	**258,97**	**258,97**	254,47	**258,97**	213,45
p07-4-9	75	**344,35**	343,12	339,95	342,7	303,67
p08-4-9	100	536,64	**537,66**	536,13	535,26	902,31
p09-4-9	150	**639,72**	635,67	633,64	621,47	785,67
p10-4-9	199	**723,47**	710,59	719,13	684,68	904,46
p13-4-9	120	279,43	264,46	294,46	**295,77**	2016,09
p14-4-9	100	**537,24**	516,20	531,53	531,94	444,02
p15-4-9	150	653,22	**654,94**	652,58	651,14	803,79
p16-4-9	199	729,40	**731,14**	726,22	719,14	888,78
CPU(min)		10,3	2,83	8,54	9,94	

5 Conclusion

In the present work, we propose a set of basic and hybrid heuristic approaches based on the ILS heuristic. The basic heuristics combine ILS with seven neighborhood operators which results in seven basic ILS heuristics. The hybrid ILS heuristics use either LNS or RVND, or a combination of LNS and a neighborhood operator, or a combination of LNS and RVND. A simple heuristic using only LNS and RVND is also provided to highlight the importance of the ILS heuristic. In addition, two perturbation procedures are tested.

The proposed approaches are evaluated on a variant of the *Vehicle Routing Problem* called *Capacitated Profitable Tour Problem*. The obtained results show that the more ILS is hybridized the better are the results.

The best implemented approach in term of average results is compared with other approaches from the literature. The experimentations show that the proposed heuristic is able to provide competitive results for the *Capacitated Profitable Tour Problem*.

A future work may consists in evaluating the performance of the proposed hybrid heuristic on other variants of the *Vehicle Routing Problem*.

References

1. Archetti, C., Feillet, D., Hertz, A., Speranza, M.G.: The capacitated team orienteering and profitable tour problems. J. Oper. Res. Soc. **60**, 831–842 (2009)
2. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search. In: Glover, F., Kochenberger, G.A. (eds) Handbook of Metaheuristics, vol. 57. Springer, Boston (2003). https://doi.org/10.1007/0-306-48056-5_11
3. Martins, A.X., Duhamel, C., Mahey, P., Saldanha, R.R., de Souza, M.C.: Variable neighborhood descent with iterated local search for routing and wavelength assignment. Comput. Oper. Res. **39**, 2133–2141 (2012)
4. Martins, I.C., Pinheiro, R.G., Protti, F., Ochi, L.S.: A hybrid iterated local search and variable neighborhood descent heuristic applied to the cell formation problem. Expert Syst. Appl. **42**, 8947–8955 (2015)
5. Chen, P., Huang, H.k., Dong, X.Y.: Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. Expert Syst. Appl. **37**, 1620–1627 (2010)
6. Subramanian, A., Drummond, L.M.D.A., Bentes, C., Ochi, L.S., Farias, R.: A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. Comput. Oper. Res. **37**, 1899–1911 (2010)
7. Subramanian, A., Uchoa, E., Ochi, L.S.: A hybrid algorithm for a class of vehicle routing problems. Comput. Oper. Res. **40**, 2519–2531 (2013)
8. Assis, L.P., Maravilha, A.L., Vivas, A., Campelo, F., Ramírez, J.A.: Multiobjective vehicle routing problem with fixed delivery and optional collections. Optimization Letters **7**, 1419–1431 (2013)
9. Subramanian, A., Battarra, M.: An iterated local search algorithm for the travelling salesman problem with pickups and deliveries. J. Oper. Res. Soc. **64**, 402–409 (2013)
10. Hernández-Pérez, H., Rodríguez-Martín, I., Salazar-González, J.J.: A hybrid heuristic approach for the multi-commodity pickup-and-delivery traveling salesman problem. Eur. J. Oper. Res. **251**, 44–52 (2016)
11. Todosijević, R., Hanafi, S., Urošević, D., Jarboui, B., Gendron, B.: A general variable neighborhood search for the swap-body vehicle routing problem. Comput. Oper. Res. **78**, 468–479 (2017)
12. Erdoğan, G., Cordeau, J.F., Laporte, G.: The pickup and delivery traveling salesman problem with first-in-first-out loading. Comput. Oper. Res. **36**, 1800–1808 (2009)
13. Hernández-Pérez, H., Rodríguez-Martín, I., Salazar-González, J.J.: A hybrid grasp/vnd heuristic for the one-commodity pickup-and-delivery traveling salesman problem. Comput. Oper. Res. **36**, 1639–1645 (2009)
14. Rodríguez-Martín, I., Salazar-González, J.J.: A hybrid heuristic approach for the multi-commodity one-to-one pickup-and-delivery traveling salesman problem. J. Heuristics **18**, 849–867 (2012)
15. Gansterer, M., Küçüktepe, M., Hartl, R.F.: The multi-vehicle profitable pickup and delivery problem. OR Spectrum **39**, 303–319 (2017)
16. Sifaleras, A., Konstantaras, I.: Variable neighborhood descent heuristic for solving reverse logistics multi-item dynamic lot-sizing problems. Comput. Oper. Res. **78**, 385–392 (2017)
17. Samà, M., Corman, F., Pacciarelli, D., et al.: A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. Comput. Oper. Res. **78**, 480–499 (2017)

18. Hassannayebi, E., Zegordi, S.H.: Variable and adaptive neighbourhood search algorithms for rail rapid transit timetabling problem. Comput. Oper. Res. **78**, 439–453 (2017)
19. Sassi, O., Cherif-Khettaf, W.R., Oulamara, A.: Multi-start iterated local search for the mixed fleet vehicle routing problem with heterogenous electric vehicles. In: Ochoa, G., Chicano, F. (eds.) EvoCOP 2015. LNCS, vol. 9026, pp. 138–149. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16468-7_12
20. Cuervo, D.P., Goos, P., Sörensen, K., Arráiz, E.: An iterated local search algorithm for the vehicle routing problem with backhauls. Eur. J. Oper. Res. **237**, 454–464 (2014)
21. Silva, M.M., Subramanian, A., Ochi, L.S.: An iterated local search heuristic for the split delivery vehicle routing problem. Comput. Oper. Res. **53**, 234–249 (2015)
22. Morais, V.W., Mateus, G.R., Noronha, T.F.: Iterated local search heuristics for the vehicle routing problem with cross-docking. Expert Syst. Appl. **41**, 7495–7506 (2014)
23. Li, J., Pardalos, P.M., Sun, H., Pei, J., Zhang, Y.: Iterated local search embedded adaptive neighborhood selection approach for the multi-depot vehicle routing problem with simultaneous deliveries and pickups. Expert Syst. Appl. **42**, 3551–3561 (2015)
24. François, V., Arda, Y., Crama, Y., Laporte, G.: Large neighborhood search for multi-trip vehicle routing. Eur. J. Oper. Res. **255**, 422–441 (2016)
25. Grangier, P., Gendreau, M., Lehuédé, F., Rousseau, L.M.: A matheuristic based on large neighborhood search for the vehicle routing problem with cross-docking. Comput. Oper. Res. **84**, 116–126 (2017)
26. Akpinar, S.: Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem. Expert Syst. Appl. **61**, 28–38 (2016)
27. Dominguez, O., Guimarans, D., Juan, A.A., de la Nuez, I.: A biased-randomised large neighbourhood search for the two-dimensional vehicle routing problem with backhauls. Eur. J. Oper. Res. **255**, 442–462 (2016)
28. Canca, D., De-Los-Santos, A., Laporte, G., Mesa, J.A.: An adaptive neighborhood search metaheuristic for the integrated railway rapid transit network design and line planning problem. Comput. Oper. Res. **78**, 1–14 (2017)
29. Chentli, H., Ouafi, R., Cherif-Khettaf, W.R.: Behaviour of a hybrid ils heuristic on the capacitated profitable tour problem. In: Proceedings of the 7th International Conference on Operations Research and Enterprise Systems: ICORES, INSTICC, SciTePress, vol. 1, pp. 115–123 (2018)
30. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. **35**, 254–265 (1987)
31. Talbi, E.G.: Metaheuristics: from design to implementation (2009)
32. Pisinger, D., Ropke, S.: A general heuristic for vehicle routing problems. Comput. Oper. Res. **34**, 2403–2435 (2007)
33. Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. In: Christofides, N., Mingozzi, A., Toth, P., Sandi, C. (eds.) Combinatorial Optimization, pp. 315–338. Wiley, Chichester (1979)