Journal of the Operational Research Society (2017) 68, 1422–1441

www.palgrave.com/journals

A bi-objective mathematical model for two-dimensional loading time-dependent vehicle routing problem

Mahdi Alinaghian[1]*, Komail Zamanlou[1] and Mohammad S. Sabbagh[1]

[1]*Department of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran*

This paper introduces two-dimensional loading time-dependent vehicle routing problem and proposes a bi-objective mathematical model. This problem assesses the process of distributing the rectangular-shaped demanded items over an urban environment; it does not, however, allow items to be loaded on top of each other. In addition to the above assumptions, the presented model also satisfies the first-in-first-out property in the time-dependent vehicle routing problem. Given the NP-hard nature of the problem, a method called elitist non-dominated sorting local search is developed to obtain its solutions. To evaluate the performance of the proposed algorithm, the solutions of this algorithm for small-scale problem instances are compared with the results of an exact method. For the medium-scale problem instances, results of NSGA-II and SPEA2 are used as the basis of comparison. The computational results demonstrate the good performance of the proposed method.

Journal of the Operational Research Society (2017) **68(11),** 1422–1441. doi:10.1057/s41274-016-0151-x; published online 8 February 2017

Keywords: two-dimensional loading time-dependent vehicle routing problem; elitist non-dominated sorting local search; NSGA-II; FIFO property

1. Introduction

Vehicle routing problem (VRP) addresses one of the most important issues of distribution operations, i.e., how to use a limited number of vehicles to supply the demand of customers from a depot. Over the years, researchers have developed multiple versions of this problem to deal with different distribution specifications. The most common constraints of VRP include maximum vehicles' capacity, maximum travel distance and serving time windows defined for customers. In some VRP applications, the size of items to be distributed decides whether it is possible to load them into loading space. A typical example of this issue is the delivery of industrial machinery. In this type of problems, expressing the demand by weight is not enough and size of the items must also be taken into consideration.

Such problems can be solved through solving a variety of two- or three-dimensional bin packing problems (2BPP-3BPP) and by using separate processes to solve VRP and loading problem. This approach, however, severely reduces the likelihood of obtaining optimal, desirable or even feasible solutions. Therefore, incorporation of the features and constraints of loading problem into VRP has led to the development of new problems that simultaneously assess both issues. Two-dimensional loading capacitated vehicle routing problem (2L-CVRP) and three-dimensional loading capacitated vehicle routing problem (3L-CVRP) are among the most applicable approaches in this regard. The difference between these two is in the nature of demanded items; the basic assumption of 2L-CVRP is that items cannot be loaded on top of each other, while 3L-CVRP has no such assumption. It should be noted that this paper is based on 2L-CVRP assumptions regarding loading constraints.

The typical 2L-CVRP assumes that each arc has a constant travel time throughout the planning horizon. But in most large cities, traffic congestion at different times of a day (especially in rush hours) can significantly alter the time required to pass a given arc. Traffic congestion also affects the labor and fleet cost structure (Figliozzi, 2010). Incorporating these concepts in the model of a distribution service operating in an urban environment can prevent the model from obtaining sub-optimal and inefficient solutions.

In this paper, travel time is modeled as a function of the time at which vehicle set out from the origin node. This study uses a piecewise linear function to inject the concept of time dependency into the travel time. The proposed model also satisfies first-in-first-out (FIFO) property, which is an important feature of time-dependent vehicle routing problem (TDVRP).

In addition, balancing the loads to be distributed by the vehicles not only prevents employees' dissatisfaction, but also reduces the abnormal depreciation of vehicles. So load balance can increase the model efficiency by providing the mentioned

*Correspondence: Mahdi Alinaghian, Department of Industrial and Systems Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran.
E-mail: alinaghian@cc.iut.ac.ir

benefits. In this paper, load balance requirement is satisfied regarding the weight of items assigned to vehicles.

This paper presents an extended version of 2L-CVRP model called two-dimensional loading time-dependent vehicle routing problem (2L-TDVRP), which integrates the concepts of two-dimensional loading constraints and time dependency with the load balancing requirements. Given the NP-hard nature of this problem, a meta-heuristic algorithm called elitist non-dominated sorting local search (ENSLS) is proposed to solve the problem. The results obtained from solving small-scale problem instances by the proposed algorithm are compared with the results of an exact solution method, and the results obtained from solving medium-scale problem instances are compared with the results of NSGA-II and SPEA2 meta-heuristic methods.

The rest of this paper is structured as follows: Section 2 reviews the literature on this topic, Section 3 describes the problem and presents its mathematical model, Section 4 describes the proposed solution approach, Section 5 presents the computational results, and Section 6 presents the conclusions and recommendations for future research.

2. Review of the literature

The first paper of VRP literature is the work of Dantzig *et al* (1954), where authors studied a large-scale traveling salesman problem (TSP) and proposed a solution method. Clarke and Wright (1964) were the first researchers who assessed this problem for more than one vehicle. However, Golden *et al* (1977) used the term "vehicle routing" in the title of their paper for the first time. More information on this subject can be found in Toth and Vigo (2002) and Kumar and Panneerselvam (2012).

Time-dependent vehicle routing problem is one of the most widely known versions of the VRP. Malandraki (1989) was the first researcher who introduced the mathematical model of TDVRP. Also Malandraki and Daskin (1992) later proposed a mixed integer mathematical model for TDVRP, which used a step function to calculate the travel time between two customers. To solve this model, they used a number of simple heuristics.

In real world, speed variations are not in the form of instant jumps, so dislike mentioned studies, Hill and Benton (1992) used a speed function to avoid these jumps and provided a compact mathematical model, a number of methods to estimate its parameters, and a solution method. Donati *et al* (2008) considered two hierarchical objective functions, which optimized the number of tours and the total travel time on selected routes. They used a meta-heuristic method based on ant colony optimization (ACO) algorithm to solve this TDVRP. Soler *et al* (2009) first used a multistage approach to convert the TDVRP to an asymmetric VRP and then solved that problem using methods available in the literature of VRP. Figliozzi (2012) presented a route construction and improvement method to solve the mentioned problem. Zhang *et al* (2014) assessed the time-dependent vehicle routing problem with simultaneous pickup and delivery. They proposed an integer programming model for that problem and solved this problem using a hybrid method composed of ACO and tabu search algorithms.

FIFO property is among the most realistic features of TDVRP and is defined as follows: when a vehicle departs the node i(origin node) at the time t_i, its arrival time at the destination node is always less than the case where it departs the node i at the time $t_i' > t_i$. Most of the previous studies that have introduced a comprehensive mathematical model for TDVRP or its real-world applications have neglected the FIFO property. Here, a piecewise linear function with linear slope restricted to values greater than -1 is used to model the travel time.

The piecewise linear function used in this paper follows an approach rather different than other studies in the literature. In previous studies, to uphold the FIFO property, travel time needs to be obtained from a speed function, and this dependence imposes extra computation; but in this paper travel time function is independent from speed function. In the paper of Ichoua *et al* (2003), it was stated that if the slope of travel time function is greater than -1, the FIFO property is respected. However, they did not develop any mathematical model based on this property, as their focus was on the use of heuristic methods to solve the problem. The mathematical model developed by Jabali *et al* (2012) for time-dependent vehicle routing problem relied on FIFO property. But the disadvantage of that model is the presence of instant jumps in the speeds of different time periods, which is not realistic. Also, the aforementioned model assumes only one daily change in speed function, and injecting a greater number of daily speed changes in their model will need complex modifications. In contrast, the present paper uses a piecewise linear travel time function, which provides the assumption of continuous daily changes in travel time. The use of this function also allows us to easily incorporate a large number of daily travel time variations into the model.

Review of the literature on simultaneous routing and loading problems shows that most of the studies in this area are focused on incorporating the loading constraints in capacitated vehicle routing problem (CVRP). This problem was first introduced by Iori *et al* (2007). They used an exact solution approach to solve the problem in small scale. Gendreau *et al* (2008) were the first researches who proposed a meta-heuristic algorithm (a method based on tabu search algorithm) to solve the large-scale instances of the mentioned problem. In their algorithm, lower bounds, heuristic and exact methods were used to check the loading constraints. Zachariadis *et al* (2009) implemented guided tabu search to solve the problem and also used a set of heuristic methods to check the loading constraints. Fuellerer *et al* (2009) proposed an ACO-based meta-heuristic method to solve problem. In this paper, heuristic methods were used to check loading constraints,

which the most important feature of these heuristic methods is their ability to consider variable orientation. Strodl *et al* (2010) used a variable neighborhood search (VNS) method to solve the problem and used exact and heuristic methods to check the loading constraints. Leung *et al* (2011) presented a developed version of guided tabu search and to check loading constraints, used some heuristic methods.

Duhamel *et al* (2011) presented a method to solve the 2L-CVRP which first converts the loading constraints to the constraints of a resource-constrained project scheduling problem, then uses GRASP × ELS algorithm to solve the converted problem and finally converts the solutions back to the 2L-CVRP form. Zachariadis *et al* (2013) presented a meta-heuristic method, whose most important feature is its compact structure. Their solution approach uses a local search to solve the problem and uses diversification strategies to avoid getting trapped in local optimums. It also uses heuristic methods to check the loading constraints. In a study by Hamdi-Dhaoui *et al* (2014), authors assessed the problem where there is a conflict between customers' items and provided a bi-objective model for this problem. Their first objective function optimizes the total service cost, while the second objective function balances the items assigned to vehicles regarding their area. To solve the problem and check loading constraints, they used a meta-heuristic algorithm and heuristic methods, respectively. Dominguez *et al* (2014) examined the problem for the instances where item rotation is allowed, and then used a routing algorithm to solve the problem. Their method uses improved heuristics to check the loading constraints.

The literature on this topic contains similar problems that are extended versions of classical routing and loading problems. Malapert *et al* (2008) proposed an extended version of 2L-CVRP with simultaneous pickup and delivery constraints. They converted loading constraints to scheduling constraints and used a constraint programming model to solve the problem. Leung *et al* (2013) assessed the VRP with heterogeneous fleet, combined the simulated annealing algorithm with local search heuristic to solve the routing problem and used a variety of heuristic methods to address the loading constraints. Khebbache-Hadji *et al* (2013) proposed the 2L-CVRP with time windows and then presented a meta-heuristic solution approach for this problem which is a combination of memetic algorithm and some loading heuristics. Dominguez *et al* (2016) also assessed the VRP with heterogeneous fleet and proposed a heuristic approach to solve it in the scenarios where item rotation is/is not allowed. Also, they used two heuristic methods to check the loading constraints. More information on VRP with two-dimensional loading constraints can be found in Iori and Martello (2010) and Iori and Martello (2013).

3. Problem statement

Assume the directed graph $G = (V, A)$ where V is the set of nodes $0, 1, \ldots, n$. In the set V, node 0 represents the depot and nodes $1, \ldots, n$ represent the customers. A denotes the set of arcs $(i, j), i, j \in V$. The first objective of the problem is to determine the routes which can minimize the serving time. The secondary objective of the problem is to balance the distribution of items assigned to vehicles regarding to the weight of items. The customers are located in an urban area, so the travel time between each two nodes depends on the departure time from the origin node. As said before, in the present paper, a piecewise linear function with linear slope restricted to values greater than -1 is used to model the travel time. The use of this function also forces the changes in travel time to be smooth rather than stepwise behavior and thereby improves the reality of the model. Figure 1 can help to clarify that how this function can guarantee the FIFO property. Suppose that at time t_A vehicle (A) starts to traverse an arc, and at time t_B another vehicle (B) starts to traverse the same arc. According to time function, duration time of travel for vehicles A and B will be D_A and D_B. Vehicle A will reach the destination at the time $t_A + D_A$. Since slope is greater than -1, $t_B - t_A$ is greater than $D_A - D_B$ and therefore $t_A + D_A$ (arrival time of vehicle A) will be less than $t_B + D_B$ (arrival time of vehicle B).

3.1. Assumptions

This modeling process is also based on several other assumptions:

- The travel time between two nodes also depends on the direction of travel.
- The fleet is homogenous (in terms of operation cost, capacity and speed).
- All available vehicles must be used.
- Customers' demanded items are rectangular shaped.
- Customers should be allocated to the vehicles with respect to weight and loading constraints.

It should be noted that loading constraints considered for the presented model are based on the assumptions of "two-dimensional unrestricted oriented loading." Further information in this regard can be found in Fuellerer *et al* (2009).

3.2. Mathematical model

This section describes the proposed mathematical model. Before presenting the model, its parameters and variables need to be introduced. The model parameters are as follows:

The set of nodes $i = 0, \ldots, n$ where 0 denotes the depot; i, j, p are the indices associated with this set.

The set of vehicles $k = 1, \ldots, K$; k is the index associated with this set.

The set of time intervals $m = 1, \ldots, M$; m is the index associated with this set.

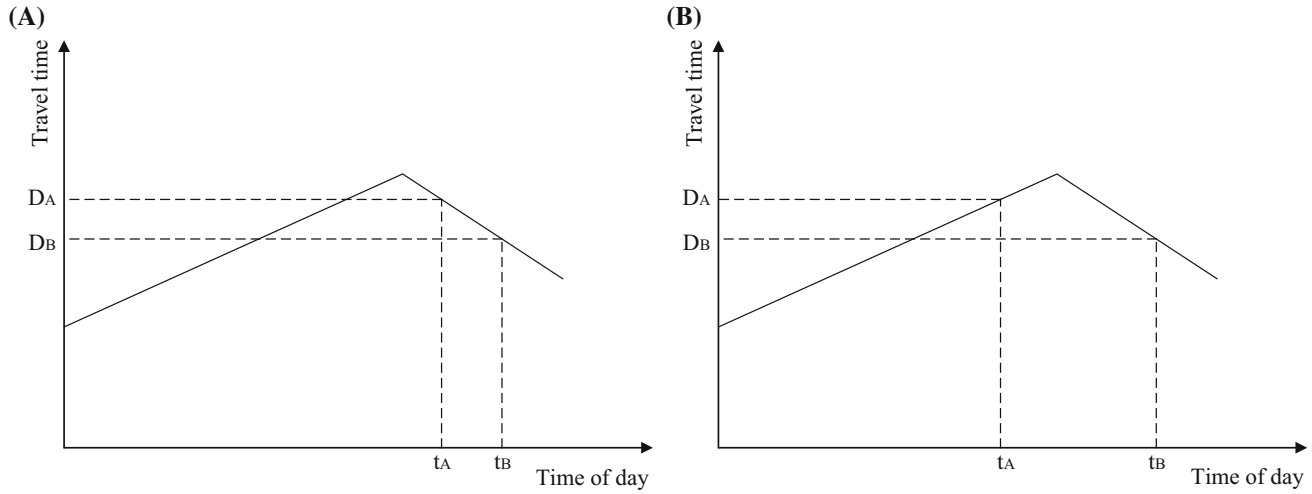

Figure 1 Guarantee of FIFO property by piecewise linear function with linear slope restricted to values greater than −1.

The set of items $it = 1, \ldots, A$; it and jt are the indices associated with this set.

T_{ij}^0 The starting time horizon
T_{ij}^m Upper bound of m-th time interval
C_{ij}^m The travel time from node i to node j when travel happens at time T_{ij}^m
st_i The service time of customer i
t_0 Departure time from the depot
D_i The demand of customer i
w_{it} The width of it-th item
h_{it} The height of it-th item
W The width of vehicles' loading space
H The height of vehicles' loading space
S Area of vehicles' loading space ($S = W \times H$)
Q Capacity of vehicles
Bel_{it}^i is one if item it belongs to customer i, and is zero otherwise
B A big number

The model variables are as follows:

x_{ij}^{km} A binary variable which equals one when at time period m vehicle k moves from node i to node j, and is zero otherwise
θ_{ij} The travel time from node i to node j
t_i The departure time from customer i
z_i^k A binary variable which equals one when vehicle k transports the items of customer i, and is zero otherwise
φ_{it}^k A binary variable which equals one when vehicle k transports item it, and is zero otherwise
xw_{it} The X-coordinate (width) of bottom-left corner of item it
yh_{it} The Y-coordinate (length) of bottom-left corner of item it
$l_{it,jt}$ A binary variable which equals one when item it is placed on the left side of the item jt, and is zero otherwise
$b_{it,jt}$ A binary variable which equals one when item it is placed on the down side of item jt, and is zero otherwise
u_i^k Auxiliary variable (sub-tour elimination)
Bl The maximum weight load on vehicles

The mathematical model is explained in the following:

$$\min O_1 = \sum_{i=0}^{n}\sum_{j=0}^{n}\theta_{ij} \qquad \min O_2 = Bl \tag{1}$$

$$\theta_{ij} = \sum_{k=1}^{K}\sum_{m=1}^{M}\left(C_{ij}^{m-1} + \left(\frac{C_{ij}^m - C_{ij}^{m-1}}{T_{ij}^m - T_{ij}^{m-1}}\right)\left(t_i - T_{ij}^{m-1}\right)\right)x_{ij}^{km}, \quad i = 0, \ldots, n;\ j = 0, \ldots, n; \tag{2}$$

$$t_0 = 0 \tag{3}$$

$$t_j = \sum_{i=0}^{n} t_i \sum_{k=1}^{K}\sum_{m=1}^{M} x_{ij}^{km} + \sum_{i=0}^{n}\theta_{ij} + st_j,\ j = 1, \ldots, n; \tag{4}$$

$$t_i + B \cdot \sum_{k=1}^{K} x_{ij}^{km} \le T_{ij}^m + B, \quad i = 0, \ldots, n;\ j = 0, \ldots, n;\ m = 1, \ldots, M; \tag{5}$$

$$t_i - T_{ij}^{m-1} \cdot \sum_{k=1}^{K} x_{ij}^{km} \ge 0, \quad i = 0, \ldots, n;\ j = 0, \ldots, n;\ m = 1, \ldots, M; \tag{6}$$

$$\sum_{i=0}^{n}\sum_{k=1}^{K}\sum_{m=1}^{M} x_{ij}^{km} = 1, \quad j = 1, \ldots, n; \tag{7}$$

$$\sum_{j=1}^{n}\sum_{m=1}^{M} x_{0j}^{km} = 1, \quad k = 1,\ldots,K; \tag{8}$$

$$\sum_{i=0}^{n}\sum_{m=1}^{M} x_{ip}^{km} - \sum_{j=0}^{n}\sum_{m=1}^{M} x_{pj}^{km} = 0, \quad p = 0,\ldots,n;\ k = 1,\ldots,K; \tag{9}$$

$$u_i^k - u_j^k + n\sum_{m=1}^{M} x_{ij}^{km} \le n-1, \quad i = 0,\ldots n;\ j = 1,\ldots n;\ k = 1,\ldots,K; \tag{10}$$

$$\sum_{j=0}^{n}\sum_{m=1}^{M} x_{ij}^{km} = z_i^k, \quad i = 0,\ldots,n;\ k = 1,\ldots,K; \tag{11}$$

$$\varphi_{it}^k = Bel_{it}^i \cdot z_i^k, \quad i = 1,\ldots,n;\ k = 1,\ldots,K;\ it = 1,\ldots,A; \tag{12}$$

$$\sum_{i=1}^{n} z_i^k \cdot D_i \le Q, \quad k = 1,\ldots,K; \tag{13}$$

$$xw_{it} + w_{it} \le W, \quad it = 1,\ldots,A; \tag{14}$$

$$yh_{it} + h_{it} \le H, \quad it = 1,\ldots,A; \tag{15}$$

$$l_{it,jt} + l_{jt,it} + b_{it,jt} + b_{jt,it} + \left(1-\varphi_{it}^k\right) + \left(1-\varphi_{jt}^k\right) \ge 1, \quad it, jt = 1,\ldots,A;\ it \ne jt;\ k = 1,\ldots,k; \tag{16}$$

$$xw_{it} + w_{it} + W.l_{it,jt} \le xw_{jt} + W, it, jt = 1,\ldots,A;\ it \ne jt; \tag{17}$$

$$yh_{it} + h_{it} + H.b_{it,jt} \le yh_{jt} + H, \quad it, jt = 1,\ldots,A;\ it \ne jt; \tag{18}$$

$$\sum_{i=0}^{n} z_i^k D_i \le Bl \quad \forall k = 1,\ldots,K; \tag{19}$$

$$x_{ij}^{km}, z_i^k, \varphi_{it}^k, l_{it,jt}, b_{it,jt} \in \{0,1\}; \quad \theta_{ij}, t_i, u_i^k, xw_{it}, yh_{it} \in R^+. \tag{20}$$

Equation (1) expresses the objective functions, which minimizes the service time and minimizes the maximum weight load on vehicles. Constraint (2) calculates the travel time for the arc $i-j$. Constraints (3) and (4) determine the vehicles' departure time from the depot and the node j, respectively. Constraints (5) and (6) ensure that at the time period m the vehicle k can pass through the arc $i-j$ only when its departure time from node i is at the same time interval. Constraint (7) ensures that each customer gets served exactly once. Constraint (8) ensures that vehicles start their tours from the depot. Constraint (9) states that each vehicle that enters a node must eventually leave it. Constraint (10) eliminates the sub-tours. Constraints (11) and (12) are related to the allocation of customers and items to vehicles, respectively. Constraint (13) is related to capacity constraint of vehicles. Constraints (14) and (15) ensure that the item demanded by customer i is placed in the loading space. Constraints (16) to (18) prevent the items from overlapping in the loading space. Constraint (19) is related to the maximum weight load on vehicles, and finally constraint (20) determines the type and domain of variables.

3.3. Linearization of the second and fourth constraints

After replacing $b_{ij}^{km} = t_i \cdot x_{ij}^{km}$, the second and fourth constraints change to the following form:

$$\theta_{ij} = \sum_{k=1}^{K}\sum_{m=1}^{M}\left(C_{ij}^{m-1}\cdot x_{ij}^{km} + \left(\frac{C_{ij}^m - C_{ij}^{m-1}}{T_{ij}^m - T_{ij}^{m-1}}\right)\left(b_{ij}^{km} - T_{ij}^{m-1}.x_{ij}^{km}\right)\right), \quad i,j = 0,\ldots,n; \tag{21}$$

$$t_j = \sum_{\substack{i=0\\ i\ne j}}^{n}\sum_{k=1}^{K}\sum_{m=1}^{M} b_{ij}^{km} + \sum_{\substack{i=0\\ i\ne j}}^{n} \theta_{ij} + st_j, \quad j = 1,\ldots,n. \tag{22}$$

This linearization also requires adding the following four constraints to the model:

$$b_{ij}^{km} \le t_i, \quad i,j = 0,\ldots,n;\ k = 1,\ldots,K;\ m = 1,\ldots,M; \tag{23}$$

$$b_{ij}^{km} \ge t_i - B\left(1 - x_{ij}^{km}\right), \quad i,j = 0,\ldots,n;\ k = 1,\ldots,K;\ m = 1,\ldots,M; \tag{24}$$

$$b_{ij}^{km} \le Bx_{ij}^{km}, \quad i,j = 0,\ldots,n; k = 1,\ldots,K; m = 1,\ldots,M; \tag{25}$$

$$b_{ij}^{km} \ge 0, \quad i,j = 0,\ldots,n; k = 1,\ldots,K; m = 1,\ldots,M. \tag{26}$$

By adding the above constraints to the model and using the mentioned linearized objective function and constraints, computational time for solving the problem is reduced which is the advantage of linear models over nonlinear ones.

4. Solution approach

This section discusses the solution method used for the described problem. It first presents a brief introduction to multi-objective optimization and then explains some notions about the heuristic algorithms used for checking the feasibility of solutions in terms of loading constraints. In the end, it

presents the proposed algorithm and discusses the general concepts of NSGA-II and SPEA2.

4.1. Multi-objective optimization

Many real-world problems can be described via several conflicting objectives. This perspective has led to the introduction and application of multi-objective optimization models and methods. Multi-objective problems express the superiority via the concept of dominance. The concept of dominance is defined as follows: suppose without loss of generality a minimization problem and suppose that x and y are the solution vectors with the j-th objective function values $f_j(x)$ and $f_j(y)$, respectively, then dominance is defined according to Equation (27):

$$\begin{aligned} x \text{ dominates } y \Leftrightarrow \forall \quad & j = 1, \ldots, m : f_j(x) \leq f_j(y) \\ \& \exists j_0 = 1, \ldots, m : & f_{j_0}(x) < f_{j_0}(y); \end{aligned} \tag{27}$$

where m is the number of objective functions. Unlike the mono-objective problems where the goal is to find an optimal solution, in multi-objective problems the goal is to find a set of solutions that any other solutions in the solution space cannot dominate them. Each of these solutions is called a Pareto optimal solution, and their set is called the Pareto optimal set. Also, the values of objective functions of Pareto optimal set is called Pareto front.

4.2. Loading heuristics

This paper uses several heuristic methods to check the feasibility of loading items into the loading space. The used heuristic methods include: bottom-left fill (Y-axis) (Chazelle, 1983), bottom-left fill (X-axis) (Chazelle, 1983), maximum touching perimeter (Lodi *et al*, 1999), maximum touching perimeter no walls (Lodi *et al*, 1999) and minimum area (Zachariadis *et al*, 2009). When the routing constraints and the weight requirements of a solution are satisfied, these methods will be used (in the mentioned order) to check the loading feasibility of that solution. These methods load items based on a pre-determined sequence. This paper uses two such sequences; when an algorithm fails to obtain a feasible solution via the first sequence, it proceeds to the second sequence. It should be noted that the first sequence is obtained by sorting all items assigned to a vehicle in a non-ascending order based on their area. The second sequence is based on the order of visits; in this sequence, items whose customers are going to be visited first will be loaded last, and all items of the same customer will be sorted in a non-ascending order based on the area. All these methods will place a given item in a location that will be on the list of accessible coordinates. After registering an item to a location in the loading space, coordinates of that location will be removed from the list and a maximum of four new coordinates will be added to that list. Each heuristic method uses its own criterion to select the coordinates from the list of accessible coordinates; these criteria are shown in Table 1. More detailed information can be found in Zachariadis *et al* (2009).

It should be noted that if none of the heuristic methods can obtain a feasible loading for an assessed route, the route will be reported as infeasible and its objective function will be penalized.

4.3. ENSLS

The proposed algorithm called ENSLS is an extended version of NSGA-II (Deb *et al*, 2002). Like NSGA-II, this algorithm ranks the solutions in the order of number of times they have been dominated by other members of population. This method (like NSGA-II) uses a non-dominated sorting algorithm for ranking the solutions and forming a number of fronts. Non-dominated sorting algorithm uses the following procedure to classify the solutions into several fronts: Algorithm first assigns all non-dominated solutions to the first front and then discards them from the solution set. In the second step, algorithm finds the non-dominated solutions in the new solution set, assigns them to the second front and again discards them from the solution set; algorithm repeats this loop unit all solutions are assigned to a class. Full and detailed information regarding this algorithm can be found in Deb *et al* (2002). Diversity preservation is an important issue in multi-objective optimizations, so (like NSGA-II) this method uses a parameter called the crowding distance to estimate the density of the neighborhood of a solution. This parameter which is defined within a front shows the distance of a member with the previous and next members. The first step for calculating this parameter is to determine the distance of each individual member i from the j-th objective function by Equation (28); this equation uses the sorted values of objective functions.

$$DI_i^j = \frac{\left| f_j(x_{i+1}) - f_j(x_{i-1}) \right|}{f_j^{\max} - f_j^{\min}}, \quad j = 1, \ldots, m; \tag{28}$$

in which $f_j(x_{i+1})$ and $f_j(x_{i-1})$ are values of the j-th objective function of next and previous members and $f_j^{\max}$, $f_j^{\min}$ are the best and worst values of the j-th objective in the front. Crowding distance of members on corners of the front is considered to be a large number. Ultimately, crowding distance of the i-th member of population can be obtained by using Equation (29).

$$Cd_i = \sum_{i=1}^{m} DI_i^j \tag{29}$$

According to Equation (29), the higher crowding distance of a member indicates that it has a higher fitness, since it is located in a less dense area.

ENSLS first generates an initial population, but unlike NSGA-II which uses the typical GA operators such as

Table 1 Coordinate selection criteria used by heuristic methods

Heuristic	*Criteria*
Bottom-left fill (*Y*-axis)	Minimum *Y*-axis coordinate, breaking ties by minimum *X*-axis coordinate
Bottom-left fill (*X*-axis)	Minimum *X*-axis coordinate, breaking ties by minimum *Y*-axis coordinate
Maximum touching perimeter	Maximum total touching perimeter (sum of the common arcs with the other items and the walls of loading space)
Maximum touching perimeter no walls	Maximum total touching perimeter (sum of the common arcs with the other items)
Minimum area	Minimum area of corresponding rectangular surface (Figure 2)

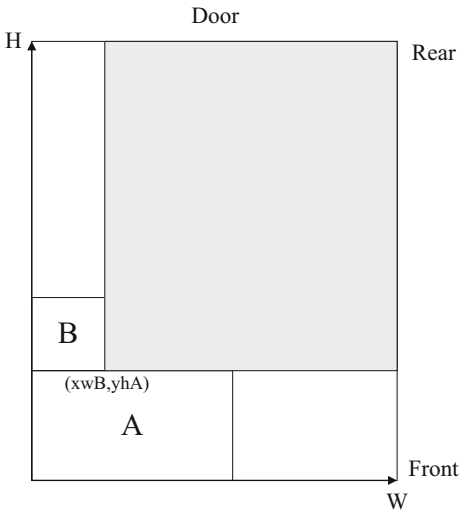

Figure 2 Corresponding rectangular surface (for coordinate (xw_B, yh_A)).

crossover and mutation to generate a new population, this method uses the concept of neighborhood for this purpose. Before running the algorithm, the probability of selection of each member for moving to its neighborhood is determined. Based on this probability, method determines the number of members to be selected in each iteration. Note that this method uses binary tournament for the selection of members. This method then merges the population of neighbors with the previous generation and uses the non-dominated sorting algorithm to classify them in several fronts. It then sorts the members of each front in the order of their crowding distance which the best solutions will be those with the highest crowding distance. This method then selects a number of solutions from the merged population (equal to the population size) and transfers them to the next generation (Figure 3). The process of generating new population, merging, sorting and transferring will be repeated until the termination condition is satisfied. After the last iteration, members of the first front will be reported as Pareto front approximates.

4.3.1. Initial population Half of the initial solutions are generated by a random method, and the other half are generated by a modified nearest neighbor random method. In the random method, customers are randomly assigned to vehicles. The modified nearest neighbor random method generates the solutions via following steps:

1. It first sorts the customers in the ascending order of their distance from the depot and then assigns the K top customers to K vehicles, where K is the number of vehicles.
2. It starts from the first vehicle, as long as the weight and loading constraints allow, randomly selects one of the two customers nearest to the last customer assigned to the vehicle and assigns it to that vehicle.
3. Once weight or loading constraints get violated, it proceeds to the next vehicle and repeats the process until all customers are allocated to vehicles.

4.3.2. Neighborhood Neighborhood is defined by three operators of swap (Waters, 1987), 2-opt (Croes, 1958; Lin, 1965) and or-opt (Waters, 1987). The swap operator switches the positions of two customers assigned to one or two different vehicles. Figure 4 shows its method of work.

This figure shows how this operator swaps the customers of a single route (left) or those from two different routes (right). In this figure, customers are marked with circles, and bolded circles represent the customers selected for the swap operation. The top and bottom sections of this figure show the status of route(s) before and after the swap. The second operator is a variant of 2-opt; Figure 5 shows how this operator processes two customers of same vehicle or those from two different vehicles.

As this figure shows, when both customers are from the same route (left), this operator reverses the sequence between the two, and when customers are from different routes (right), it replaces the sequences located after the two (including the selected customers).

The third operator is a variant of or-opt. This operator changes the location of a single customer. Figure 6 shows its method of function on a single route (left) or two different routes (right).

4.3.3. Flowchart of ENSLS Figure 7 shows different steps of the proposed method.

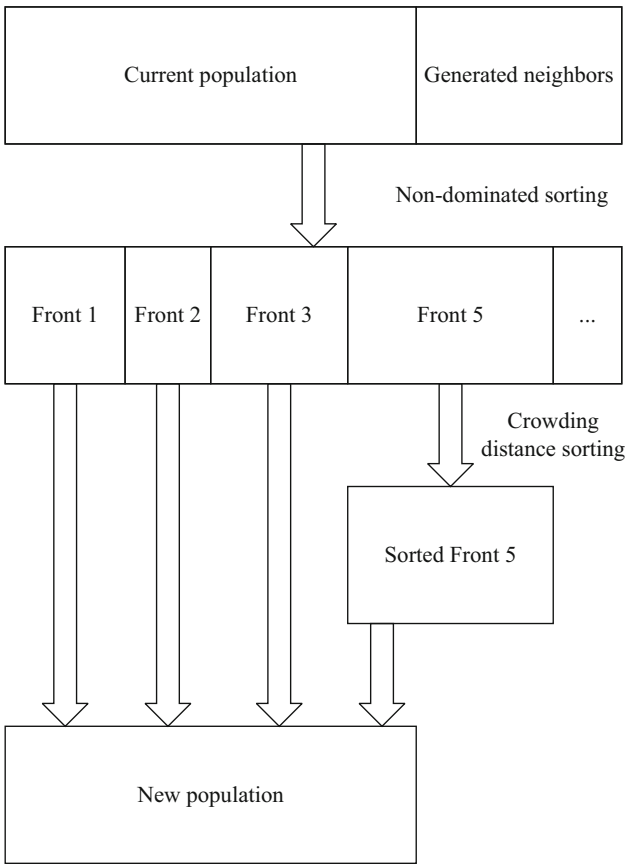

Figure 3 Method of updating the population in ENSLS.

4.4. An introduction to NSGA-II

NSGA-II is an extended version of genetic algorithm (Holland, 1975) focused on multi-objective optimization. The previous sections described those parts of NSGA-II that are utilized by the proposed algorithm; so this section only discusses the different part, i.e., the children generation.

4.4.1. Crossover and mutation operators in NSGA-II After selecting the parents which is done via a binary tournament in this paper, four operators including two-point, three-point, OX (Oliver *et al*, 1987) and AEX (Grefenstette *et al*, 1985) (all with equal probabilities of being used) are employed to generate the children. More information regarding OX and AEX operators can be found in (Puljić and Manger, 2013). Also the swap operator is used for mutation.

4.5. An introduction to SPEA2

SPEA2 first introduced by Zitzler *et al* (2001). Unlike NSGA-II and ENSLS that store the elite population within the main one, SPEA2 stores this population in another set called "archive." At the start of algorithm, archive is empty. Algorithm then generates an initial population and, at each iteration, merges the members of the main and archived population; it then assigns a fitness value to each member of merged population. This value is based on the number of times this member has dominated by other members, as well as the density of the region where it is located. This fitness value is

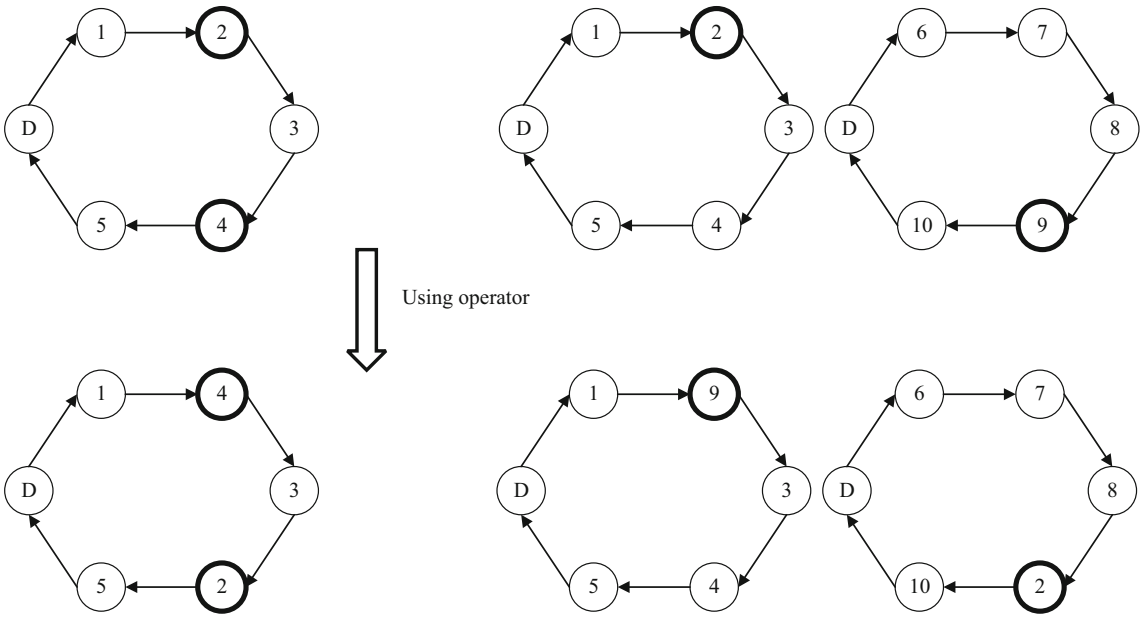

Figure 4 Swap operator.

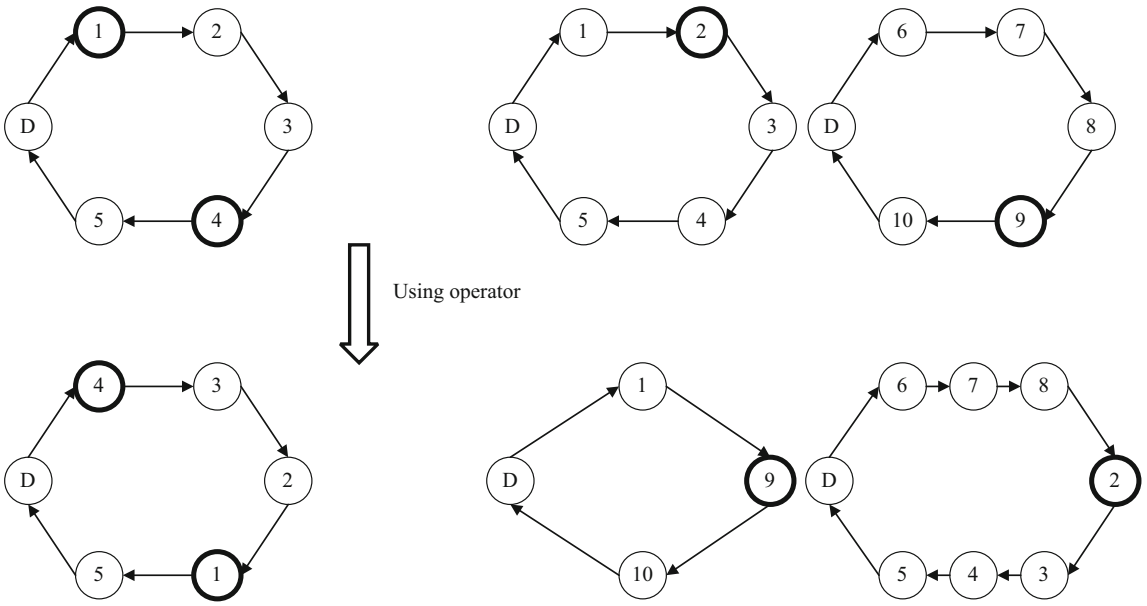

Figure 5 2-opt operator.

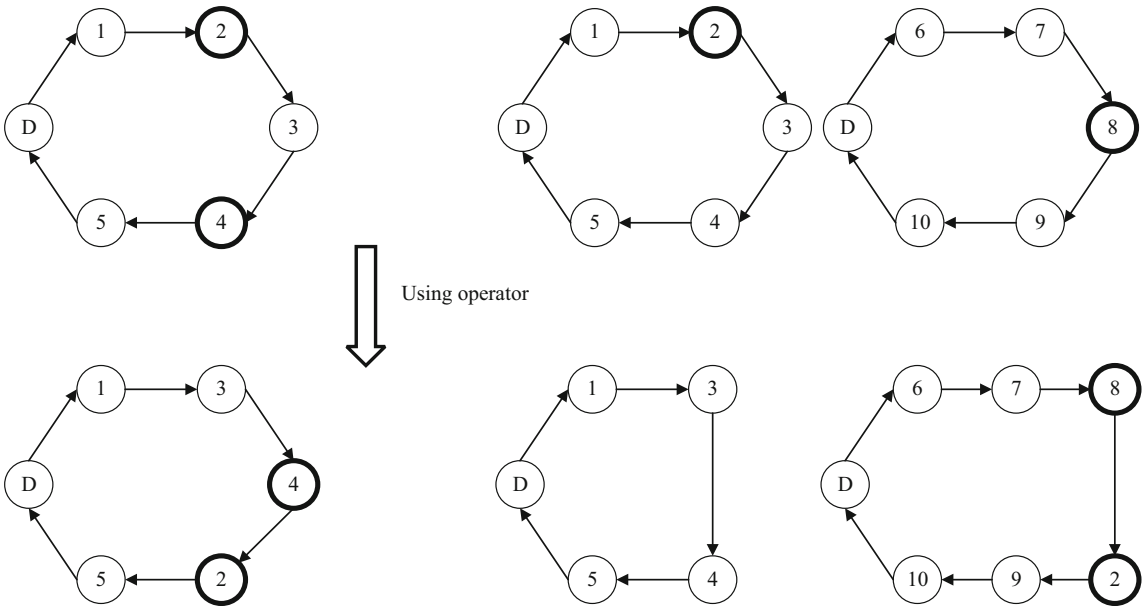

Figure 6 Or-opt operator.

obtained via following steps: Algorithm first calculates the number of members that are dominated by member i, and names it S_i. It then obtains the primary fitness of member i by summing the S values of those members that dominate i. Finally, it calculates the secondary fitness of each member (a value less than one) based on the density of the region where this member is located. In the end, it calculates the final fitness by summing the primary and secondary fitness values.

The algorithm must then generate a new archive; if the size of non-dominated members of merged population is less than or equal to the size of archive, algorithm sorts the members of merged population in the order of their fitness value and moves them to the archive; otherwise, it uses a truncation operator to select the best non-dominated members as much as the size of archive allows. Truncation operator calculates the distance of each member from the rest and then sorts these distances for each member. It then starts to remove the members with lowest distances; when two or more members have the same distance, operator moves to the next shortest distance and when necessary continues this process down to

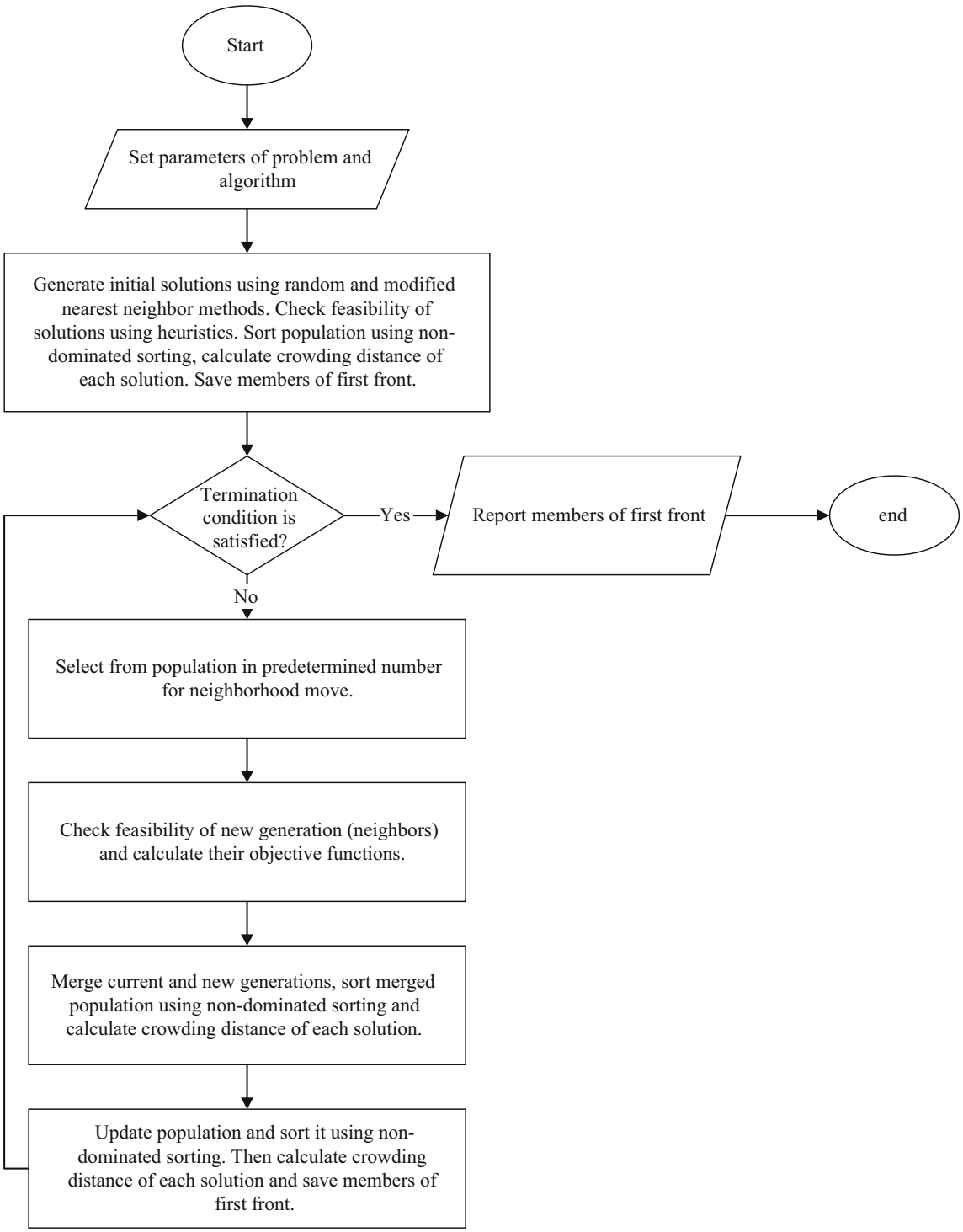

Figure 7 Flowchart of ENSLS.

$(k-1)th$ member (where k is the number of non-dominated members).

Finally, algorithm uses mutation and crossover operators to generate a new population. It should be noted that the new population will replace the current one. This process of merging, calculating the fitness value, creating the new archive and generating the new population will be repeated until termination condition is satisfied. After the last iteration, non-dominated members of the archive will be reported as the Pareto front approximates.

This algorithm generates the initial solutions by the same procedure explained in Section 4.3.1. It performs crossover and mutation through the same operators described in Section 4.4.1. Further information in this regard can be found in Zitzler *et al* (2001).

5. Computational results

This section presents and analyzes the computational results. To achieve this aim, first the method of generating problem instances will be described; then the method of tuning the parameters of the proposed algorithms will be explained; afterward the performance evaluation criteria will be defined; and finally the performance of the proposed algorithms by solving small- and medium-scale problem instances will be evaluated. It is worth noting that NSGA-II, SPEA2 and the proposed algorithm were coded in MATLAB and executed on a Core i5 2.5 GHz with 6 GB RAM PC under windows 8.1.

5.1. Problem instances

Since this is the first paper that studies this particular problem, we had no access to readily available problem instances, and the needed instances were generated. Two groups of instances were generated for this problem. The first group includes 12 instances with 5–10 customers and 2–3 vehicles. These instances are based on E016-03m, E022-04g and E033-03n instances available in 2L-CVRP literature (these instances can be found in http://www.or.deis.unibo.it/research.html). Coordinates and demands of customers in our instances are in accordance with these problem instances. Since our problems have 5–10 customers, coordinates and demands of this number of customers are used. Dimensions of items are changed from those in original instances, and only one item is attributed to each customer, because otherwise these problems could not be solved by the exact method. The vehicles capacity of the original instances is also changed to obtain more challenging problems. In the rest of this paper, these problems will be referred to as small-scale instance problems. The number of customers and the number of vehicles for these instances can be seen in Table 2:

The second group of problems, which are named medium-scale instances problems, includes 25 problems with 15–100 customers and 4–22 vehicles. In these instances, coordinates, demands and dimensions of items and vehicles are similar to selected 2L-CVRP problems. Five instances were selected from each class of mentioned problems, which included E016-05m, E036-11h, E051-05e, E072-04f and E101-14s. The number of customers and the number of vehicles for these instances can be seen in Table 3:

To add the concept of time dependency to the problems, a traffic pattern was designed and attached to each arcs. This traffic pattern consists of five time intervals and starts with the start of workday. The time required to travel the arc within the first time interval is considered as the distance of that arc, and then T_4 is defined in the form of Equation (30):

$$T_4 = \left(\frac{n}{K} + 1\right) \cdot \bar{T} \tag{30}$$

In the above equation, n is the number of customers, K is the number of vehicles, and $\bar{T}$ is the average distance between customers. The procedures shown in Table 4 are used to calculate the length of other time intervals (t_i denotes the end of i-th time interval).

In Table 4, $T_{\max}$ is the size of longest arc in the problem. It is obvious that the travel time on each arc depends on the time interval, the slope of the line in that interval and the length of the arc. Figure 8 shows an instance of the piecewise linear function with five time intervals described in Table 4.

Table 2 Number of customers and vehicles for small-scale instances

Sample	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
Number of customers	5	5	5	7	7	7	9	9	9	10	10	10
Number of vehicles	2	2	2	2	2	2	3	3	3	3	3	3

Table 3 Number of customers and vehicles for medium-scale instances

Sample	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*
Number of customers	15	35	50	71	100	15	35	50	71	100	15	35	50
Number of vehicles	5	11	5	4	14	5	11	11	14	19	5	11	11
Sample	14	15	16	17	18	19	20	21	22	23	24	25	
Number of customers	71	100	15	35	50	71	100	15	35	50	71	100	
Number of vehicles	15	22	5	11	12	16	22	5	11	12	16	22	

Table 4 Specifications of defined time interval

i	t_i	$slope_i$
1	$T_4/2$	0
2	$t_1 + T_4/8$	0.9
3	$t_2 + T_4/4$	0
4	$t_3 + T_4/8$	−0.9
5	$nT_{\max}$	0

5.2. Parameter setting

Performance of meta-heuristic methods largely depends on the values of their parameters, so the proper setting of parameters can have a significant impact on the performance of any approach that employs these methods. This paper uses the Taguchi method to set the parameters. Taguchi method for the design of experiments, introduced in 1960 by Taguchi, is among the most reliable methods used for adjusting the parameters; this method can provide the good conditions via the smallest possible number of experiment (Roy, 2010). Using this method instead of classic full factorial approach significantly reduces the time and cost of tests required to set the parameters. Based on the number of selected parameters and the factor levels, Taguchi method uses several orthogonal arrays as experiment matrices. Parameters of ENSIS include: the probability of selecting a solution for moving to its neighborhood (P_{nb}), the size of population in each iteration (*npopualtion*) and the number of iterations without any improvement (*non-improve*). Parameters of NSGA-II are: the probability of using crossover and mutation operators (P_c and P_m, respectively), the size of population in each iteration (*npopualtion*) and the number of iterations without any improvement (*non-improve*). SPEA2 also has the same parameters plus another parameter called the size of archive (*narchive*). It can be seen that ENSLS has a lower number of parameters, which can be considered as an advantage. Values of all the above parameters were determined by the use of Taguchi method. These values are presented in Table 5.

5.3. Performance metrics

In mono-objective optimization, the goal is to find a single optimal solution, so the solution methods proposed for these problems can be compared by comparing their reported objective function values. Runtime or computation time is another common metric for such comparison. In multi-objective optimization, however, solutions should not only have a good quality in terms of objective function values, but also have a good diversity and spread to cover more points of the Pareto front. In this paper, the following parameters are used to evaluate the performance of the proposed algorithms.

5.3.1. Quality metric (QM) To check the quality of the solutions, all solutions obtained from all methods are compared with each other. A new merged set of non-dominated solutions is obtained, and eventually the contribution of each algorithm to this set is determined. The higher contribution of an algorithm points out its better performance.

5.3.2. Spacing metric (SM) The spacing metric measures the uniformity of distribution of solutions. Literature has provided several measures for this metric, but this paper uses the measures introduced by (Deb *et al*, 2002). This metric can be calculated via Equation (31):

$$\mathrm{SM} = \frac{\sum_{i=1}^{n-1} \left| d_i - \bar{d} \right|}{(n-1)\bar{d}} \tag{31}$$

where n denotes the number of obtained solutions, d_i is the distance between two adjacent solutions in the objective space, and $\bar{d}$ is the average of all d_i s. Lower values of this metric point to better performance of the algorithm.

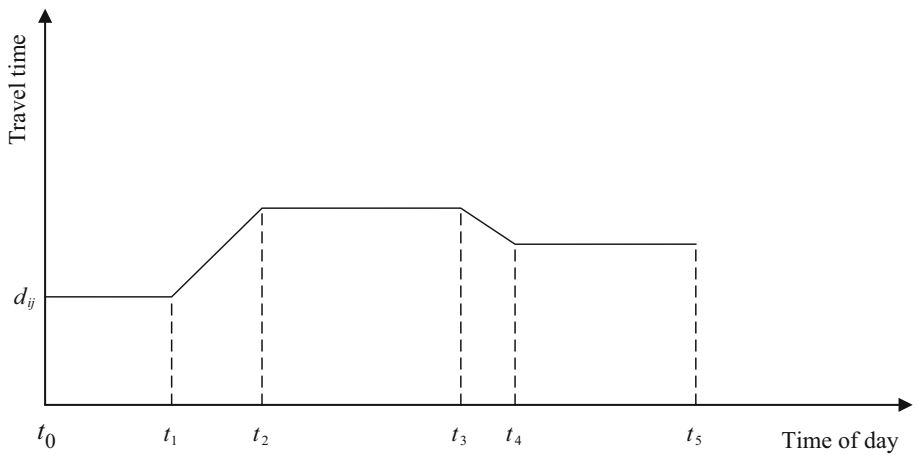

Figure 8 Piecewise linear function used in problem instances.

Table 5 Values set for parameters

Parameter/algorithm	*ENSLS*	*NSGA-II*	*SPEA2*
npopulation	15	15	15
non-improved	$10{\cdot}n$	$10{\cdot}n$	$10{\cdot}n$
P_{nb}	0.8	–	–
P_c	–	0.6	0.8
P_m	–	0.4	0.2
narchive	–	–	15

Table 6 Computational results obtained for small-scale problem instances (part 1)

Sample	*GAMS*		*ENSLS*			*NSGA-II*			*SPEA2*		
	Obj. 1	*Obj. 2*	*Obj. 1*	*%gap*	*Obj. 2*	*Obj. 1*	*%gap*	*Obj. 2*	*Obj. 1*	*%gap*	*Obj. 2*
1	248.29	3200	248.29	0.00	3200	248.29	0.00	3200	248.29	0.00	3200
2	138.69	44	138.69	0.00	44	138.69	0.00	44	138.69	0.00	44
3	1475.37	25705	1475.37	0.00	25705	1475.37	0.00	25705	1475.37	0.00	25705
4	256.94	3700	256.94	0.00	3700	256.94	0.00	3700	256.94	0.00	3700
5	201.36	59	201.36	0.00	59	201.36	0.00	59	201.36	0.00	59
6	1609.93	25705	1609.93	0.00	25705	1610.49	0.03	25705	1610.49	0.03	25705
7	293.70	2700	293.70	0.00	2700	295.51	0.62	2700	293.70	0.00	2700
8	253.33	51	253.33	0.00	51	253.33	0.00	51	253.33	0.00	51
9	1773.71	25705	1776.66	0.17	25705	1776.66	0.17	25705	1779.61	0.33	25705
10	328.07	2900	328.07	0.00	2900	328.07	0.00	2900	328.65	0.18	2900
11	273.51	53	273.51	0.00	53	273.51	0.00	53	274.36	0.31	53
12	1643.32	25705	1696.71	3.25	25705	1696.71	3.25	25705	1723.32	4.87	25705
Ave.	708.02	9627.25	712.71	0.28	9627.25	712.91	0.34	9627.25	715.34	0.48	9627.25

5.3.3. Diversity metric (DM) The metric, introduced by Zitzler (1999), demonstrates the extent of non-dominated solutions of an algorithm and can be calculated by Equation (32).

$$DM = \sqrt{\left(\frac{\max\limits_i f_{1i} - \min\limits_i f_{1i}}{f_{1\text{total}}^{\max} - f_{1\text{total}}^{\min}}\right)^2 + \left(\frac{\max\limits_i f_{2i} - \min\limits_i f_{2i}}{f_{2\text{total}}^{\max} - f_{2\text{total}}^{\min}}\right)^2} \quad (32)$$

In the above equation, $f_{i\text{total}}^{\max}$ and $f_{i\text{total}}^{\min}$ are the maximum and minimum values of i-th objective function in the solutions of all of the compared algorithms, respectively. The higher the value of DM, the better the performance of the algorithm.

5.3.4. Objective functions value In this paper, one of the criteria used for the comparison of algorithms is the best value for each objective function obtained by each algorithm.

5.3.5. Computational time Another criterion used for performance evaluation is the computation time. Time to find best solutions (OPT) and time to completion (TOT) are reported for all assessed algorithms.

5.4. Performance of the proposed algorithm in solving the small-scale problem instances

To evaluate the performance of the proposed method, its results have been compared with the results of two other algorithms and also with the results of an exact method. In this stage, metrics of comparison were the best value obtained for each objective function and the computational time. To find the optimal values of objective functions using exact method, the exact method was used to solve the model for one objective function regardless to the other one, and then the same process was followed for the other function. This gave the best possible value of each objective function. It should be mentioned that the time reported for the exact method is the sum of times it takes to obtain the optimal solutions for the two objectives functions. Then, ENSLS, NSGA-II and SPEA2 were run three times and the best values of objective functions obtained by each algorithm, time to find best solutions and time to completion were determined. In the end, the averages of above mentioned values obtained by each algorithm were calculated. The results are presented in Tables 6 and 7. Note that "gap%" in Table 6 shows the deviation of solutions from the solution of exact method. All three methods showed similar performances in terms of second objective function, so the deviations of this objective function were not calculated.

As the table shows, the proposed algorithm outperformed the other tested algorithms and generated solutions closest to the solutions of exact method. The average error of ENSLS, NSGA-II and SPEA2 for the first objective function is 0.28, 0.34 and 0.48%, respectively, and this demonstrates the good performance of ENSLS in this respect. In the small-scale problem instances, all algorithms have yielded almost identical

Table 7 Computational results obtained for small-scale problem instances (part 2)

Sample	*GAMS*	*ENSLS*		*NSGA-II*		*SPEA2*	
	Time	*OPT*	*TOT*	*OPT*	*TOT*	*OPT*	*TOT*
1	3.09	0.02	2.81	0.08	2.87	0.12	3.59
2	3.81	0.27	2.53	0.26	3.15	0.47	4.06
3	8.01	0.30	2.57	0.21	2.98	0.07	2.66
4	48.35	1.33	4.63	0.24	4.13	1.09	4.62
5	86.97	1.10	4.40	0.55	4.74	1.26	4.88
6	42.36	1.49	4.84	1.18	5.27	0.84	4.54
7	1612.67	2.26	6.55	7.06	12.70	6.27	11.46
8	628.90	1.83	6.17	2.53	7.99	2.52	7.58
9	1895.83	1.00	5.56	1.70	6.92	1.59	8.28
10	393.12	2.96	8.02	4.93	10.89	4.40	11.40
11	2604.95	1.24	6.18	2.37	8.65	1.80	7.30
12	7023.98	1.86	7.44	5.54	12.46	1.90	8.68
Ave.	1196.00	1.30	5.14	2.22	6.90	1.86	6.59

values for the second objective function, and this is because of small size of the problems. In the small-scale problems, there are only a handful of states regarding the amount of load to be allocated to each vehicle, so all algorithms can achieve optimal solutions. The computational time obtained for small-scale problem instances is shown in Table 7.

In terms of OPT Time, on average, ENSLS is about 41 and 30% faster than the NSGA-II and SPEA2 algorithms, respectively. Average values for OPT Time for ENSLS, NSGA-II and SPEA2 are 1.30, 2.22 and 1.86, respectively, that demonstrate the good performance of proposed algorithm. Also all algorithms showed approximately the similar performance in terms of the difference between OPT Time and TOT Time.

5.5. Performance of the proposed algorithm in solving the medium-scale problem instances

To gauge the performance of the proposed algorithms, like before, each of these three algorithms was run three times. The best run performed by each of these algorithms was used to compare them in terms of quality. However, the average values of diversity and spacing metrics, best values of objective functions as well as computational time through three runs were used as the basis of comparison in these terms. The results are presented in Tables 8 and 9.

The results presented in Table 8 indicate that in terms of quality metric (QM), the proposed algorithm is completely superior to the other two methods, as more than half of the best non-dominated solutions have been obtained by this algorithm. In terms of SM and DM metrics, the difference between the methods is negligible. The graph of Figure 9 provides a more clear understanding about the quality of solutions obtained by each algorithm for each instance. Note that higher values of the quality metric show the better performance of the algorithm.

Figure 9 shows that in most problem instances, the proposed algorithm provided higher quality solutions, and this quality difference increases with the increase in the number of customers. This difference in performance is reflected in the fact that in 64% of the problem instances, half or more than half of the final non-dominated solutions were the contributions of this algorithm; also, in 28% of the problem instances, all final non-dominated solutions were the contributions of this algorithm, which demonstrates the absolute superiority of ENSLS in terms of quality metric.

The SM values obtained by each algorithm for each problem instance are plotted in Figure 10. Note that lower values of the spacing metric represent the better performance of the algorithm.

Figure 10 shows the minor advantage of SPEA2. The results show that SPEA2, ENSLS and NSGA-II have had the best performance in 48, 32 and 20% of instances. In terms of average value of all SM values, the performances of all tested algorithms were somewhat similar.

The next stage of comparison is the diversity metric. Figure 11 shows the DM values for the tested algorithms for each problem instance. Note that higher values of the DM demonstrate the better performance of the algorithm.

Figure 11 shows the minor advantage of the proposed algorithm. The proposed method has had the best performance in 40% of the problems, while NSGA-II and SPEA2 have been the best algorithms in 32 and 28% of the instances. In terms of average value of all DM values, there is no significant difference between the performances of the algorithms. The objective functions values and computation time criteria obtained for medium-scale problem instances are presented in Table 9.

The results presented in Table 9 demonstrate the good performance of the proposed method in terms of best objective function value. The average errors of ENSLS, NSGA-II and SPEA2 for the first objective function are, 0.26, 4.22 and

Table 8 Computational results obtained for medium-scale problem instances (part 1)

Sample	*ENSLS*			*NSGA-II*			*SPEA2*		
	QM	*SM*	*DM*	*QM*	*SM*	*DM*	*QM*	*SM*	*DM*
1	0.40	0.24	0.89	0.20	0.51	1.37	0.40	0.07	1.08
2	0.33	0.27	1.10	0.67	0.24	1.01	0.00	0.31	1.23
3	1.00	0.30	0.66	0.00	0.30	0.52	0.00	0.31	1.19
4	1.00	0.94	1.26	0.00	0.74	1.21	0.00	0.67	0.85
5	1.00	0.54	1.04	0.00	0.28	0.74	0.00	0.40	1.13
6	0.25	0.00	1.30	0.25	0.31	1.30	0.50	0.22	1.11
7	0.00	0.00	0.71	0.50	0.09	0.88	0.50	0.31	1.10
8	0.71	0.74	0.98	0.29	0.90	0.99	0.00	0.68	0.83
9	1.00	1.02	1.08	0.00	1.06	1.15	0.00	0.35	0.34
10	0.83	0.70	1.05	0.00	0.77	0.97	0.17	0.85	1.01
11	0.25	0.00	0.59	0.25	0.28	0.76	0.50	0.21	1.27
12	0.33	0.00	0.62	0.33	0.04	1.11	0.33	0.12	1.07
13	0.86	0.59	0.87	0.14	0.69	0.92	0.00	0.53	0.94
14	1.00	0.52	0.57	0.00	0.80	0.93	0.00	0.73	1.05
15	1.00	0.70	1.12	0.00	0.57	0.55	0.00	0.33	0.74
16	0.40	0.44	1.23	0.40	0.35	1.14	0.20	0.33	0.95
17	0.67	0.10	0.84	0.33	0.49	1.23	0.00	0.00	0.72
18	0.60	0.41	1.05	0.30	0.51	1.17	0.10	0.39	1.01
19	0.50	1.27	1.24	0.50	1.09	0.83	0.00	0.77	0.77
20	0.60	0.62	1.15	0.40	0.90	0.99	0.00	0.45	0.74
21	0.25	0.53	1.30	0.50	0.32	1.04	0.25	0.45	0.98
22	0.25	0.14	0.77	0.75	0.64	1.29	0.00	0.13	0.88
23	0.60	0.81	1.26	0.00	0.31	0.92	0.40	0.76	0.95
24	1.00	0.97	0.64	0.00	0.89	1.04	0.00	0.93	1.02
25	0.82	0.54	1.07	0.18	0.60	0.88	0.00	0.55	0.83
Average	0.63	0.50	0.98	0.24	0.55	1.00	0.13	0.43	0.95

6.04%, respectively, and for the second objective function, these average errors are 0.40, 1.42 and 1.21%.

In terms of OPT Time, on average, SPEA2 is about 15 and 4% faster than the ENSLS and NSGA-II algorithms, respectively. Also, in terms of TOT Time, on average, SPEA2 is about 16 and 4% faster than the ENSLS and NSGA-II algorithms, respectively. Figures 12 and 13 show the graphs of time to find best solutions and time to completion for every proposed algorithm, respectively, and every individual problem instance.

Examining these graphs shows that SPEA2, ENSLS and NSGA-II have had the best OPT time in 44, 28 and 28% of problem instances, respectively, and the best TOT time in 44, 32 and 24% of problem instances, respectively. Also, as mentioned, SPEA2 has shown the best average performance in terms of both OPT time and TOT time.

Overall, these comparisons show that the proposed algorithm completely outperformed the two other algorithms in terms of solution quality and also has an advantage in terms of solution diversity. In terms of spacing metric, however, SPEA2 showed a better performance. Given the simultaneous importance of solution quality and computational time, we can conclude that the proposed algorithm (ENSLS) performed better than NSGA-II and SPEA2. The graph of non-dominated solutions obtained by the tested algorithms for the instance #23 is presented in Figure 14.

Figure 14 demonstrates the superiority of the proposed method over NSGA-II and SPEA2 in exploring the Pareto front. To evaluate the solution improvement process, the graph of non-dominated solutions of instance #18 after 500, 1000, 2000 and 3000 iterations is plotted in Figure 15.

Figure 15 shows the good performance of the proposed algorithm in solving the problem and the significant improvement of the solutions after 3000 iterations.

6. Conclusions and recommendations for future studies

This paper studied the two-dimensional loading time-dependent vehicle routing problem. As previously mentioned, despite the possible applications of this problem in distribution networks, the literature on simultaneous routing and loading problems lacks proper investigations related to this problem. This paper first introduced this problem and then presented it as a bi-objective mathematical model that incorporates FIFO property for TDVRP. Given the NP-hard nature of this problem, an approach called ENSLS was developed to obtain its solutions. To evaluate the performance of the proposed algorithm in small- and medium-scale problem instances, its results were compared with the results of two other algorithms (SPEA2, NSGA-II). While all algorithms exhibited good performance in solving the small-scale problem instances, the

Table 9 Computational results obtained for medium-scale problem instances (part 2)

Sample	*ENSLS*				*NSGA-II*				*SPEA2*			
	Obj. 1	*Obj. 2*	*OPT*	*TOT*	*Obj. 1*	*Obj. 2*	*OPT*	*TOT*	*Obj. 1*	*Obj. 2*	*OPT*	*TOT*
1	422.25	52.67	4.60	12.61	416.49	52.33	11.54	20.31	422.79	52.33	6.24	14.09
2	908.14	65.33	39.65	62.12	917.11	65.00	56.02	80.20	928.41	65.00	49.48	71.63
3	1066.59	156.00	196.21	240.86	1114.22	156.00	140.50	186.05	1119.60	156.00	122.85	162.05
4	1068.13	28710.67	1195.42	1282.32	1160.80	28712.00	759.88	839.56	1191.72	28713.67	428.63	504.07
5	1781.43	106.67	1129.45	1265.45	1967.87	108.00	1088.98	1215.68	1923.49	106.67	776.93	879.68
6	418.72	53.00	10.78	22.27	425.61	53.00	10.79	22.79	418.84	53.00	14.22	25.86
7	914.75	65.67	99.39	128.75	942.14	65.67	49.16	82.35	944.75	65.33	86.87	116.45
8	1199.57	78.67	390.68	471.66	1270.79	78.33	194.15	281.54	1301.88	76.67	312.44	393.69
9	1035.62	21615.00	920.39	1059.47	1078.75	21629.67	894.51	1076.88	1076.70	21832.00	1189.77	1398.35
10	2089.17	72.33	2835.99	3098.26	2216.29	72.67	2187.91	2358.65	2324.19	71.67	1375.95	1541.20
11	443.99	53.00	5.68	18.29	448.96	53.00	5.61	19.10	446.26	52.33	7.82	19.56
12	914.81	66.00	93.77	134.55	944.12	65.33	78.08	112.84	942.80	65.33	61.50	95.82
13	1193.28	78.00	523.88	613.70	1311.91	81.33	440.92	527.26	1319.17	84.67	329.90	418.01
14	980.46	21619.00	1567.81	1750.98	1052.59	21625.00	1050.68	1248.61	1082.92	21663.33	881.84	1067.82
15	2156.44	79.33	3805.45	4168.88	2338.45	85.67	2720.69	3000.91	2461.62	82.33	3648.31	3952.31
16	422.15	52.00	10.00	22.13	418.65	52.33	16.98	30.00	431.77	52.33	9.61	20.86
17	932.85	65.67	116.22	165.85	963.54	65.33	105.13	153.43	938.47	66.00	144.41	190.10
18	1191.17	73.00	575.64	695.07	1240.07	76.33	741.42	863.55	1273.95	74.00	595.96	707.58
19	966.85	21620.33	1974.75	2153.60	988.39	21621.67	1814.46	1985.67	1016.07	21624.33	1517.31	1718.60
20	1995.35	77.00	4000.15	4251.12	2145.96	79.00	3477.37	3757.46	2196.44	79.00	3643.06	3891.57
21	433.79	52.33	3.86	18.17	424.26	52.67	18.14	33.12	427.43	52.33	7.58	22.20
22	917.89	65.33	82.45	129.93	900.61	64.67	144.04	185.31	929.98	65.67	39.23	90.80
23	1076.49	70.00	466.07	602.43	1107.50	71.67	646.17	767.91	1132.19	69.33	925.77	1042.63
24	857.34	21611.00	3383.34	3650.00	890.61	21611.00	2424.74	2650.37	890.31	21611.00	3474.65	3678.11
25	2106.54	77.33	4495.77	4838.57	2236.74	81.00	5462.48	5721.70	2488.77	84.67	3857.12	4000.25
Average	1099.75	4665.41	1117.10	1234.28	1156.90	4667.15	981.61	1088.85	1185.22	4676.76	940.30	1040.93

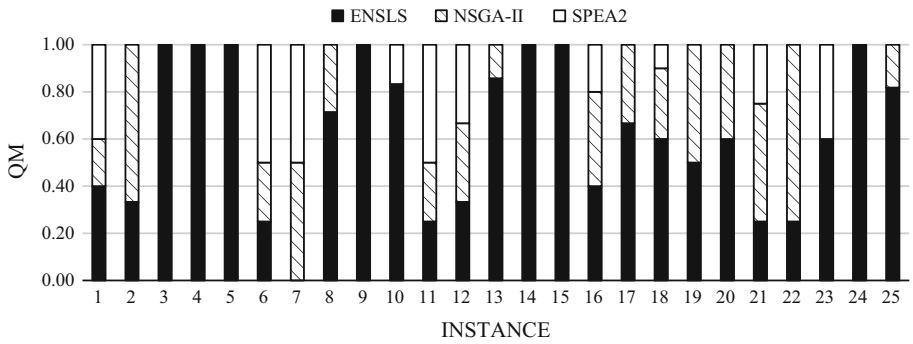

Figure 9 *QM* values obtained for each medium-scale problem instances.

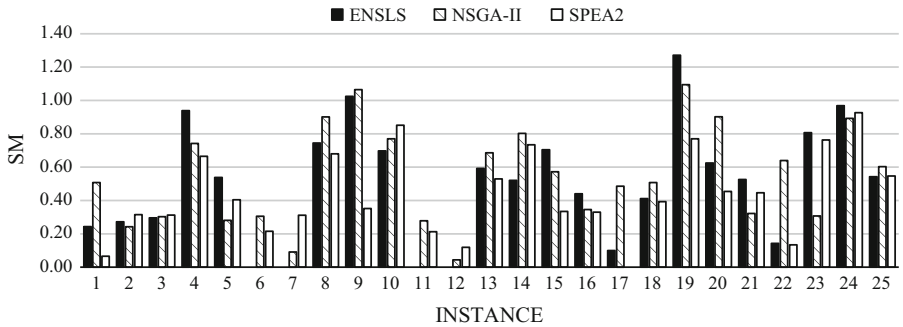

Figure 10 SM values obtained for each medium-scale problem instance.

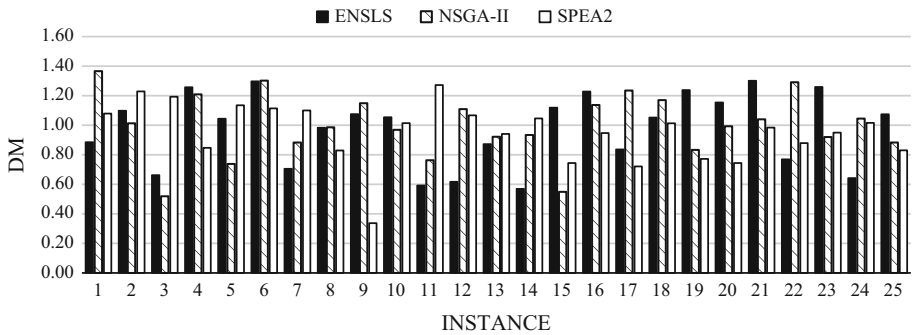

Figure 11 DM values obtained for each medium-scale problem instance.

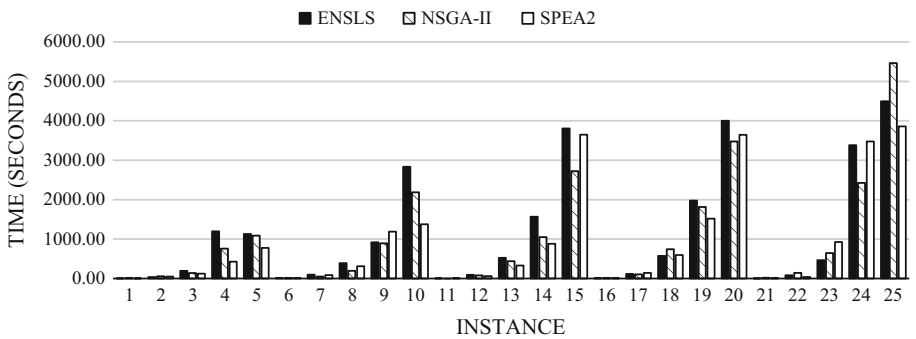

Figure 12 Time to find best solutions elapsed for each medium-scale problem instance.

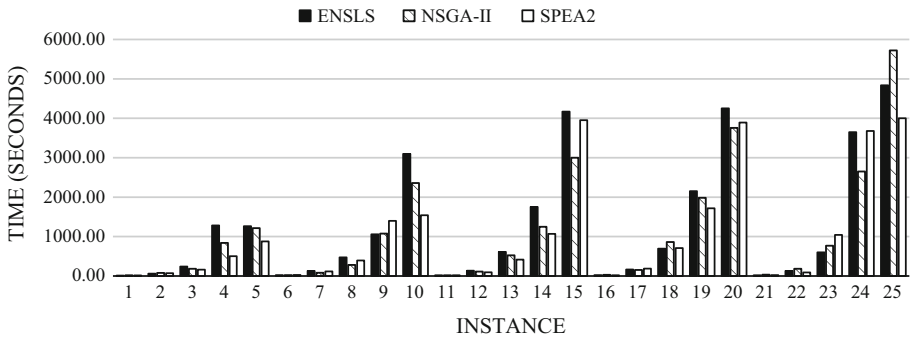

Figure 13 Time to run to completion elapsed for each medium-scale problem instance.

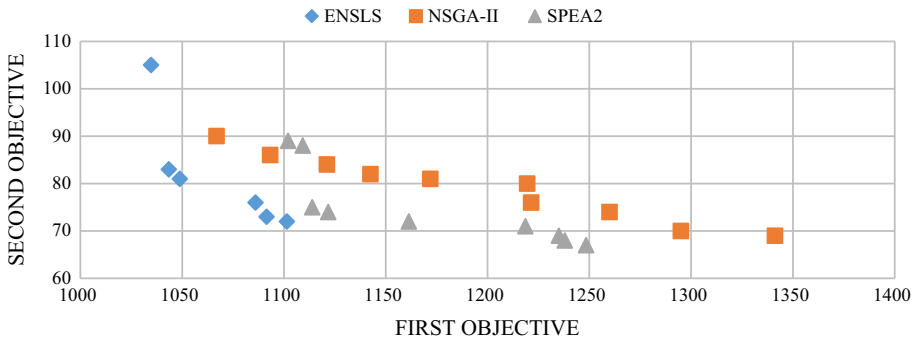

Figure 14 Graph of non-dominated solutions obtained by the tested meta-heuristic algorithms for the instance #23.

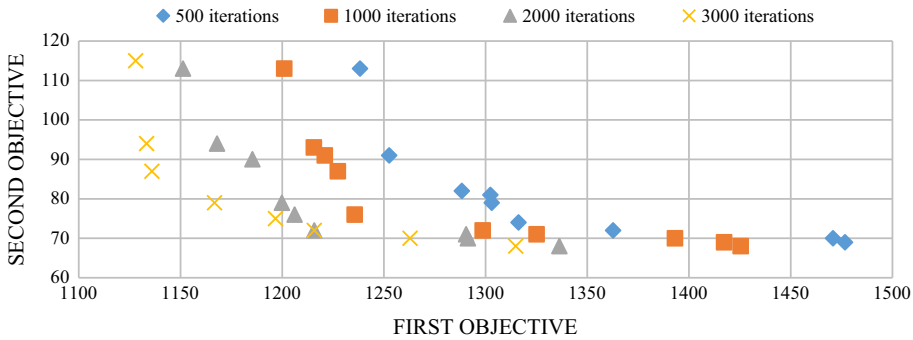

Figure 15 Graph of non-dominated solutions of instance #18 obtained by the ENSLS algorithm after 500, 1000, 2000 and 3000 iterations.

proposed algorithm showed the best performance, since it only had 0.28% error in the first objective function, while errors were 0.34 and 0.48% for NSGA-II and SPEA2, respectively. Also, average time to find best solutions and average time to run to completion for ENSLS were 1.30 and 5.14, respectively. These values were 2.22 and 6.90 for NSGA-II, respectively, and 1.86 and 6.59 for SPEA2, respectively. In the medium-scale problems, the average values of QM, SM, DM, best value of objective 1, best value of objective 2, time to find best solutions and time to run to completion were 0.63, 0.50, 0.98, 1099.75, 4665.41, 1117.10 and 1234.28 for the proposed algorithm, 0.24, 0.55, 1.00, 1156.90, 4667.15, 981.61 and 1088.85 for NSGA-II and 0.13, 0.43, 0.95, 1185.22, 4676.76, 940.30 and 1040.93 for SPEA2. These results show the good performance of the proposed algorithm in solving small-, medium-scale problems instances and therefore its applicability in solving the assessed problem.

The future researches could consider this problem in the scenarios where items demanded by the customers have a non-rectangular cross section. Integrating the features of time-dependent vehicle routing problem with pickup and delivery into loading constraints could also be an interesting field of research. Researchers could also explore other approaches to solve the presented problem with a higher performance and accuracy.

References

Chazelle B (1983). The bottom-left bin-packing heuristic: An efficient implementation. *IEEE Transactions on Computers* **100**(8):697–707.

Clarke GU and Wright JW (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* **12**(4):568–581.

Croes GA (1958). A method for solving traveling-salesman problems. *Operations Research* **6**(6):791–812.

Dantzig G, Fulkerson R and Johnson S (1954). Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America* **2**(4):393–410.

Deb K, Pratap A, Agarwal S and Meyarivan TAMT (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* **6**(2):182–197.

Dominguez O, Juan AA and Faulin J (2014). A biased-randomized algorithm for the two-dimensional vehicle routing problem with and without item rotations. *International Transactions in Operational Research* **21**(3):375–398.

Dominguez O, Juan AA, Barrios B, Faulin J and Agustin A (2016). Using biased randomization for solving the two-dimensional loading vehicle routing problem with heterogeneous fleet. *Annals of Operations Research* **236**(2):383–404.

Donati AV, Montemanni R, Casagrande N, Rizzoli AE and Gambardella LM (2008). Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research* **185**(3):1174–1191.

Duhamel C, Lacomme P, Quilliot A and Toussaint H (2011). A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle routing problem. *Computers & Operations Research* **38**(3):617–640.

Figliozzi MA (2010). The impacts of congestion on commercial vehicle tour characteristics and costs. *Transportation Research Part E: Logistics and Transportation Review* **46**(4):496–506.

Figliozzi MA (2012). The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E: Logistics and Transportation Review* **48**(3):616–636.

Fuellerer G, Doerner KF, Hartl RF and Iori M (2009). Ant colony optimization for the two-dimensional loading vehicle routing problem. *Computers & Operations Research* **36**(3):655–673.

Gendreau M, Iori M, Laporte G and Martello S (2008). A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. *Networks* **51**(1):4–18.

Golden BL, Magnanti TL and Nguyen HQ (1977). Implementing vehicle routing algorithms. *Networks* **7**(2):113–148.

Grefenstette J, Gopal R, Rosmaita B and Van Gucht D (1985). Genetic algorithms for the traveling salesman problem. In: *Proceedings of the first International Conference on Genetic Algorithms and their Applications* (pp. 160–168). Lawrence Erlbaum: New Jersey.

Hamdi-Dhaoui K, Labadie N and Yalaoui A (2014). The bi-objective two-dimensional loading vehicle routing problem with partial conflicts. *International Journal of Production Research* **52**(19):5565–5582.

Hill AV and Benton WC (1992). Modelling intra-city time-dependent travel speeds for vehicle scheduling problems. *The Journal of the Operational Research Society* **43**(4):343–351.

Holland JH (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. U Michigan Press: Ann Arbor.

Ichoua S, Gendreau M and Potvin JY (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research* **144**(2):379–396.

Iori M and Martello S (2010). Routing problems with loading constraints. *TOP* **18**(1):4–27.

Iori M and Martello S (2013). An annotated bibliography of combined routing and loading problems. *Yugoslav Journal of Operations Research* **23**(3):311–326.

Iori M, Salazar-González JJ and Vigo D (2007). An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science* **41**(2):253–264.

Jabali O, Woensel T and de Kok AG (2012). Analysis of travel times and CO_2 emissions in time-dependent vehicle routing. *Production and Operations Management* **21**(6):1060–1074.

Khebbache-Hadji S, Prins C, Yalaoui A and Reghioui M (2013). Heuristics and memetic algorithm for the two-dimensional loading capacitated vehicle routing problem with time windows. *Central European Journal of Operations Research* **21**(2): 307–336.

Kumar SN and Panneerselvam R (2012). A survey on the vehicle routing problem and its variants. *Intelligent Information Management* **4**(3):66–74.

Leung SC, Zhou X, Zhang D and Zheng J (2011). Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem. *Computers & Operations Research* **38**(1):205–215.

Leung SC, Zhang Z, Zhang D, Hua X and Lim MK (2013). A meta-heuristic algorithm for heterogeneous fleet vehicle routing problems with two-dimensional loading constraints. *European Journal of Operational Research* **225**(2):199–210.

Lin S (1965). Computer solutions of the traveling salesman problem. *The Bell System Technical Journal* **44**(10):2245–2269.

Lodi A, Martello S and Vigo D (1999). Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing* **11**(4):345–357.

Malandraki C (1989). *Time Dependent Vehicle Routing Problems: Formulations, Solution Algorithms and Computational Experiments*. Northwestern University: Evanston.

Malandraki C and Daskin MS (1992). Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science* **26**(3):185–200.

Malapert A, Guéret C, Jussien N, Langevin A and Rousseau LM (2008). Two-dimensional pickup and delivery routing problem with loading constraints. In: *First CPAIOR Workshop on Bin Packing and Placement Constraints (BPPC'08)*, Paris, France.

Oliver IM, Smith D and Holland JR (1987). Study of permutation crossover operators on the traveling salesman problem. In: *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms: July 28–31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*. L. Erlhaum Associates: Hillsdale, NJ.

Puljić K and Manger R (2013). Comparison of eight evolutionary crossover operators for the vehicle routing problem. *Mathematical Communications* **18**(2):359–375.

Roy RK (2010). *A Primer on the Taguchi Method*. Society of Manufacturing Engineers: Mich, Dearborn.

Soler D, Albiach J and Martínez E (2009). A way to optimally solve a time-dependent vehicle routing problem with time windows. *Operations Research Letters* **37**(1):37–42.

Strodl J, Doerner KF, Tricoire F and Hartl RF (2010). On Index Structures in Hybrid Metaheuristics for Routing Problems with Hard Feasibility Checks: An Application to the 2-Dimensional Loading Vehicle Routing Problem. In: *International Workshop on Hybrid Metaheuristics* (pp. 160–173). Springer: Berlin.

Toth P and Vigo D (2002). The Vehicle Routing Problem. In *SIAM Monographs on Discrete Mathematics and Applications*. Vol. 9. Philadelphia: Siam.

Waters CDJ (1987). A solution procedure for the vehicle-scheduling problem based on iterative route improvement. *Journal of the Operational Research Society* **38**(9):833–839.

Zachariadis EE, Tarantilis CD and Kiranoudis CT (2009). A guided tabu search for the vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research* **195**(3):729–743.

Zachariadis EE, Tarantilis CD and Kiranoudis CT (2013). Integrated distribution and loading planning via a compact metaheuristic algorithm. *European Journal of Operational Research* **228**(1):56–71.

Zhang T, Chaovalitwongse WA and Zhang Y (2014). Integrated ant colony and tabu search approach for time dependent vehicle routing problems with simultaneous pickup and delivery. *Journal of Combinatorial Optimization* **28**(1):288–309.

Zitzler E (1999). *Evolutionary algorithms for multiobjective optimization: Methods and applications*. Shaker: Ithaca.

Zitzler E, Laumanns M and Thiele L (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In: *Eurogen* (Vol. 3242, pp. 95–100). Swiss Federal Institute of Technology: Zurich.

Received 5 February 2016;
accepted 8 November 2016