Expert Systems With Applications 61 (2016) 28–38

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem

Sener Akpinar*

*Dokuz Eylul University, Faculty of Engineering, Department of Industrial Engineering, 35397 Izmir, Turkey*

ARTICLE INFO

*Article history:*
Received 12 March 2016
Revised 12 May 2016
Accepted 13 May 2016
Available online 13 May 2016

*Keywords:*
Capacitated vehicle routing problem
Hybrid meta-heuristic
Large neighbourhood search
Ant colony optimization

ABSTRACT

This paper presents a new hybrid algorithm that executes large neighbourhood search algorithm in combination with the solution construction mechanism of the ant colony optimization algorithm (LNS–ACO) for the capacitated vehicle routing problem (CVRP). The proposed hybrid LNS–ACO algorithm aims at enhancing the performance of the large neighbourhood search algorithm by providing a satisfactory level of diversification via the solution construction mechanism of the ant colony optimization algorithm. Therefore, LNS–ACO algorithm combines its solution improvement mechanism with a solution construction mechanism. The performance of the proposed algorithm is tested on a set of CVRP instances. The hybrid LNS–ACO algorithm is compared against two other LNS variants and some of the formerly developed methods in terms of solution quality. Computational results indicate that the proposed hybrid LNS–ACO algorithm has a satisfactory performance in solving CVRP instances.

## 1. Introduction

Optimization is the processes of systematically making a design, system, or decision as effective as possible through a set of logically connected rules, generally referring to an optimization algorithm. Developing efficient optimization algorithms is an exploratory research field, since numerous real world problems could be modelled as an optimization problem and they need to be solved to optimality or near-optimality in reasonable time limits. In this respect, algorithmic design literature has been growing continuously within two directions: developing new search strategies and improving the performance of the existing search procedures by some modifications or hybridization one search procedure with other search procedures.

Optimization algorithms can be classified into two groups as exact and approximation methods. Exact solution procedures depend on full enumeration and guarantee the optimal solution of the problem on hand; however, they require excessive amount of time to identify the optimal solution as the problem size gets larger. Therefore, approximation algorithms are used widely than the exact solution procedures. From the optimization point of view, the key point is to select the appropriate optimization method for the problem on hand. Approximation algorithms generally start with a single solution or a pre-defined number of solutions while trying to identify the optimal solution for an optimization problem. They usually use some specific rules with the aim of guiding the algorithm towards the promising regions of solution space of the problem. These rules, namely heuristics, often provide some neighbourhood moves that change the current solution in order to explore the solution space of the problem. Specifying the heuristics to be used and connecting them within a logical manner are the key issues while developing efficient approximation algorithms.

The approximation algorithms are generally called as meta-heuristics, and they have the capability to provide satisfactory results for different types of optimization problems in reasonable time limits. However, meta-heuristic algorithms could also have some limitations as the premature convergence, which may cause the algorithm to trap in local optima or to stagnate and therefore it is a challenging problem for the meta-heuristics approaches. The best way to improve the effectiveness of a meta-heuristic algorithm and thus to overcome such limitations is to develop hybrid approaches, which generally combine superiorities of different search procedures. The reader can be suggested to see the review papers of Blum, Puchinger, Raidl, and Roli (2011), Crainic and Toulouse (2003), Preux and Talbi (1999), Raidl (2006), Talbi (2002) and Ting, Yang, Cheng, and Huang (2015) for detailed discussions about the hybrid meta-heuristics.

In this present paper, we propose a hybrid large neighbourhood search algorithm for the capacitated vehicle routing problem (CVRP). The proposed hybrid algorithm is a combination of large neighbourhood search (LNS) algorithm (Shaw, 1998) and ant colony optimization (ACO) algorithm (Dorigo, Maniezzo, Colorni, & Maniezzo, 1991). To the best of our knowledge, Elhassania, Jaouad,

* Corresponding author. Fax: +90 232 301 76 08.
*E-mail address:* sener.akpinar@deu.edu.tr

http://dx.doi.org/10.1016/j.eswa.2016.05.023
0957-4174/© 2016 Elsevier Ltd. All rights reserved.

and Ahmed (2013) firstly introduced the concept of hybridizing the LNS and ACO algorithms for a similar vehicle routing problem discussed in this paper, however search characteristics of their hybrid ACOLNS algorithm differ from our hybrid LNS-ACO algorithm. The similarities and differences between these two hybrid algorithms will be discussed in Section 4. The LNS algorithm uses two basic operators, destruction of a solution by a removal heuristic and reparation of the destroyed solution by an insertion heuristic. A removal heuristic generates an infeasible solution from the current solution and then the insertion heuristic rebuilds this solution to generate another feasible solution (Pisinger & Ropke, 2010). After that, LNS algorithm decides whether to accept or not this newly generated feasible solution as the new current solution via the guidance of an acceptance criterion (Shaw, 1998), (Schrimpf, Schneider, Stamm-Wilbrandt, & Dueck, 2000), (Ropke & Pisinger, 2006). In this paper, the proposed hybrid algorithm accepts only the improving solutions; otherwise, the algorithm generates a new solution via the solution construction mechanism of ACO with the guidance of the information derived from the incumbent solution. By doing so, the proposed algorithm achieves a solution construction characteristic besides the improvement characteristic of LNS algorithm, and therefore, the algorithm gains the ability to utilize the positive feedback mechanism of ACO. For detailed information about the variants and extensions of the LNS algorithm, the reader can be suggested to see the paper of Pisinger and Ropke (2010).

The remainder of the paper is organized as follows. The definition of the capacitated vehicle routing problem is given in Section 2. Basic steps of the LNS algorithm are given in Section 3. The proposed hybrid LNS–ACO algorithm is presented in Section 4. Computational study is presented in Section 5. Conclusions are given in Section 6.

## 2. Capacitated vehicle routing problem

The capacitated vehicle routing problem (CVRP) is the classical version of the routing problems. This problem consists of a single depot and a set of customers to be served from the depot via a set of $K$ homogeneous capacitated vehicles. The objective of this problem is to determine the routes that minimize the total cost, that is, the problem is about the assignment of customers to vehicles and determining customer visit sequences for each route with the aim of minimizing the total travelled distance by the vehicles. Given a set of $N$ customers and a single depot, the CVRP can be defined via a graph as $G = (N,\ E)$, where $N = \{0, \ldots, n\}$ is the node set and $E = \{(i,\ j) :\ i,\ j \in N\}$ is the edge set. Node 0 represents the depot that is the start and end node of the vehicles for their trips. The other nodes represent the customers having a known demand $d_i$ and each customer must be served by exactly one vehicle. The travel distance from node $i$ to node $j$ is defined by $d_{ij} > 0$ and each vehicle has a unique capacity of $Q_k$. The total demand of the customers assigned to a route must not exceed the unique vehicle's capacity of $Q_k$. In accordance with these explanations, CVRP can be formulated as given below, where $X_{ij}^k$ equals to 1 if vehicle $k$ travels from node $i$ to node $j$ and 0 otherwise (Lin, Lee, Ying, & Lee, 2009).

$$Min\ \sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=1}^{K} d_{ij}X_{ij}^k \tag{1}$$

Subject to

$$\sum_{k=1}^{K}\sum_{i=0}^{N} X_{ij}^k = 1 \quad j \in \{1, \ldots, N\} : i \neq j \tag{2}$$

$$\sum_{k=1}^{K}\sum_{j=0}^{N} X_{ij}^k = 1 \quad i \in \{1, \ldots, N\} : i \neq j \tag{3}$$

$$\sum_{i=0}^{N}\sum_{j=0}^{N} X_{ij}^k d_i \leq Q_k \quad k \in \{1, \ldots, K\} \tag{4}$$

$$\sum_{j=1}^{N} X_{ij}^k = \sum_{j=1}^{N} X_{ji}^k \leq 1 \quad \text{for } i = 0 \text{ and } k \in \{1, \ldots, K\} \tag{5}$$

$$\sum_{k=1}^{K}\sum_{j=1}^{N} X_{ij}^k \leq K \quad \text{for } i = 0 \tag{6}$$

Objective function (1) minimizes the total distance travelled by the vehicles. Constraint sets (2) and (3) guarantee that each customer is served by exactly one vehicle. Constraint set (4) ensures that the total demand of the customers assigned to a route does not exceed the vehicle capacity. Constraint set (5) indicates that the depot is the start and end node for the trips of each vehicle. Constraint set (6) guarantees that there are maximum $K$ routes for serving the customers.

For comprehensive information about the variants of the vehicle routing problem and their solution procedures, the reader can be suggested to see the first and second editions of the book edited by Toth and Vigo (2002, 2014), and the survey papers by Eksioglu et al. (2009), Laporte (2009) and Braekers, Ramaekers, and Van Nieuwenhuyse (2016). Developing effective solution procedures for the CVRP remains as a challenging research field, however a large number of solution methods have been proposed in the literature during the last 50 years (Jin, Crainic, & Løkketangen, 2014) and CVRP is still far from being satisfactorily solved (Semet, Toth, & Vigo, 2014). CVRP is the basic problem in the literature on vehicle routing, which is also a challenging research field, and thereby any solution procedure developed for the CVRP could be adapted to solve any variant of this problem. For that reason, we have the motivation to solve CVRP instances effectively with an improved search strategy, hybrid LNS-ACO algorithm, within the scope of this paper. Additionally, our proposed hybrid algorithm can be extended to have the capability of solving other variants of CVRP within the context of the succeeding studies.

## 3. Large neighbourhood search algorithm

Shaw (1998) initially proposed the LNS algorithm for solving vehicle routing problem with time windows (VRPTW). LNS algorithm uses a neighbourhood generation mechanism to explore the solution space of the optimization problem. The neighbourhood generation mechanism of the LNS algorithm depends on two successive operators: destruction of a solution by a removal heuristic and reparation of the destroyed solution by an insertion heuristic. A removal heuristic removes some components of the current solution by considering a criterion, while the insertion heuristic fixes the destroyed solution by reinserting the removed parts via a greedy rule. After that, LNS algorithm evaluates the newly generated solution via an acceptance function in order to decide to accept or reject this solution as the new current solution. LNS algorithm executes these steps successively until the stopping condition is met as can be seen from Fig. 1.

Ropke and Pisinger (2006) successfully modified the basic LNS algorithm by using different types of removal and insertion heuristics during the same search instead of using one method for removal and one method for insertion. They called the new version of the LNS algorithm as adaptive large neighbourhood search (ALNS) algorithm. In addition, there are many successful implementations of the ALNS algorithm in literature such as the papers of Adulyasak, Cordeau, and Jans (2012), Belo-Filho, Amorim, and Almada-Lobo (2015), Demir et al. (2012), Laporte, Musmanno, and Vocaturo (2010), Luo, Qin, Zhang, and Lim (2016),

```
Function $LNS(s \in \{solutions\}, q \in \mathbb{N})$
    solution $s_{best} = s$
    repeat
        $s' = s$
        remove $q$ requests from $s'$
        reinsert removed requests into $s'$
        if $(f(s') < f(s_{best}))$ then
            $s_{best} = s'$
        if accept$(s', s)$ then
            $s = s'$
    until stop-criterion met
return $s_{best}$
```

**Fig. 1.** Main steps of LNS algorithm (Ropke & Pisinger, 2006).

Masson, Lehuédé, and Péton (2013), Muller, Spoorendonk, and Pisinger (2012) and Rifai et al. (2016).

Moreover, from our standpoint, LNS algorithm could be modified by considering the acceptance function. In the original LNS algorithm, only the improving solutions are accepted as current solution, while later applications have used an acceptance function similar to the solution acceptance criterion of the simulated annealing algorithm (Pisinger & Ropke, 2010). In this paper, the proposed version of LNS only accepts the improving solutions instead of using an acceptance function and utilizes ACO when no improvement is achieved for the current solution. Instead of going back to a formerly discovered solution, we prefer constructing a new solution by the guidance of the positive feedback mechanism of ACO and setting this new solution as the current solution. While constructing a new solution, we use the information derived from the best solution that the algorithm identified until the current iteration. Hereby, the algorithm is able to construct a new solution, which has almost the same quality level of the current best solution and placed in another region of the search space. Therefore, the algorithm may reach a satisfactory level of diversification. The proposed hybrid LNS–ACO is explained in the following section more detailed.

## 4. Hybrid large neighbourhood search algorithm

This section introduces the proposed hybrid LNS–ACO algorithm in depth. The rationale why we attempt to develop this hybrid algorithm is the belief that the hybrid algorithms generally reveal synergy (Blum et al., 2011). Main steps of the proposed hybrid LNS–ACO algorithm are depicted in Fig. 2.

The proposed hybrid LNS–ACO algorithm starts with a randomly generated initial feasible solution and tries to improve this solution by executing removal and insertion heuristics successively until the stopping condition is met. The original LNS algorithm and its variants may be classified as members of the improving type of meta-heuristic algorithms as distinct from the proposed hybrid LNS–ACO algorithm, since it has both the solution improvement and construction mechanisms. Variants of the LNS algorithm use a solution acceptance criterion in order to specify the current solution and continue their search by exploring the neighbour area of this current solution. In this current version of the LNS algorithm, we allowed the algorithm to accept only the improving solutions in comparison to the current solution and to construct a new solution to set it as the current solution for the next iteration if no improvement is achieved. Therefore, the hybrid LNS–ACO algorithm utilizes both the solution improvement and construction mechanisms throughout its execution.

As mentioned before, Elhassania et al. (2013) hybridized the ACO and LNS algorithms for the dynamic vehicle routing problem, which is an extension of CVRP. They incorporated the LNS algorithm into ACO with the aim of improving the performance of ACO; however, our hybrid algorithm has the aim of improving the performance of LNS. Their hybrid algorithm firstly generates some solutions via ACO and then it tries to improve these solutions via LNS and this characteristic makes their algorithm a population based search strategy and a sequential type of hybrid algorithm. On the other hand, our hybrid algorithm tries to improve the quality of a single solution and it is a parallel type of hybrid algorithm, since the hybrid LNS-ACO algorithm uses solution construction mechanism of ACO as an operator into LNS. Because of these issues, these two algorithms have different search strategies; however, both of them have solution construction and improvement characteristics due to ACO and LNS respectively. The following sub-sections introduce the main steps of the proposed hybrid LNS–ACO algorithm in depth.

### *4.1. Removal heuristic*

The proposed hybrid LNS-ACO uses the worst removal heuristic (Ropke & Pisinger, 2006) in order to destroy the current solution. This heuristic was based on the idea of removing customers inducing high cost to serve it in its current position and inserting them into another position in the solution *s*. Within this respect, this heuristic calculates the cost of serving a customer *i* in its

```
Function $LNS(s \in \{solutions\}, q \in \mathbb{N})$
        solution $s_{best} = s$
        repeat
                $s' = s$
                remove $q$ requests from $s'$
                reinsert removed requests into $s'$
                if $(f(s') < f(s))$ then
                        $s = s'$
                else if $(f(s') \geq f(s))$ then
                        $s = ACO(s_{best})$
                if $(f(s) < f(s_{best}))$ then
                        $s_{best} = s$
        until stop-criterion met
return $s_{best}$
```

**Fig. 2.** Main steps of the hybrid LNS–ACO algorithm.

```
Function $WorstRemoval(s \in \{solutions\}, q \in \mathbb{N}, p \in \mathbb{R}_+)$
    while $q > 0$ do
        Array: $L$= All planned request $i$ sorted by descending $cost(i,s)$
        choose a random number $y$ in the interval $[0,1)$
        request: $r = L[y^p|L|]$
        remove $r$ from solution $s$
        $q = q - 1$
    end while
```

**Fig. 3.** Worst removal heuristic (Ropke & Pisinger, 2006).

current position in the solution $s$ as $cost(i,\ s) = f(s) - f_{-i}(s)$ where $f(s)$ is the cost of the complete solution and $f_{-i}(s)$ is the cost of the solution without customer $i$. The algorithm of the worst removal heuristic is depicted in Fig. 3.

The worst removal heuristic randomizes its destruction via the parameter $p$, which avoids removing same customers again and again. This heuristic also tries to determine the misplaced customers in a solution and therefore determines the customers required to move to another position in a solution. In the adaptive versions of the LNS algorithm, various removal heuristics were developed, but there is no information about the best removal heuristic in literature. Therefore, we select the worst removal heuristic, the more reasonable one for us, since our focus is not selecting the best removal heuristic within the context of this study.

### 4.2. Insertion heuristic

In this paper, we select the regret–2 heuristic (Ropke & Pisinger, 2006), which was developed to improve the basic greedy heuristic, as the solution reparation operator. Basic greedy heuristic tries to insert the removed customers at such positions as the least objective value increment is obtained, however it has an obvious shortage as postponing the placement of the customers with higher cost increments to the last iterations and therefore decreasing the number of positions for inserting these customers. In order to overcome this disadvantage, regret heuristic uses a variable $x_{im} \in \{1, \ldots, K\}$, indicates the route for which the customer $i$ has the $m$th lowest insertion cost. In the regret–2 heuristic, a regret value is defined as $r_i^* = \Delta f_{i,\ x_{i2}} - \Delta f_{i,\ x_{i1}}$, where $\Delta f_{i,\ x_{im}}$ is the change in objective function value incurred by inserting customer $i$ into the $m$th best position. The regret value defines the difference in the cost of

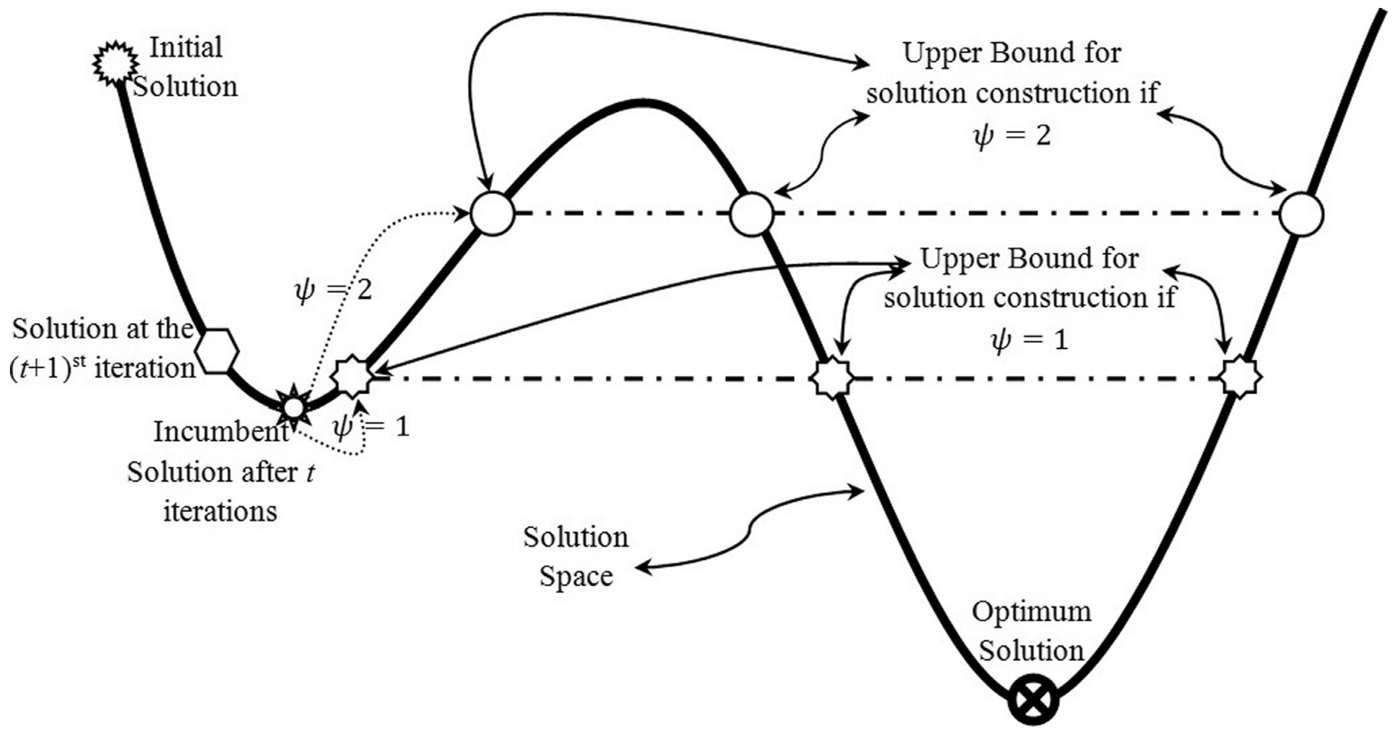
Initial Solution
Upper Bound for solution construction if $\psi = 2$
$\psi = 2$
Solution at the $(t+1)^{st}$ iteration
Upper Bound for solution construction if $\psi = 1$
Incumbent Solution after $t$ iterations
$\psi = 1$
Solution Space
Optimum Solution

**Fig. 4.** Solution construction mechanism.

inserting the customer in its best position and its second-best position. Regret–2 heuristic chooses customer $i$ that maximizes $\underset{i \in U}{max}\ r_i^*$ in each iteration, where $U$ is the set of removed customers by the worst removal heuristic. Here we must state again that our focus is not selecting the best insertion heuristic, since we mainly focus on the solution acceptance/generation heuristic within this context of this study.

### 4.3. Solution acceptance/generation heuristic

As stated before, original LNS algorithm accepts only improving solutions as the current solution, however later variants of LNS algorithm specify the current solution via a solution acceptance function. To our point of view, LNS algorithm could be modified by using a different mechanism to identify the current solution in place of this solution acceptance function and therefore to form a novel hybrid LNS algorithm. Within this respect, the proposed hybrid algorithm accepts only the improving solutions in comparison to the current solution and constructs a new solution to set it as the current solution for the next iteration via the solution construction mechanism of ACO if the current solution could not be improved. The solution construction mechanism Eq. (7) used in this study is a modification of the one that originally developed for assembly line balancing problems (Akpinar, Bayhan, & Baykasoglu, 2013; Vilarinho & Simaria, 2006)

$$j = \begin{cases} J_1 = \underset{j \in A_i}{argmax}\{[\tau_{(i,j)}]^{\alpha} \times [\eta_{ij}]^{\beta}\} & \text{if } r \le r_1 \ \ (exploitation) \\ J_2 : p(i, j_2) = \frac{[\tau_{(i,j_2)}]^{\alpha} \times [\eta_{ij_2}]^{\beta}}{\sum_{j \in A_i}([\tau_{(i,j)}]^{\alpha} \times [\eta_{ij_2}]^{\beta})} & \text{if } r_1 < r \le r_1 + r_2 \\ & (biased\ exploration) \\ J_3 : random\ selection\ of\ \ j \in A_i & \text{if } r_1 + r_2 < r \le r_1 + r_2 + r_3 \end{cases} \tag{7}$$

where $r \in [0, 1]$ is a random number, $r_1$, $r_2$ and $r_3$ are user defined parameters such that $0 \le r_1, r_2, r_3 \le 1$ and $r_1 + r_2 + r_3 = 1$, $\tau_{(i,\ j)}$ is the pheromone trail intensity in the path selecting node $j$ after selecting node $i$. $\eta_{ij}$ is the heuristic information between nodes $i$ and $j$ (e.g. the priority rule value between nodes $i$ and $j$). $A_i$ is the set of available node after the selection of node $i$, and $\alpha$ and $\beta$ are parameters that determine the relative importance of pheromone intensity versus heuristic information. As the heuristic information we used the saving value (Clarke & Wright, 1964) between nodes which is calculated as $S_{ij} = d_{i0} + d_{0j} - d_{ij}\ \forall\ i, j \in \{1, \ldots, N\} : i \neq j$.

The key issue of constructing a new solution via the solution construction mechanism is the guidance of the information derived from the incumbent solution identified until the current iteration. Within this respect, we determined the maximum tour length of the incumbent solution and multiply this by a user defined parameter $\psi \in \{1, \ldots, 2\}$, and set the obtained value as the desired maximum tour length for each vehicle while constructing a new solution. Thus, the proposed hybrid LNS–ACO algorithm realizes the positive feedback mechanism and has the ability to construct a new solution, which has almost the same quality level of the incumbent solution. Another advantage of constructing a new solution by this way is to prevent the algorithm to execute redundant iterations during its search as can be obviously seen in Fig. 4. Additionally, the solution construction heuristic constructs several routes in parallel, that is, heuristic starts with $K$ routes initially stationed at the depot and there are $N$ customers that are assigned one by one to these routes sequentially (Pang, 2011). The heuristic holds one route after another and tries to find the best-unassigned customer for the current route via Eq. (7) until all customers are assigned.

The proposed hybrid LNS–ACO algorithm starts its search via a randomly generated initial feasible solution. As stated before, the algorithm tries to improve this solution via removal and insertion heuristics. After each iteration, algorithm decides whether to accept the new solution as the current solution or not. Hybrid LNS–ACO algorithm accepts only the improving solutions as the current solution for the next iteration or it constructs a new solution if the current solution could not be improved. During the solution construction process, the incumbent solution of the $t^{th}$ iteration and a user defined parameter $\psi$ guide the algorithm as can be seen from Fig. 4. After executing removal and insertion heuristics on the solution of the $t^{th}$ iteration, if the solution of the $(t+1)^{st}$ iteration is getting worse then the algorithm constructs another solution instead of going back to the solution of the $t^{th}$ iteration. Thus, it is aimed that to achieve a satisfactory level of diversification in the proposed hybrid LNS–ACO algorithm as stated before. Via the guidance of incumbent solution of the $t^{th}$ iteration and the user defined parameter $\psi$, the algorithm determines an upper bound and tries to generate a new solution by considering this upper bound. The solution construction heuristic may make several attempts until returning a new solution having an objective value lower than this upper bound. Hence, the computational time of the algorithm may increase due to these attempts. Therefore, the abovementioned upper bound must be carefully determined when the algorithm needs to construct a new solution. Here we must state that the parameter $\psi$ is not a necessity, but using such a parameter may significantly affect the hybrid LNS–ACO algorithm's performance. Therefore, we will determine the effectiveness of this parameter on the algorithm's performance through some analysis given in Section 5.1. The performance of the proposed hybrid LNS–ACO algorithm will also be tested on a set of benchmark problems within the endeavour of clarifying the effect of solution construction mechanism on the algorithm's performance.

## 5. Computational study

This section presents a computational study, in three parts, to assess the performance of the proposed hybrid LNS-ACO algorithm. The first part is about to analyse the effect of the parameter $\psi$ over the performance of the proposed hybrid LNS–ACO algorithm in terms of computational time and solution quality. In the second part, we compare the hybrid LNS–ACO algorithm with large neighbourhood search algorithm by accepting only the improving solutions (LNSi) and large neighbourhood search algorithm by using a solution acceptance criterion (LNSa) similar to (Ropke & Pisinger, 2006). For LNS–ACO, LNSi and LNSa algorithms, the abovementioned removal and insertion heuristics remain same in order to emphasize the effectiveness of the solution construction mechanism based on the one that of ant colony optimization algorithm. We also compare the obtained results by these three algorithms to Stanojević, Stanojević, and Vujošević (2013) results obtained by their SC-ESA algorithm (Set-covering based extended savings algorithm) and the best-known solutions (BKS) for the instances of the benchmark set directly in the second part of the computational study. The third part numerically compares the hybrid LNS-ACO algorithm against some formerly developed CVRP methods. For the numerical experiments in the first and second parts, the benchmark set (Stanojević et al., 2013) composed of seventy-three instances (A, B and P) from (Augerat et al., 1995), eight instances (E) from (Christofides and Eilon, 1969) and seven instances (CMT) from (Christofides, Mingozzi, & Toth, 1979) is used. In the third part of the computational study, 14 instances (CMT) from (Christofides et al., 1979) are used. All the problem instances were downloaded from the site <http://branchandcut.org>. These three algorithms (LNS–ACO, $LNS_i$ and $LNS_a$) have been coded in MATLAB 7.9, and then the results were obtained by running the coded algorithms on a Core(TM) i7-2640 CPU 2.80 GHz personal computer.

It is obvious that the hybrid LNS-ACO algorithm requires some parameters to tune. The parameters with their values are presented in Table 1 except the parameter $\psi$, because its value will be determined through a set of experiments in the following subsection.

The values for the parameters $r1$, $r2$, $r3$, $\alpha$ and $\beta$ were selected after some preliminary experiments, while the value of the parameter $p$ was taken from (Ropke & Pisinger, 2006). Additionally, the hybrid LNS-ACO algorithm randomly determines the parameter $q$ that defines the number of nodes removed from the solution at each iteration in the same way done by Ropke and Pisinger (2006) as presented in Table 1. On the other hand, for the $LNS_a$, we choose initial temperature $T_{start}$ as 100 and the cooling ratio $c$ as 0.99975.

**Table 1**
Parameter set.

| Parameter | $r1$ | $r2$ | $r3$ | $\alpha$ | $\beta$ | $q$ | $p$ |
|---|---|---|---|---|---|---|---|
| Value | 0.5 | 0.3 | 0.2 | 1 | 1 | $4 \le randi() \le \min(100,\ 0.4n)$ | 6 |

### 5.1. Analyses on the effect of the parameter $\psi$

As stated in Section 4.3, the parameter $\psi$ may have a significant effect over the performance of the algorithm. Therefore, its value must be determined carefully. In this subsection, we analyse the algorithm's behaviour with the parameter $\psi \in [1, 2]$ and without the parameter $\psi$. Within this context, we defined five levels for this parameter as 1, 1.25, 1.5, 1.75 and 2, and we run the algorithm via the parameter $\psi \in \{1, 1.25, 1.5, 1.75, 2\}$ and without the parameter $\psi$ on a sub-set of benchmark problems (10 problems: 2 problems from each A, B, P, E and CMT sets) for 1000 iterations. Results depicted in Table 2 are the best ones for five independent runs and the effect of this parameter is analysed in terms of solution quality and computational time in seconds.

As can be obviously seen from Table 2, using the parameter $\psi$ significantly affects the performance of the proposed hybrid LNS–ACO algorithm in terms of solution quality; however, it increases the computational time of the algorithm. When the algorithm calls the solution generation heuristic during its search, the parameter $\psi$ acts as a tightness factor and therefore defines an upper bound for the solution construction heuristic. This situation restricts the search area of the solution construction heuristic, so the algorithm tries to construct a feasible solution within this restricted area and the heuristic success rate decreases due to this restriction. That is to say, the solution construction heuristic may do several attempts to construct a feasible solution and therefore the CPU time of the algorithm may increase due to these attempts as can be clearly seen from Fig. 5.

For each problem the best solutions were obtained by using different $\psi$ values and the relative deviations $\%\text{Dev} = ((Best - BKS)/BKS) * 100$ were computed and visualized in Fig. 6 in order to emphasize the effect of the parameter $\psi$. When the balance between the solution quality and CPU time (see Figs. 5 and 6) is considered, it is reasonable to set the value of the parameter $\psi$ as 1.5 for the performance evaluation test of the hybrid LNS–ACO algorithm over the test problems.

### 5.2. Performance evaluation of the hybrid LNS–ACO algorithm

As stated above, the performance of the hybrid LNS–ACO algorithm is tested against the LNSi, LNSa and the SC–ESA algorithm (Stanojević et al., 2013) over 88 best-known capacitated vehicle routing problems. The BKS values for all the test problems were taken from (Stanojević et al., 2013) and the costs of arcs for all the test problems have been calculated as Euclidean distances. The comparisons have been done by considering the relative percentage deviations $\%\text{Dev} = ((Best - BKS)/BKS) * 100$ from the BKS values and the obtained results are presented in Tables 3–7 for the problem sets A, B, P, E and CMT respectively. The Best and Worst values for %Dev, and StdDev values for algorithms' results, presented in Tables 3–7 have been obtained by running the algorithms 30 times independently for each instance of the problem set. The

**Table 2**
Results for preliminary experiments done for determining the value of $\psi$.

| Problem | BKS | Results without $\psi$ | | Results with $\psi$ | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $\psi = 1$ | | $\psi = 1.25$ | | $\psi = 1.5$ | | $\psi = 1.75$ | | $\psi = 2$ | |
| | | Best | CPU | Best | CPU | Best | CPU | Best | CPU | Best | CPU | Best | CPU |
| P-n16-k8 | 450 | 450 | 40.28 | 450 | 63.71 | 450 | 57.08 | 450 | 53.35 | 450 | 49.15 | 450 | 43.53 |
| P-n23-k8 | 529 | 529 | 58.09 | 529 | 105.37 | 529 | 97.27 | 529 | 85.04 | 529 | 74.49 | 529 | 63.38 |
| B-n35-k5 | 955 | 993 | 144.13 | 958 | 197.51 | 958 | 184.88 | 958 | 172.97 | 961 | 167.82 | 973 | 157.05 |
| B-n45-k5 | 751 | 792 | 193.19 | 751 | 280.73 | 751 | 268.91 | 751 | 241.04 | 764 | 230.53 | 766 | 212.54 |
| A-n55-k9 | 1073 | 1153 | 437.82 | 1091 | 550.99 | 1091 | 533.57 | 1089 | 492.09 | 1094 | 460.43 | 1098 | 438.04 |
| A-n65-k9 | 1174 | 1206 | 707.97 | 1183 | 840.31 | 1185 | 810.13 | 1183 | 788.04 | 1187 | 759.72 | 1191 | 728.35 |
| E-n76-k8 | 735 | 789 | 862.02 | 764 | 1104.19 | 764 | 1043.29 | 764 | 973.85 | 770 | 921.26 | 770 | 885.44 |
| E-n76-k14 | 1021 | 1103 | 980.70 | 1032 | 1228.64 | 1034 | 1199.93 | 1034 | 1167.31 | 1059 | 1143.88 | 1059 | 1089.17 |
| CMT-n101-k8 | 817 | 886 | 1900.76 | 847 | 2150.57 | 847 | 2087.32 | 847 | 2030.48 | 861 | 1992.28 | 866 | 1955.20 |
| CMT-n121-k7 | 1034 | 1107 | 2583.34 | 1064 | 3111.79 | 1064 | 2982.97 | 1064 | 2893.16 | 1072 | 2848.91 | 1072 | 2695.45 |

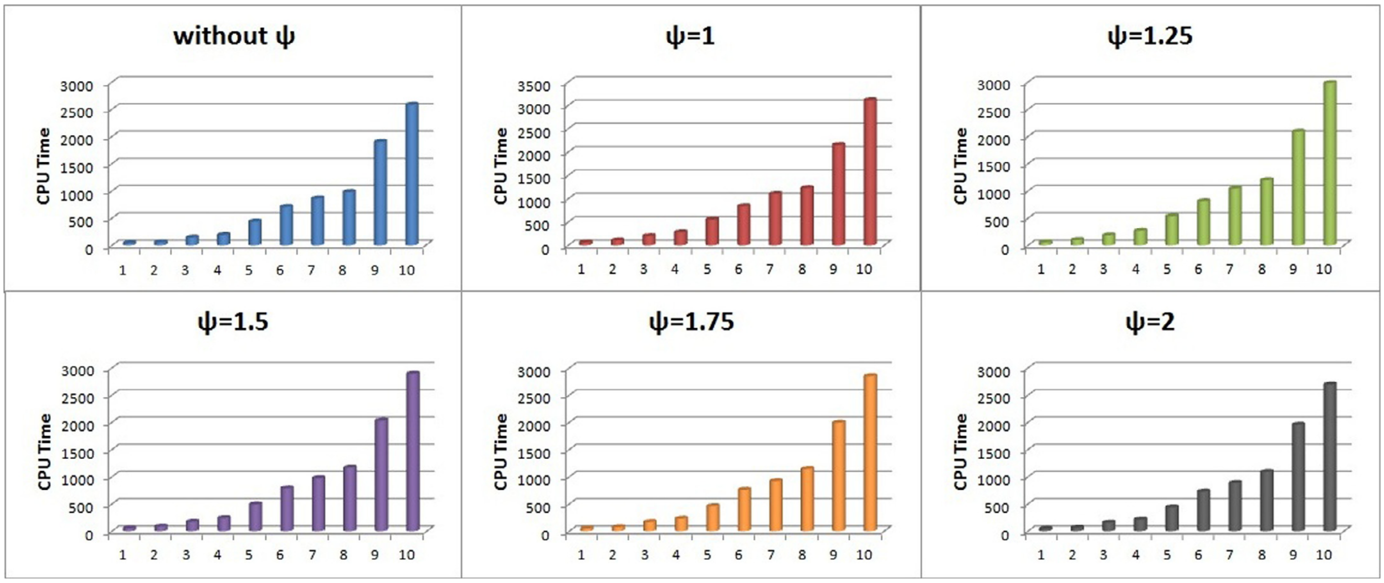

**Fig. 5.** CPU time variations according to different $\psi$ values.

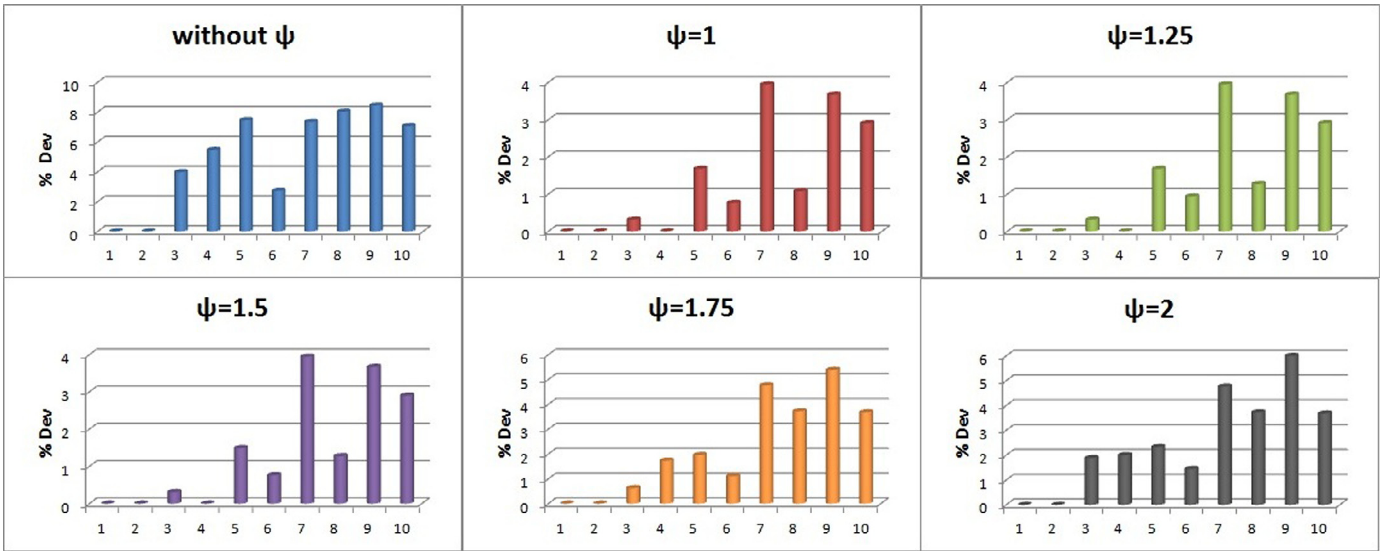

**Fig. 6.** Percentage deviations of best results from the best-known solutions according to different $\psi$ values.

three LNS variants return their best solutions when a predefined number of iterations (5000 iterations) are achieved.

From the observations of Tables 3–7, we can conclude that the proposed hybrid LNS–ACO algorithm outperforms other variants of LNS algorithms, LNSi and LNSa, in terms of solution quality. However, LNS–ACO has larger CPU times than the other algorithms due to the several attempts of the solution construction heuristic of the LNS–ACO algorithm to generate feasible solutions when required. Moreover, LNS–ACO algorithm was able to produce competitive results in comparison to SC–ESA algorithm, since LNS–ACO algorithm

**Table 3**
Computational results for the problem set A.

| Problem | BKS | SC-ESA | LNSi | | | | LNSa | | | | LNS-ACO | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | % Dev | % Dev | | StdDev | CPU | % Dev | | StdDev | CPU | % Dev | | StdDev | CPU |
| | | | Worst | Best | | | Worst | Best | | | Worst | Best | | |
| A-n32-k5 | 784 | 0.00 | 0.00 | 0.00 | 0.00 | 834.81 | 0.00 | 0.00 | 0.00 | 839.09 | 0.00 | 0.00 | 0.00 | 856.21 |
| A-n33-k5 | 661 | 0.00 | 0.61 | 0.00 | 1.36 | 877.56 | 0.00 | 0.00 | 0.00 | 882.06 | 0.00 | 0.00 | 0.00 | 900.06 |
| A-n33-k6 | 742 | 0.00 | 0.54 | 0.00 | 1.12 | 924.16 | 0.00 | 0.00 | 0.00 | 928.90 | 0.00 | 0.00 | 0.00 | 947.86 |
| A-n34-k5 | 778 | 0.00 | 1.54 | 0.77 | 1.78 | 886.31 | 0.00 | 0.00 | 0.00 | 890.86 | 0.00 | 0.00 | 0.00 | 909.04 |
| A-n36-k5 | 799 | 0.00 | 1.13 | 0.75 | 0.70 | 1029.21 | 0.75 | 0.00 | 1.82 | 1034.49 | 0.00 | 0.00 | 0.00 | 1055.60 |
| A-n37-k5 | 669 | 0.00 | 2.09 | 1.20 | 1.79 | 1076.11 | 0.60 | 0.00 | 1.31 | 1081.63 | 0.00 | 0.00 | 0.00 | 1103.70 |
| A-n37-k6 | 949 | 0.00 | 1.26 | 0.42 | 2.12 | 1085.57 | 0.32 | 0.00 | 0.86 | 1091.13 | 0.00 | 0.00 | 0.00 | 1113.40 |
| A-n38-k5 | 730 | 0.00 | 2.33 | 1.37 | 1.98 | 1110.72 | 0.82 | 0.00 | 1.69 | 1116.42 | 0.00 | 0.00 | 0.00 | 1139.20 |
| A-n39-k5 | 822 | 0.00 | 2.55 | 1.70 | 1.99 | 1172.73 | 0.61 | 0.00 | 1.40 | 1178.74 | 0.00 | 0.00 | 0.00 | 1202.80 |
| A-n39-k6 | 831 | 0.00 | 2.17 | 1.44 | 1.51 | 1234.35 | 0.96 | 0.00 | 2.50 | 1240.68 | 0.24 | 0.00 | 0.67 | 1266.00 |
| A-n44-k6 | 937 | 0.00 | 2.67 | 1.49 | 3.19 | 1528.61 | 1.49 | 0.32 | 3.21 | 1536.44 | 0.53 | 0.00 | 1.53 | 1567.80 |
| An-45-k6 | 944 | 0.00 | 4.34 | 3.07 | 3.71 | 1684.90 | 1.69 | 0.74 | 3.05 | 1693.54 | 2.01 | 1.48 | 1.61 | 1728.10 |
| A-n45-k7 | 1146 | 0.00 | 3.05 | 2.36 | 2.41 | 1698.16 | 0.79 | 0.35 | 1.73 | 1706.87 | 0.52 | 0.00 | 1.64 | 1741.70 |
| A-n46-k7 | 914 | 0.00 | 3.17 | 1.75 | 3.21 | 1759.39 | 1.09 | 0.44 | 1.65 | 1768.41 | 0.33 | 0.00 | 0.84 | 1804.50 |
| A-n48-k7 | 1073 | 1.03 | 3.17 | 2.24 | 2.50 | 1929.14 | 2.14 | 1.58 | 1.65 | 1939.03 | 1.03 | 1.03 | 0.00 | 1978.60 |
| A-n53-k7 | 1010 | 0.10 | 3.56 | 2.38 | 3.65 | 2265.61 | 0.99 | 0.50 | 1.37 | 2277.23 | 0.69 | 0.00 | 2.01 | 2323.70 |
| A-n54-k7 | 1167 | 0.09 | 2.31 | 1.71 | 1.92 | 2434.97 | 1.20 | 0.51 | 1.97 | 2447.45 | 0.60 | 0.00 | 1.94 | 2497.40 |
| A-n55-k9 | 1073 | 0.00 | 3.08 | 2.05 | 3.50 | 2701.73 | 1.58 | 0.37 | 3.59 | 2715.58 | 0.09 | 0.00 | 0.28 | 2771.00 |
| A-n60-k9 | 1354 | 0.07 | 2.81 | 1.92 | 3.20 | 3261.77 | 1.11 | 0.44 | 2.61 | 3278.49 | 0.37 | 0.00 | 1.28 | 3345.40 |
| A-n61-k9 | 1034 | 0.00 | 6.19 | 4.93 | 3.52 | 3271.81 | 4.16 | 3.68 | 1.43 | 3288.59 | 4.06 | 3.19 | 2.74 | 3355.70 |
| A-n62-k8 | 1288 | 0.78 | 4.58 | 3.34 | 5.63 | 3279.22 | 3.11 | 2.17 | 3.65 | 3296.03 | 1.94 | 1.55 | 1.36 | 3363.30 |
| A-n63-k10 | 1314 | 0.08 | 4.72 | 3.65 | 4.17 | 3705.10 | 2.36 | 1.52 | 2.55 | 3724.10 | 1.75 | 1.14 | 2.50 | 3800.10 |
| A-n63-k9 | 1616 | 0.50 | 4.21 | 2.85 | 6.42 | 3559.92 | 2.78 | 2.23 | 2.84 | 3578.18 | 2.35 | 2.04 | 1.42 | 3651.20 |
| A-n64-k9 | 1401 | 0.57 | 4.07 | 3.00 | 4.15 | 3736.01 | 2.50 | 1.36 | 4.70 | 3755.16 | 1.50 | 1.00 | 1.89 | 3831.80 |
| A-n65-k9 | 1174 | 0.34 | 4.77 | 3.24 | 5.06 | 3757.85 | 2.64 | 1.36 | 4.23 | 3777.12 | 1.45 | 0.94 | 1.83 | 3854.20 |
| A-n69-k9 | 1159 | 0.00 | 4.31 | 3.02 | 4.59 | 4349.38 | 3.19 | 1.81 | 4.39 | 4371.68 | 1.64 | 0.95 | 2.53 | 4460.90 |
| A-n80-k10 | 1763 | 0.74 | 5.67 | 4.71 | 5.29 | 6331.26 | 4.20 | 3.40 | 3.71 | 6363.73 | 3.35 | 2.95 | 2.23 | 6493.60 |
| Averages | | 0.16 | 3.00 | 2.05 | 2.98 | 2277.27 | 1.52 | 0.84 | 2.14 | 2288.95 | 0.91 | 0.60 | 1.05 | 2335.66 |

**Table 4**
Computational results for the problem set B.

| Problem | BKS | SC-ESA | LNSi | | | | LNSa | | | | LNS-ACO | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | % Dev | % Dev | | StdDev | CPU | % Dev | | StdDev | CPU | % Dev | | StdDev | CPU |
| | | | Worst | Best | | | Worst | Best | | | Worst | Best | | |
| B-n31-k5 | 672 | 0.00 | 0.00 | 0.00 | 0.00 | 815.78 | 0.00 | 0.00 | 0.00 | 819.92 | 0.00 | 0.00 | 0.00 | 828.20 |
| B-n34-k5 | 788 | 0.00 | 0.00 | 0.00 | 0.00 | 894.88 | 0.00 | 0.00 | 0.00 | 899.42 | 0.00 | 0.00 | 0.00 | 908.51 |
| B-n35-k5 | 955 | 0.84 | 0.00 | 0.00 | 0.00 | 983.95 | 0.00 | 0.00 | 0.00 | 988.95 | 0.00 | 0.00 | 0.00 | 998.94 |
| B-n38-k6 | 805 | 1.24 | 0.00 | 0.00 | 0.00 | 1201.60 | 0.50 | 0.00 | 1.32 | 1207.70 | 0.00 | 0.00 | 0.00 | 1219.90 |
| B-n39-k5 | 549 | 0.18 | 4.19 | 2.19 | 3.79 | 1222.39 | 1.28 | 0.00 | 2.02 | 1228.59 | 0.00 | 0.00 | 0.00 | 1241.00 |
| B-n41-k6 | 829 | 4.46 | 2.41 | 1.33 | 2.53 | 1371.22 | 1.33 | 0.48 | 1.86 | 1378.18 | 0.24 | 0.00 | 0.57 | 1392.10 |
| B-n43-k6 | 742 | 0.54 | 2.96 | 1.48 | 3.33 | 1479.47 | 0.81 | 0.00 | 1.64 | 1486.98 | 0.40 | 0.00 | 1.03 | 1502.00 |
| B-n44-k7 | 909 | 1.32 | 2.09 | 1.10 | 2.57 | 1598.95 | 0.88 | 0.00 | 2.45 | 1607.07 | 0.00 | 0.00 | 0.00 | 1623.30 |
| B-n45-k5 | 751 | 0.00 | 3.33 | 1.60 | 2.86 | 1597.37 | 1.07 | 0.00 | 2.39 | 1605.48 | 0.00 | 0.00 | 0.00 | 1621.70 |
| B-n45-k6 | 678 | 1.18 | 3.98 | 2.36 | 3.13 | 1632.83 | 1.33 | 0.59 | 1.52 | 1641.12 | 0.29 | 0.00 | 0.51 | 1657.70 |
| B-n50-k7 | 741 | 0.00 | 3.78 | 2.16 | 3.17 | 2142.18 | 0.00 | 0.00 | 0.00 | 2153.05 | 0.00 | 0.00 | 0.00 | 2174.80 |
| B-n50-k8 | 1312 | 1.30 | 2.67 | 1.75 | 3.31 | 2136.66 | 1.14 | 0.61 | 2.09 | 2147.51 | 0.53 | 0.53 | 0.00 | 2169.20 |
| B-n51-k8 | 1016 | 0.00 | 2.36 | 1.28 | 2.80 | 2195.07 | 0.59 | 0.00 | 1.43 | 2206.22 | 0.20 | 0.00 | 0.64 | 2228.50 |
| B-n52-k7 | 747 | 0.67 | 3.35 | 1.47 | 4.16 | 2254.96 | 0.94 | 0.40 | 0.99 | 2266.41 | 0.13 | 0.00 | 0.31 | 2289.30 |
| B-n56-k7 | 707 | 0.00 | 3.54 | 1.98 | 3.52 | 2669.15 | 1.70 | 0.71 | 1.79 | 2682.70 | 0.00 | 0.00 | 0.00 | 2709.80 |
| B-n57-k8 | 1140 | 0.00 | 2.54 | 1.49 | 3.46 | 2801.73 | 0.88 | 0.53 | 1.18 | 2815.96 | 0.00 | 0.00 | 0.00 | 2844.40 |
| B-n57-k9 | 1598 | 0.13 | 2.25 | 1.13 | 5.55 | 3009.96 | 0.94 | 0.56 | 1.44 | 3025.24 | 0.31 | 0.00 | 1.53 | 3055.80 |
| B-n63-k10 | 1496 | 2.81 | 3.68 | 2.41 | 5.64 | 3694.64 | 2.14 | 1.60 | 1.96 | 3713.39 | 1.20 | 1.20 | 0.00 | 3750.90 |
| B-n64-k9 | 861 | 0.00 | 5.92 | 3.60 | 6.02 | 3777.08 | 2.90 | 2.21 | 1.83 | 3796.25 | 2.09 | 1.51 | 1.41 | 3834.60 |
| B-n66-k9 | 1316 | 1.90 | 3.65 | 2.43 | 3.68 | 4178.96 | 1.90 | 1.52 | 1.35 | 4200.17 | 1.52 | 1.06 | 1.55 | 4242.60 |
| B-n67-k10 | 1032 | 1.74 | 5.23 | 3.59 | 4.39 | 4455.16 | 2.91 | 2.42 | 1.62 | 4477.77 | 2.52 | 1.74 | 2.48 | 4523.00 |
| B-n68-k9 | 1272 | 1.57 | 4.09 | 2.83 | 4.52 | 4329.27 | 2.44 | 1.89 | 2.16 | 4351.25 | 1.97 | 1.42 | 2.10 | 4395.20 |
| B-n78-k10 | 1221 | 2.05 | 3.11 | 1.64 | 5.11 | 5958.46 | 1.88 | 1.06 | 3.04 | 5988.71 | 1.23 | 0.57 | 1.99 | 6049.20 |
| Averages | | 0.95 | 2.83 | 1.64 | 3.20 | 2452.25 | 1.20 | 0.63 | 1.48 | 2464.70 | 0.55 | 0.35 | 0.61 | 2489.59 |

has lover averages of %Dev values for the best achievements for the problem sets B, P and CMT. For the problem set E, LNS–ACO has almost the same average of %Dev values with SC–ESA algorithm for the best achievements. LNS–ACO has the worst performance on the problem set A in comparison to SC–ESA algorithm in terms of % Dev values for the best achievements.

For providing a better understanding, we summarized the results given in Tables 3–7 in Table 8 by considering the optimum achievements for each problem set. To our point of view, this summary emphasizes the effectiveness of the proposed hybrid LNS–ACO algorithm in comparison to other algorithms more clearly.

**Table 5**
Computational results for the problem set P.

| Problem | BKS | SC-ESA | LNSi | | | | LNSa | | | | LNS-ACO | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | % Dev | % Dev | | StdDev | CPU | % Dev | | StdDev | CPU | % Dev | | StdDev | CPU |
| | | | Worst | Best | | | Worst | Best | | | Worst | Best | | |
| P-n16-k8 | 450 | 0.00 | 0.00 | 0.00 | 0.00 | 711.49 | 0.00 | 0.00 | 0.00 | 715.18 | 0.00 | 0.00 | 0.00 | 737.30 |
| P-n19-k2 | 212 | 3.30 | 0.00 | 0.00 | 0.00 | 351.16 | 0.00 | 0.00 | 0.00 | 352.98 | 0.00 | 0.00 | 0.00 | 363.90 |
| P-n20-k2 | 216 | 0.93 | 0.00 | 0.00 | 0.00 | 340.68 | 0.00 | 0.00 | 0.00 | 342.45 | 0.00 | 0.00 | 0.00 | 353.04 |
| P-n21-k2 | 211 | 0.47 | 0.00 | 0.00 | 0.00 | 385.89 | 0.00 | 0.00 | 0.00 | 387.89 | 0.00 | 0.00 | 0.00 | 399.88 |
| P-n22-k2 | 216 | 0.00 | 0.00 | 0.00 | 0.00 | 398.66 | 0.00 | 0.00 | 0.00 | 400.72 | 0.00 | 0.00 | 0.00 | 413.12 |
| P-n22-k9 | 590 | 0.00 | 1.19 | 0.00 | 2.24 | 543.14 | 0.00 | 0.00 | 0.00 | 545.96 | 0.00 | 0.00 | 0.00 | 562.84 |
| P-n23-k8 | 529 | 0.00 | 1.70 | 0.00 | 2.75 | 593.98 | 0.76 | 0.00 | 1.09 | 597.06 | 0.00 | 0.00 | 0.00 | 615.53 |
| P-n40-k5 | 458 | 0.22 | 5.02 | 3.28 | 1.98 | 1184.54 | 1.31 | 0.00 | 1.74 | 1190.68 | 0.00 | 0.00 | 0.00 | 1227.50 |
| P-n45-k5 | 510 | 0.20 | 4.51 | 2.75 | 2.53 | 1514.47 | 0.98 | 0.00 | 1.46 | 1522.32 | 0.00 | 0.00 | 0.00 | 1569.40 |
| P-n50-k7 | 554 | 0.00 | 4.87 | 2.71 | 3.69 | 1954.70 | 1.26 | 0.54 | 1.14 | 1964.83 | 0.90 | 0.00 | 1.55 | 2025.60 |
| P-n50-k8 | 631 | 0.95 | 5.86 | 4.28 | 2.52 | 2067.32 | 2.85 | 2.54 | 0.64 | 2078.03 | 2.85 | 1.90 | 1.81 | 2142.30 |
| P-n50-k10 | 696 | 0.14 | 3.45 | 1.87 | 2.95 | 2265.63 | 0.86 | 0.43 | 0.75 | 2277.37 | 0.57 | 0.00 | 1.24 | 2347.80 |
| P-n51-k10 | 741 | 0.00 | 4.32 | 2.70 | 3.86 | 2363.38 | 2.02 | 1.48 | 1.10 | 2375.63 | 1.48 | 0.81 | 1.51 | 2449.10 |
| P-n55-k7 | 568 | 1.06 | 4.93 | 2.11 | 4.43 | 2444.25 | 1.76 | 0.53 | 2.01 | 2456.91 | 0.00 | 0.00 | 0.00 | 2532.90 |
| P-n55-k8 | 588 | 0.00 | 3.91 | 2.04 | 3.14 | 2556.14 | 0.85 | 0.00 | 1.49 | 2569.38 | 0.17 | 0.00 | 0.26 | 2648.85 |
| P-n55-k10 | 694 | 0.14 | 2.88 | 2.02 | 1.68 | 2775.92 | 0.43 | 0.00 | 0.79 | 2790.30 | 0.72 | 0.00 | 1.36 | 2876.60 |
| P-n55-k15 | 989 | -0.95 | 2.22 | 1.31 | 2.73 | 2946.63 | 0.71 | 0.00 | 1.73 | 2961.90 | 0.00 | 0.00 | 0.00 | 3053.50 |
| P-n60-k10 | 744 | 0.13 | 5.91 | 3.76 | 5.05 | 3273.96 | 3.23 | 2.55 | 1.49 | 3290.92 | 2.15 | 1.48 | 1.55 | 3392.70 |
| P-n60-k15 | 968 | 0.00 | 3.82 | 2.69 | 3.38 | 3857.30 | 2.27 | 1.65 | 1.64 | 3877.28 | 1.55 | 0.93 | 1.73 | 3997.20 |
| P-n65-k10 | 792 | 0.51 | 3.79 | 2.65 | 2.45 | 3747.10 | 2.27 | 1.77 | 1.08 | 3766.51 | 1.77 | 1.01 | 1.73 | 3883.00 |
| P-n70-k10 | 827 | 0.00 | 5.20 | 2.90 | 5.48 | 4478.18 | 2.18 | 1.81 | 0.82 | 4501.38 | 2.06 | 1.21 | 2.20 | 4640.60 |
| P-n76-k4 | 593 | 2.36 | 6.75 | 3.54 | 5.33 | 4877.79 | 2.36 | 1.69 | 1.17 | 4903.06 | 1.69 | 0.84 | 1.39 | 5054.70 |
| P-n76-k5 | 627 | 3.03 | 7.50 | 4.78 | 5.89 | 4767.10 | 4.31 | 3.67 | 1.24 | 4791.80 | 3.67 | 2.87 | 1.25 | 4940.00 |
| Averages | | 0.54 | 3.38 | 1.97 | 2.70 | 2191.28 | 1.32 | 0.81 | 0.93 | 2202.63 | 0.85 | 0.48 | 0.77 | 2270.75 |

**Table 6**
Computational results for the problem set E.

| Problem | BKS | SC-ESA | LNSi | | | | LNSa | | | | LNS-ACO | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | % Dev | % Dev | | StdDev | CPU | % Dev | | StdDev | CPU | % Dev | | StdDev | CPU |
| | | | Worst | Best | | | Worst | Best | | | Worst | Best | | |
| E-n22-k4 | 375 | 0.00 | 0.80 | 0.00 | 0.86 | 425.02 | 0.00 | 0.00 | 0.00 | 438.44 | 0.00 | 0.00 | 0.00 | 447.39 |
| E-n23-k3 | 569 | 0.00 | 0.88 | 0.00 | 1.51 | 373.42 | 0.00 | 0.00 | 0.00 | 385.22 | 0.00 | 0.00 | 0.00 | 393.08 |
| E-n30-k4 | 503 | 0.00 | 1.79 | 0.00 | 2.60 | 664.03 | 0.00 | 0.00 | 0.00 | 685.00 | 0.00 | 0.00 | 0.00 | 698.98 |
| E-n33-k4 | 835 | 0.48 | 1.68 | 0.36 | 3.20 | 777.13 | 0.84 | 0.00 | 2.33 | 801.67 | 0.00 | 0.00 | 0.00 | 818.03 |
| E-n76-k14 | 1021 | 0.00 | 5.29 | 4.21 | 3.59 | 5881.17 | 3.33 | 2.45 | 2.55 | 6066.89 | 4.21 | 0.88 | 9.06 | 6190.70 |
| E-n76-k7 | 682 | 2.05 | 6.74 | 4.99 | 3.50 | 4926.99 | 3.96 | 3.08 | 1.55 | 5082.57 | 1.91 | 1.91 | 0.00 | 5186.30 |
| E-n76-k8 | 735 | 1.09 | 6.26 | 4.22 | 4.26 | 5025.22 | 3.13 | 2.04 | 2.25 | 5183.91 | 3.13 | 1.22 | 3.77 | 5289.70 |
| E-n101-k14 | 1067 | 1.41 | 5.25 | 3.47 | 5.59 | 11,437.05 | 2.72 | 1.87 | 2.51 | 11,798.22 | 1.41 | 1.41 | 0.00 | 12,039.00 |
| Averages | | 0.63 | 3.59 | 2.16 | 3.14 | 3688.75 | 1.75 | 1.18 | 1.40 | 3805.24 | 1.33 | 0.68 | 1.60 | 3882.90 |

**Table 7**
Computational results for the problem set CMT.

| Problem | BKS | SC-ESA | LNSi | | | | LNSa | | | | LNS-ACO | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | % Dev | % Dev | | StdDev | CPU | % Dev | | StdDev | CPU | % Dev | | StdDev | CPU |
| | | | Worst | Best | | | Worst | Best | | | Worst | Best | | |
| CMT-n51-k5 | 521 | 0.00 | 4.22 | 3.45 | 1.29 | 2006.81 | 1.15 | 0.58 | 0.89 | 2017.21 | 0.00 | 0.00 | 0.00 | 2079.60 |
| CMT-n76-k10 | 830 | 0.96 | 3.25 | 2.41 | 1.99 | 5138.84 | 1.33 | 0.84 | 1.08 | 5165.46 | 0.24 | 0.24 | 0.00 | 5325.22 |
| CMT-n101-k8 | 817 | 1.84 | 3.92 | 2.82 | 2.79 | 9974.24 | 1.47 | 0.86, | 1.58 | 10,025.92 | 0.00 | 0.00 | 0.00 | 10,336.00 |
| CMT-n101-k10 | 820 | 0.00 | 3.05 | 2.32 | 1.98 | 10,559.03 | 0.00 | 0.00, | 0.00 | 10,613.74 | 0.00 | 0.00 | 0.00 | 10,942.00 |
| CMT-n121-k7 | 1034 | 0.19 | 3.09 | 1.84 | 3.26 | 15,355.08 | 1.35 | 0.87 | 1.39 | 15,434.64 | 0.00 | 0.00 | 0.00 | 15,912.00 |
| CMT-n151-k12 | 1053 | 0.76 | 6.08 | 4.75 | 4.35 | 28,722.26 | 3.13 | 2.47 | 2.01 | 28,871.08 | 2.47 | 1.80 | 1.94 | 29,764.00 |
| CMT-n200-k17 | 1291 | 3.33 | 9.45 | 8.91 | 1.66 | 61,884.49 | 6.04 | 4.88 | 4.54 | 62,205.13 | 5.89 | 3.87 | 6.70 | 64,129.00 |
| Averages | | 1.01 | 4.72 | 3.78 | 2.47 | 19,091.54 | 2.07 | 1.50 | 1.64 | 19,190.46 | 1.23 | 0.85 | 1.23 | 19,783.97 |

**Table 8**
Summary of the computational study.

| Algorithm | Problem set | | | | | | | | | | | | | | | Overall | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | | | B | | | P | | | E | | | CMT | | | | | |
| | OA | NP | SR | OA | NP | SR | OA | NP | SR | OA | NP | SR | OA | NP | SR | OA | NP | SR |
| SC-EAS | 16 | 27 | 0.59 | 8 | 23 | 0.35 | 9 | 23 | 0.39 | 4 | 8 | 0.50 | 2 | 7 | 0.29 | 39 | 88 | 0.44 |
| LNSi | 3 | | 0.11 | 4 | | 0.17 | 7 | | 0.30 | 3 | | 0.38 | 0 | | 0.00 | 17 | | 0.19 |
| LNSa | 10 | | 0.37 | 9 | | 0.39 | 12 | | 0.52 | 3 | | 0.38 | 1 | | 0.14 | 35 | | 0.40 |
| LNS-ACO | 17 | | 0.63 | 16 | | 0.70 | 15 | | 0.65 | 4 | | 0.50 | 4 | | 0.57 | 56 | | 0.64 |

***OA***: Number of optimum achievements; ***NP***: Number of problems; ***SR***: Success rate (OA/NP)

**Table 9**
Summary of the computational efforts of LNS variants.

| Algorithm | Averages | | | |
|---|---|---|---|---|
| | %Dev | | StdDev | CPU |
| | Worst | Best | | |
| LNS-ACO | 0.97 | 0.59 | 1.05 | 6152.57 |
| LNSi | 3.50 | 2.32 | 2.90 | 5940.22 |
| LNSa | 1.57 | 0.99 | 1.52 | 5990.40 |

From the observation of the summary given by Table 8, we can conclude that the proposed hybrid LNS–ACO has a satisfactory performance in solving CVRP problems especially in terms of solution quality; however, it is a little bit slower as stated above. The algorithm solved the problems sets A, B, P, E and CMT with success rates of 0.63, 0.70, 0.65, 0.50 and 0.57 respectively. LNS–ACO algorithm is capable to solve 56 problem instances out of 88 problems and it refers to an average success rate of 0.64. Additionally, the results given in Tables 3–7 are summarized via Table 9 in terms of worst and best values of %Dev, StdDev and CPU times by considering overall average values of these performance metrics.

The averages (worst and best) of %Dev values indicate the capability of LNS-ACO algorithm in producing optimal or near-optimal solutions for all the 88 instances used in this part of the computational study. Besides, the average of the StdDev values indicates the consistency of the proposed LNS-ACO algorithm in comparison to other two LNS variants; however the average of CPU times indicates the slower speed of the LNS-ACO algorithm. As a result, this summary emphasizes the satisfactory performance of LNS-ACO algorithm in solving CVRP instances.

### 5.2.1. Statistical analyses

In this subsection, a paired *t*-test, is executed on Excel 2007 in order to understand whether the differences in obtained results of the algorithms (LNS-ACO, LNSi and LNSa) are due to random chance or not. This paired *t*-test is executed on the obtained results by running the LNS variants 30 times and Table 10 represents the results of this test.

From the observation of the *p*-values presented in Table 10, we can conclude that LNS-ACO outperforms both LNSi and LNSa and LNSa outperforms LNSi for all the problem sets. Because of the second part of the computational study, we can strongly conclude that the proposed hybrid LNS-ACO algorithm is capable to tackle the CVRP instances with a high-level performance in comparison to other variants of LNS algorithm.

## 5.3. Comparison of hybrid LNS–ACO algorithm against some other CVRP methods

In this third part of the computational study, it is aimed to indicate the effectiveness of the hybrid LNS-ACO algorithm over 7 formerly developed CVRP methods, OS (Osman, 1993), GHL (Gendreau, Hertz, & Laporte, 1994), CGL (Cordeau, Gendreau, & Laporte, 1997), TV (Toth & Vigo, 1998), WH (Wark & Holt, 1994), RR (Rego & Roucairol, 1996) and BB (Berger & Barkaoui, 2003). Within this purpose, we used the benchmark set (14 CMT instances) used by Berger and Barkaoui (2003), since they also provided a comprehensive performance evaluation test over 6 formerly developed CVRP methods. The results for all the algorithms and the best-known solutions (BKS) for all the test problems were taken from Berger and Barkaoui (2003). Moreover, the hybrid LNS-ACO algorithm is compared against these algorithms in Table 11. The results given in Table 11 for the LNS-ACO algorithm are the results of a single run, since Berger and Barkaoui (2003) also provided results of a single run for the other methods. The best-known results achieved by the algorithms are marked in boldfaces.

The performance evaluation test given in Table 11 indicates the effectiveness of the hybrid LNS-ACO algorithm, since it was able to produce best-known results for nine problems out of 14. On the other hand, we did not compared the computational times of the algorithms, because of the differences of the hardware and software used to run the algorithms. Because of the complete computational study, we can strongly claim that the proposed hybrid LNS-ACO algorithm has an effective performance in solving the CVRP instances in comparison to other LNS variants and solution procedures.

**Table 10**
*p*-values of the paired *t*-test ($\alpha = 0.05$).

| | *p* values | | | | |
|---|---|---|---|---|---|
| | Problem Set A | Problem Set B | Problem Set P | Problem Set E | Problem Set CMT |
| LNS–ACO versus LNSi | 1.7E–247 | 1.9E–216 | 1.7E–172 | 2.8E–048 | 9.1E–087 |
| LNS–ACO versus LNSa | 9.1E–112 | 1.2E–137 | 2.1E–089 | 2.2E–015 | 1.2E–047 |
| LNSa versus LNSi | 2.7E–223 | 1.4E–183 | 3.7E–160 | 9.9E–052 | 9.9E–072 |

**Table 11**
Numerical results for LNS-ACO compared to other CVRP methods.

| Problem instance | BKS | OS | GHL | CGL | TV | WH | RR | BB | LNS-ACO |
|---|---|---|---|---|---|---|---|---|---|
| CMT1 | 524.61 | **524.61** | **524.61** | **524.61** | **524.61** | **524.61** | **524.61** | **524.61** | **524.61** |
| CMT2 | 835.26 | 844 | 835.77 | 835.45 | 838.60 | 835.8 | 835.32 | **835.26** | **835.26** |
| CMT3 | 826.14 | 838 | 829.45 | 829.44 | 828.56 | 830.7 | 827.53 | 827.39 | **826.14** |
| CMT4 | 1028.42 | 1044.35 | 1036.16 | 1038.44 | 1033.21 | 1038.5 | 1044.35 | 1036.16 | 1046.9 |
| CMT5 | 1291.45 | 1334.55 | 1322.65 | 1305.87 | 1318.25 | 1321.3 | 1334.55 | 1324.06 | 1341.4 |
| CMT6 | 555.43 | **555.43** | **555.43** | **555.43** | **555.43** | **555.4** | **555.43** | **555.43** | **555.43** |
| CMT7 | 909.68 | 911.00 | 913.23 | **909.68** | 920.72 | 911.8 | **909.68** | **909.68** | **909.68** |
| CMT8 | 865.94 | 878.00 | **865.94** | 866.38 | 869.48 | 878.0 | 866.75 | 868.32 | **865.94** |
| CMT9 | 1162.55 | 1184.00 | 1177.76 | 1171.81 | 1173.12 | 1176.5 | 1164.12 | 1169.15 | 1164.93 |
| CMT10 | 1395.85 | 1441.00 | 1418.51 | 1415.40 | 1435.74 | 1418.3 | 1420.84 | 1418.79 | 1419.70 |
| CMT11 | 1042.11 | 1043.33 | 1073.47 | 1074.13 | 1042.87 | 1043.4 | **1042.11** | 1043.11 | **1042.11** |
| CMT12 | 819.56 | 819.59 | **819.56** | **819.56** | **819.56** | **819.56** | **819.56** | **819.56** | **819.56** |
| CMT13 | 1541.14 | 1547.00 | 1573.81 | 1568.91 | 1545.51 | 1548.3 | 1550.17 | 1553.12 | 1547.10 |
| CMT14 | 866.37 | **866.37** | **866.37** | 866.53 | **866.37** | 866.4 | **866.37** | **866.37** | **866.37** |
| Average deviation from BKS | | 1.03 | 0.86 | 0.69 | 0.64 | 0.63 | 0.55 | 0.48 | 0.55 |

## 6. Conclusions

In this paper, we aimed at developing a novel hybrid meta-heuristic algorithm for the capacitated vehicle routing problem. Main idea was to improve the search capability of the large neighbourhood search algorithm, which has a satisfactory performance in solving the vehicle routing problems. In the proposed hybrid algorithm, we embed the solution construction mechanism of ACO into the LNS. The basic LNS and its variants usually realize a solution acceptance criterion/function, which we projected this criterion/function out of the LNS algorithm and incorporate the solution construction mechanism of ACO instead. Thus, we aimed to bring a satisfactory level of diversification in the basic LNS within a hybridization manner. Additionally, the proposed hybrid LNS–ACO algorithm uses both improvement and construction type of heuristics during its search and the algorithm got a more powerful search capacity. The performance of the proposed hybrid LNS–ACO algorithm was tested on a set of CVRP instances against two LNS variants and seven formerly developed CVRP methods. Computational results indicate that the proposed hybrid LNS–ACO algorithm is able to solve CVRP instances with a satisfactory performance.

Future researches will focus on implementing the proposed hybrid LNS–ACO algorithm to different types of vehicle routing problems as well as to different combinatorial optimization problems.

## References

Adulyasak, Y., Cordeau, J. F., & Jans, R. (2012). Optimization-based adaptive large neighborhood search for the production routing problem. *Transportation Science, 48*(1), 20–45.

Akpinar, S., Bayhan, G. M., & Baykasoglu, A. (2013). Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. *Applied Soft Computing, 13*(1), 574–589.

Augerat, P., Belenguer, J.M., Benavent, E., Corbern, A., Naddef, D., & Rinaldi, G. (1995). Computational results with a branch and cut code for the capacitated vehicle routing problem, Research Report 949-M, Universite Joseph Fourier, Grenoble, France.

Belo-Filho, M. A. F., Amorim, P., & Almada-Lobo, B. (2015). An adaptive large neighbourhood search for the operational integrated production and distribution problem of perishable products. *International Journal of Production Research, 53*(20), 6040–6058.

Berger, J., & Barkaoui, M. (2003). A new hybrid genetic algorithm for the capacitated vehicle routing problem. *Journal of the Operational Research Society, 54*(12), 1254–1262.

Blum, C., Puchinger, J., Raidl, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing, 11*(6), 4135–4151.

Braekers, K., Ramaekers, K., & Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*. doi:10.1016/j.cie.2015.12.007.

Christofides, N., & Eilon, S. (1969). An algorithm for the vehicle dispatching problem. *Operational Research Society, 20*, 309–318.

Christofides, N., Mingozzi, A., & Toth, P. (1979). The vehicle routing problem. In N. Christofides, A. Mingozzi, P. Toth, & L. Sandi (Eds.), *Combinatorial optimization* (pp. 315–338). Chicheste: Wiley.

Clarke, G., & Wright, J. V. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research, 12*, 568–581.

Cordeau, J. F., Gendreau, M., & Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks, 30*(2), 105–119.

Crainic, T. G., & Toulouse, M. (2003). Parallel strategies for meta-heuristics. In F. Glover, & G. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 475–513). Dordrecht: Kluwer Academic Publishers.

Demir, E., Bektaş, T., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research, 223*(2), 346–359.

Dorigo, M., Maniezzo, V., Colorni, A., & Maniezzo, V. (1991). Positive feedback as a search strategy. Technical Report No: 91-016, Politecnico di Milano, Italy.

Eksioglu, B., Vural, A. V., & Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering, 57*(4), 1472–1483.

Elhassania, M. J., Jaouad, B., & Ahmed, E. A. (2013). A new hybrid algorithm to solve the vehicle routing problem in the dynamic environment. *International Journal of Soft Computing, 8*(5), 327–334.

Gendreau, M., Hertz, A., & Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science, 40*(10), 1276–1290.

Jin, J., Crainic, T. G., & Løkketangen, A. (2014). A cooperative parallel metaheuristic for the capacitated vehicle routing problem. *Computers & Operations Research, 44*, 33–41.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science, 43*(4), 408–416.

Laporte, G., Musmanno, R., & Vocaturo, F. (2010). An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science, 44*(1), 125–135.

Lin, S. W., Lee, Z. J., Ying, K. C., & Lee, C. Y. (2009). Applying hybrid meta-heuristics for capacitated vehicle routing problem. *Expert Systems with Applications, 36*(2), 1505–1512.

Luo, Z., Qin, H., Zhang, D., & Lim, A. (2016). Adaptive large neighborhood search heuristics for the vehicle routing problem with stochastic demands and weight-related cost. *Transportation Research Part E: Logistics and Transportation Review, 85*, 69–89.

Masson, R., Lehuédé, F., & Péton, O. (2013). An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science, 47*(3), 344–355.

Muller, L. F., Spoorendonk, S., & Pisinger, D. (2012). A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research, 218*(3), 614–623.

Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research, 41*(4), 421–451.

Pang, K. W. (2011). An adaptive parallel route construction heuristic for the vehicle routing problem with time windows constraints. *Expert Systems with Applications, 38*(9), 11939–11946.

Pisinger, D., & Ropke, S. (2010). Large neighborhood search. In M. Gendreau, & J.-Y. Potvin (Eds.), *Handbook of metaheuristics* (pp. 399–419). New York: Springer.

Preux, P., & Talbi, E. G. (1999). Towards hybrid evolutionary algorithms. *International Transactions in Operational Research, 6*(6), 557–570.

Raidl, G. R. (2006). A unified view on hybrid metaheuristics. In *Hybrid metaheuristics* (pp. 1–12). Berlin Heidelberg: Springer.

Rego, C., & Roucairol, C. (1996). A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In I. H. Osman, & J. P. Kelly (Eds.), *Meta-heuristics: Theory and applications* (pp. 661–675). U.S.: Springer.

Rifai, A. P., Nguyen, H. T., & Dawal, S. Z. M. (2016). Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling. *Applied Soft Computing, 40*, 42–57.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science, 40*(4), 455–472.

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

Semet, F., Toth, P., & Vigo, D. (2014). Classical exact algorithms for the capacitated vehicle routing problem. *Vehicle routing: problems, methods, and applications*, 37–57.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings CP-98 (fourth international conference principles practice constraint programming)*.

Stanojević, M., Stanojević, B., & Vujošević, M. (2013). Enhanced savings calculation and its applications for solving capacitated vehicle routing problem. *Applied Mathematics and Computation, 219*(20), 10302–10312.

Talbi, E. G. (2002). A taxonomy of hybrid metaheuristics. *Journal of Heuristics, 8*(5), 541–564.

Ting, T. O., Yang, X. S., Cheng, S., & Huang, K. (2015). Hybrid metaheuristic algorithms: Past, present, and future. In *Recent advances in swarm intelligence and evolutionary computation* (pp. 71–83). Springer International Publishing.

Toth, P., & Vigo, D. (1998). The granular tabu search and its application to the vehicle routing problem. Technical report OR/ 98/9 DEIS, University of Bologna, Bologna, Italy.

Toth, P., & Vigo, D. (Eds.). (2002). *The vehicle routing problem*. Siam.

Toth, P., & Vigo, D. (Eds.) (2014). *Vehicle routing: problems, methods, and applications* (18). Siam.

Vilarinho, P. M., & Simaria, A. S. (2006). ANTBAL: An ant colony optimization algorithm for balancing mixed-model assembly lines with parallel workstations. *International Journal of Production Research, 44*(2), 291–303.

Wark, P., & Holt, J. (1994). A repeated matching heuristic for the vehicle routeing problem. *Journal of the Operational Research Society, 45*(10), 1156–1167.