Applied Soft Computing xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

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

The generalized covering traveling salesman problem

Q1 Mohammad H. Shaelaie[a], Majid Salari[a,*], Zahra Naji-Azimi[b]

[a] Department of Industrial Engineering, Ferdowsi University of Mashhad, Mashhad, Iran
[b] Department of Management, Ferdowsi University of Mashhad, Mashhad, Iran

ARTICLE INFO

Article history:
Received 20 March 2013
Received in revised form 25 August 2014
Accepted 25 August 2014
Available online xxx

Keywords:
Traveling salesman problem
Humanitarian relief transportation
Memetic algorithm
Variable neighborhood search

ABSTRACT

In this paper we develop a problem with potential applications in humanitarian relief transportation and telecommunication networks. Given a set of vertices including the depot, facility and customer vertices, the goal is to construct a minimum length cycle over a subset of facilities while covering a given number of customers. Essentially, a customer is covered when it is located within a pre-specified distance of a visited facility on the tour. We propose two node-based and flow-based mathematical models and two metaheuristic algorithms including memetic algorithm and a variable neighborhood search for the problem. Computational tests on a set of randomly generated instances and on set of benchmark data indicate the effectiveness of the proposed algorithms.

1. Introduction

The traveling salesman problem (TSP) is one of the most studied problems in the field of combinatorial optimization. Given a graph $G=(N, E)$, the TSP is to find a minimum length Hamiltonian cycle of the vertices in N in which each vertex has to be visited exactly once [13].

The quota traveling salesman problem (QTSP) is a generalization of the TSP in which the salesman has to visit a given quota of vertices while minimizing the total travel cost [4,10]. In this problem, each vertex has a pre-determined amount of prize that could be collected by the salesperson by being visited on the tour. The k-TSP is a special case of the QTSP in which each vertex has one unit of prize and the goal is to visit k vertices, in order to collect k units of prize [15,30].

Introduced by Balas in 1989 [5], in the prize collecting traveling salesman problem (PCTSP) it is given a set of vertices in which each vertex has a pre-determined amount of prize. The goal of the PCTSP is to collect a certain amount of prize, while minimizing the routing and penalty costs. Essentially, the routing cost is the total distance traveled by the salesman and the penalty cost is the cost occurred by not visiting a vertex on the tour [5,9].

Many variations of the standard TSP are introduced in the literature including the online TSP [38], the TSP with pickup-and-delivery [42], clustered TSP [6], generalized TSP [27,29], multi-depot multiple TSP [17] and etc. To the complete survey on the TSP and its variants, we refer the interested readers to the book by Gutin and Punnen [19].

There is a vast body of literature dedicated to covering problems. The purpose of such problems is to satisfy the customers' demand in two different ways. Essentially, different customers' demand can be provided at the same location at which they are located or it can be delivered at a destination within a pre-specified distance of the customer locations [41].

Several variations of the TSP with potential applications in route designing of the healthcare teams in developing countries [16], and applications from design of the distributed networks [21] have been introduced. In these two problems and some others which are proposed in the literature, there is no need to visit all the customers on the tour structure and the objectives can be satisfied by visiting a limited number of customers. As an example we can refer to the case of emergency management in which it is not possible for the medical team to visit all the customers on the tour. In such situation, a subset of customers can be visited on the tour and the rest of them are covered, i.e., located within a pre-specified distance of at least on visited customer.

The covering salesman problem (CSP) is a generalization of the TSP in which we have to satisfy all the customers' demand by visiting or covering them [11]. Essentially, for each customer, say i, it is given a covering radius d_i, within that all the located customers will be covered. The goal of the CSP is to construct a minimum length tour over a subset of the given customers such that each customer, not visited on the tour, is within the covering distance of at least one

* Corresponding author. Tel.: +98 511 8805113.
E-mail addresses: hesam_shaelaie@stu.um.ac.ir (M.H. Shaelaie), msalari@um.ac.ir, salarimajid@yahoo.com (M. Salari), znajiazimi@um.ac.ir (Z. Naji-Azimi).

http://dx.doi.org/10.1016/j.asoc.2014.08.057
1568-4946/© 2014 Published by Elsevier B.V.

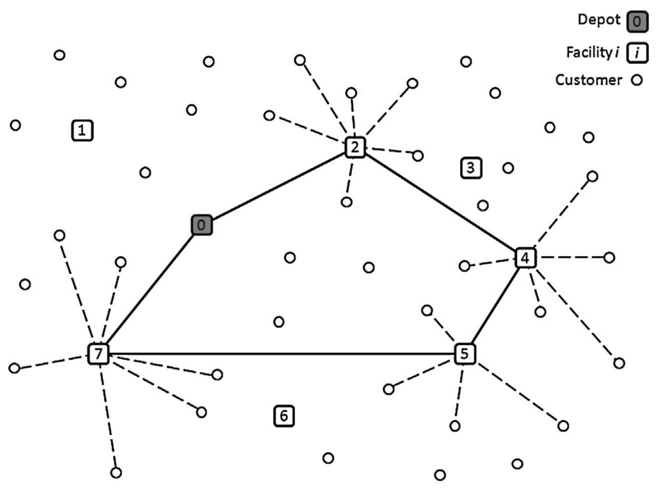

Fig. 1. An illustrative example of the problem.

visited customer [11]. Many papers have discussed on the application of this problem and some of its generalization in the fields of emergency management and disaster planning [1,34,35,37].

What happens in reality when satisfying a given set of customers' demand, is somehow different from the assumptions made in many problems, introduced in the literature. As an example, usually the distribution centers cannot be constructed at the same location at which the customer zones are located. Essentially, there can be a situation in which it is not possible for the health care team to assess the customer zone by vehicles. So, they need to locate the facilities in a location different from the customer zone. Moreover, in the QTSP case the customers can travel to a facility location to receive their demands.

Taking into account the above assumptions, in this paper we propose a generalization of the TSP with applications in humanitarian relief transportation. Suppose we are given a set of vertices including a central depot, facility and customer vertices. Each customer i has one unit of demand (prize) that will be satisfied when it is located within a certain coverage radius of at least one facility j visited by the tour. Essentially, each facility j has a coverage radius d_j within that all the available customers are covered. The goal of the problem is to construct a minimum length Hamiltonian cycle over a subset of facilities, while collecting the specified amount of prize P.

The introduced problem is a generalization of the CSP, TSP, k-TSP and the QTSP and all applications already introduced for these problems could be easily extended to the new developed problem. Letting $P=21$, Fig. 1 represents an example in which 4 facilities have been visited by the tour and 21 customer vertices are covered by the visited facilities.

The paper is organized as follows. Section 2 develops a formal description of the problem. The details of the proposed solution methods are provided in Section 3. Extensive computational tests for the studied problem and one of its variants (CSP) are reported in Section 4, followed by conclusion and future directions given in Section 5.

2. Problem modeling

The problem is defined on a directed graph $G=(V, A)$, where $V=\{0\}\cup W\cup T$ contains three sets of vertices including the depot, $\{0\}$, the set of customers, W, and the set of facilities, T. Moreover, the set of arcs is given by $A=\{(i,j)|i,j\in T\cup\{0\}\}$. To each $(i,j)\in A$ is associated a travel cost represented by t_{ij} and each customer i has a prize, p_i, that will be covered by the tour when it is located within the predefined coverage distance of at least one routed facility. We denote by T_i those facilities that are able to cover customer i. The goal of the problem is to collect a pre-determined amount of prize, P, while minimizing the total traveled length.

The decision variables to model the problem are defined as follows:

$$x_{ij}=\begin{cases}1 & \text{if arc } (i,j) \text{ is visited by the tour,}\\ 0 & \text{otherwise.}\end{cases} \quad \forall (i,j)\in A \tag{1}$$

$$z_{ij}=\begin{cases}1 & \text{if the demand of customer i is allocated to the facility j,}\\ 0 & \text{otherwise.}\end{cases} \quad \forall i\in W, j\in T \tag{2}$$

Finally, for each $i\in T$, u_i is a continuous variable representing the tour load before visiting vertex i.

In what follows, we extend two mathematical models for the studied problem. Based on the formulations developed for the TSP and its variants, we divide the models into two groups namely, node-based and flow-based formulations.

2.1. Node-based formulation

In this section, we develop a node-based formulation for the studied problem. This kind of formulation, applies the well known Miller–Tucker–Zemlin subtour elimination constraints proposed for the TSP [14,25,31].

$$\min \sum_{(i,j)\in A} t_{ij}x_{ij} \tag{3}$$

subject to:

$$\sum_{i\in W}\sum_{j\in T_i} p_i z_{ij} \geq P \tag{4}$$

$$\sum_{j\in T_i} z_{ij} \leq 1 \quad \forall i\in W, \tag{5}$$

$$\sum_{j\in T\cup\{0\}} x_{0j} = 1 \tag{6}$$

$$\sum_{j\in T\cup\{0\}} x_{ij} = \sum_{j\in T\cup\{0\}} x_{ji} \quad \forall i\in T\cup\{0\}, \tag{7}$$

$$z_{ij} \leq \sum_{k\in T\cup\{0\}} x_{kj} + \sum_{k\in T\cup\{0\}} x_{jk} \quad \forall i\in W, j\in T_i, \tag{8}$$

$$u_i - u_j + (|T|+1)X_{ij} \leq |T| \quad \forall i,j\in T, \tag{9}$$

$$x_{ij}\in\{0,1\} \quad \forall i,j\in T\cup\{0\}, \tag{10}$$

$$z_{ij}\in\{0,1\} \quad \forall i\in W, j\in T_i, \tag{11}$$

$$u_j \geq 0 \quad \forall j\in T\cup\{0\}. \tag{12}$$

The objective function (3) is to minimize the total traveling cost. Constraint (4) assures the total collected prize, by the eligible facilities, to be greater than or equal to the pre-specified value P. For each $i\in W$, constraint (5) assures customer i to be allocated to at most one facility $j\in T_i$. Constraint (6) makes sure that the tour to be started from the depot, followed by the in-degree and out-degree constraints (for each $i\in T\cup\{0\}$) represented by constraints set (7). For each customer i and facility $j\in T_i$, constraint (8) shows that, i could be allocated to j only if j is visited by the tour. Constraint set (9) models the sub-tour elimination constraints. Finally, constraints (10)–(12) define the model variables.

2.2. Flow-based formulation

Several papers have reported successful results of applying flow-based formulations for different routing problems [26,39]. In this section we adapt this kind of formulation to be used for our developed problem. To do so, for each $i, j \in T \cup \{0\}$ a set of continuous variables are introduced as follows.

u_{ij} is a continuous variable, representing the tour load after leaving vertex i and just before visiting vertex j. Based on the description of the new introduced variables (u_{ij}), the model reads as follows.

$$\min \sum_{(i,j)\in A} t_{ij}x_{ij} \tag{13}$$

subject to:

$$\sum_{i\in W}\sum_{j\in T_i} p_i z_{ij} \geq P \tag{14}$$

$$\sum_{j\in T_i} z_{ij} \leq 1 \quad \forall i \in W, \tag{15}$$

$$\sum_{j\in T\cup\{0\}} x_{0j} = 1 \tag{16}$$

$$\sum_{j\in T\cup\{0\}} x_{ij} = \sum_{j\in T\cup\{0\}} x_{ji} \quad \forall i \in T \cup \{0\}, \tag{17}$$

$$z_{ij} \leq \sum_{k\in T\cup\{0\}} x_{kj} + \sum_{k\in T\cup\{0\}} x_{jk} \quad \forall i \in W, j \in T_i, \tag{18}$$

$$\sum_{j\in T} u_{0j} = \sum_{i\in W}\sum_{j\in T_i} z_{ij} \tag{19}$$

$$\sum_{j\in T\cup\{0\}} u_{ji} - \sum_{j\in T\cup\{0\}} u_{ij} = \sum_{j\in W} z_{ji} \quad \forall i \in T, \tag{20}$$

$$\sum_{j\in T} u_{j0} = 0 \tag{21}$$

$$u_{ij} \leq |W| x_{ij} \forall i, j \in T \cup \{0\}, \tag{22}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in T \cup \{0\}, \tag{23}$$

$$z_{ij} \in \{0, 1\} \quad \forall i \in W, j \in T_i, \tag{24}$$

$$u_{ij} \geq 0 \quad \forall i, j \in T \cup \{0\}. \tag{25}$$

The objective function (13) and the constraints (14)–(18) are the same as those already explained for the node-based formulation (i.e., (3)–(8), respectively). Constraints (19)–(21) are the subtour elimination constraints. Essentially, constraint (19) assures that the initial tour load, before leaving the depot, to be equal to the total prize that will be collected by the tour. Constraint (20) makes sure that the load difference of the tour before and after visiting a facility, say i, is exactly equal to the prize allocated to the facility i. Constraint (21) sets the total load of the vehicle, when returning to the depot, to 0. For each $(i, j) \in A$, constraint (22) models the relation between the routing (x_{ij}) and the flow (u_{ij}) variables. Essentially, we are not allowed to send flow along arc $(i, j) \in A$ when its corresponding variable x_{ij} is not visited by the tour. Finally, decision variables are given in (23)–(25).

3. Solution method

Two metaheuristic algorithms, namely memetic algorithm (MA) and variable neighborhood search (VNS) are developed for the introduced problem. In the following, the details of these algorithms are provided.

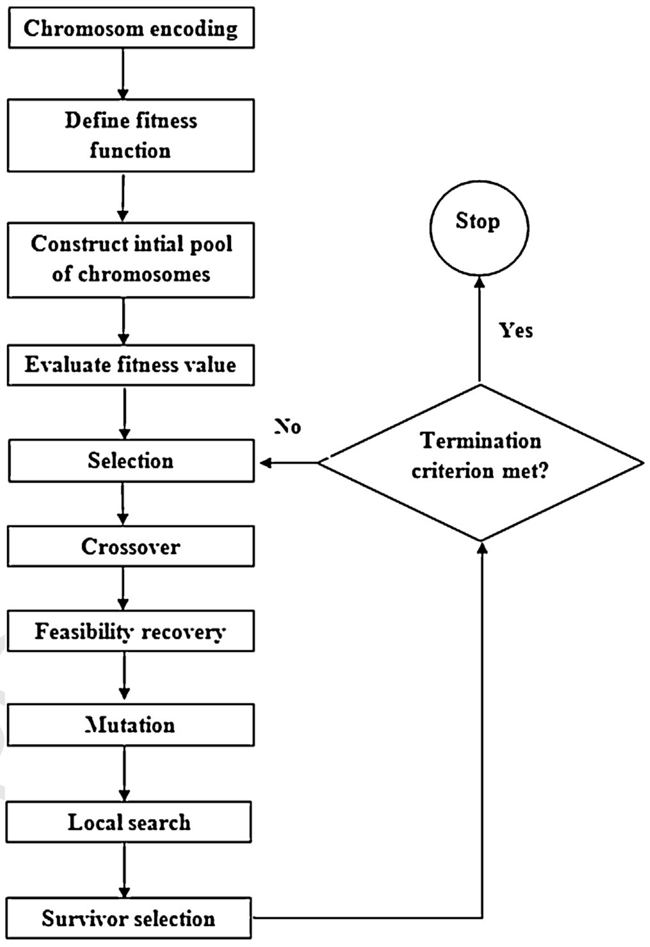

Fig. 2. The flow chart of the MA.

3.1. Memetic algorithm

The genetic algorithm (GA) is a global search optimization technique, developed by Holland [22], with effective results in different areas including business, science, and engineering [12,24,28,40]. MA is the GA hybridized with the local search ideas [33]. Several variants of the TSP have been solved by MA with very good results. For example we can refer to the works by Bontouxa et al. [7], Buriol et al. [8] and Gutin and Karapetyan [20].

In this section we develop a MA for the studied problem which its flow chart is depicted in Fig. 2. It starts by constructing the initial population set containing a set of N_p chromosomes (Section 3.1.2). At each iteration, the *crossover* (Section 3.1.5) is applied over a set of individuals selected during the *selection* phase as described in Section 3.1.4. Since the generated offspring could be infeasible, the *feasibility recovery* procedure is developed to make the solutions feasible for which the details are provided in Section 3.1.6. Upon this step, the *mutation* and *local search* phases are applied to the offspring (see Sections 3.1.7–3.1.8, respectively) and the repeated solutions are eliminated from the set of new generated offspring. Finally, the survivors are selected (Section 3.1.9) and the algorithms iterates as long as the best solution has not been improved for γ successive iterations. In what follows, the details of the MA's different procedures are provided in details.

3.1.1. Chromosome representation

Permutation representation is used to the genetic representation of the solutions. Using this kind of representation, each chromosome contains an ordered set of facilities and the depot

Number	1	2	3	4
Facility	2	4	5	7

Fig. 3. Example of chromosome representation.

which are visited by the tour. Fig. 3 provides the representation of the solution shown in Fig. 1, where totally four facilities are visited by the tour.

3.1.2. Population structure and initialization

A set of N_p individuals is generated by following four different procedures developed to this purpose.

- *Construction rule 1*: the algorithm applies a randomized variation of the nearest neighbor algorithm, proposed for the traveling salesman problem [3]. Starting from the depot, at each step of this procedure the next facility to be visited by the tour is randomly selected from those that are among the three nearest facilities of the last visited vertex covering at least one unit of unsatisfied demand. The algorithm keeps adding the facilities to the tour, as long as the total collected prize is less than P. Finally, the tour will be terminated by visiting the depot as the last vertex of the tour.
- *Construction rule 2*: within this procedure, the facilities are selected from those that are able to cover the largest number of uncovered customers. Starting from the depot, the next vertex to be visited by the tour is randomly selected, from the three facilities that are able to cover the largest number of uncovered customers. Upon adding the selected facility to the tour, the coverage of customers will be updated and the process continues until at least P units of prize is collected by the tour. The Hamiltonian tour will be terminated by visiting the depot as the last vertex of the tour.
- *Construction rule 3*: in this procedure an initial solution is generated by combining the first two methods. Essentially, at each step the new added facility is selected from those with the minimum distance from the last visited facility or those which are able to cover the largest number of uncovered customers. To do so, as long as the solution is infeasible the new facilities are added to the tour by following rules 1 and 2 in turn.
- *Construction rule 4*: the Hamiltonian cycle is constructed by selecting the facilities, randomly. Particularly, as long as the solution is infeasible, a facility is randomly selected, from those unvisited facilities that are able to cover at least one unit of unsatisfied demand. At each iteration, the new facility is visited in the last position of the tour.

Finally, 50, 30, 15 and 5 percent of the Np individuals, are generated by following rules 1 to 4, respectively.

3.1.3. Fitness value

According to the objective function definition, the fitness of each individual is calculated by using the objective function (3) which is the summation of the total length (time) traveled by the tour.

3.1.4. Selection

By applying the *selection* phase, we aim at selecting a set of K individuals to perform the crossover and mutation. To do so, we propose three rules in order to select the individuals.

- *Roulette wheel selection*: the roulette wheel selection is applied to select the individuals. Essentially, the probability of selecting the individuals is directly proportional to their fitness values. Letting f_i to be the fitness value of each indivisual i, the probability of selecting i is as follows:

$$p_i = \frac{f_i}{(\sum_{i=1}^{Np} f_i)} \tag{26}$$

- *Random selection*: by following this rule, all individuals have the same probability to be selected. Particularly, the probability for selecting each individual is $1/Np$.
- *Best selection*: the chromosomes are listed based on their fitness value and the first K individuals, i.e., those having the best fitness function, are selected.

3.1.5. Crossover

To do the crossover, two chromosomes have to be chosen from the set of K individuals selected within the *selection* phase. To select such two chromosomes, we propose to use the following rules:

- *Crossover rule 1*: to do the crossover we always select the chromosome with the best fitness value as one of the parents and the next is selected from the remaining $K-1$ individuals. As a result, totally $K-1$ pairs of chromosomes are evaluated using this rule.
- *Crossover rule 2*: two parent chromosomes are selected uniformly randomly. So, a set of $K/2$ pairs will be available by following this rule.

Upon selecting the parents, we apply one point move crossover to obtain the offspring. To do so, a position along each chromosome string is selected, randomly, and the substrings of the parents will be exchanged to generate the children. An example is represented in Fig. 4.

3.1.6. Feasibility recovery

The offspring, resulted from the crossover may be infeasible. In order to work with the feasible chromosomes, a simple procedure is developed (see Algorithm 1, that makes the offspring feasible. Essentially, the infeasibility may occur because of two problems: there could be some facilities visited more than once or there could be a situation in which the total number of covered customers, by the available facilities in the tour structure, is less than P. To generate a feasible solution from an infeasible one, the extra facilities are removed from the tour and while the solution is infeasible, by not being able to cover at least P units of customers' demand, a set of promising facilities are added to the tour to recover the feasibility. Essentially, the promising facilities are those that are able to cover the maximum number of uncovered customers.

Algorithm 1. Feasibility recovery.

```
Starting from the first facility visited by the tour, extract all illegal
  facilities, i.e., those that are visited more than once;
While(The solution is not feasible)do
  $j$:= Select an unvisited facility that by adding to the tour, covers the
  maximum number of uncovered customers;
  Add the facility $j$ to its best position, i.e., the position that leads to the
  minimum extra insertion cost;
end
```

3.1.7. Mutation

Each visited facility in a given chromosome has a probability β to be extracted from the solution and being substituted with a set of other potential facilities. Essentially, upon extracting a facility, say j, from the tour, a facility will be randomly selected from the θ unvisited nearest ones to the j and is added to its best position in the tour, i.e the position leading to the minimum extra insertion cost. As long as the solution is not feasible, this procedure continues adding the facilities to the tour by randomly selecting the facilities. If the procedure is not able to recover the feasibility by adding all

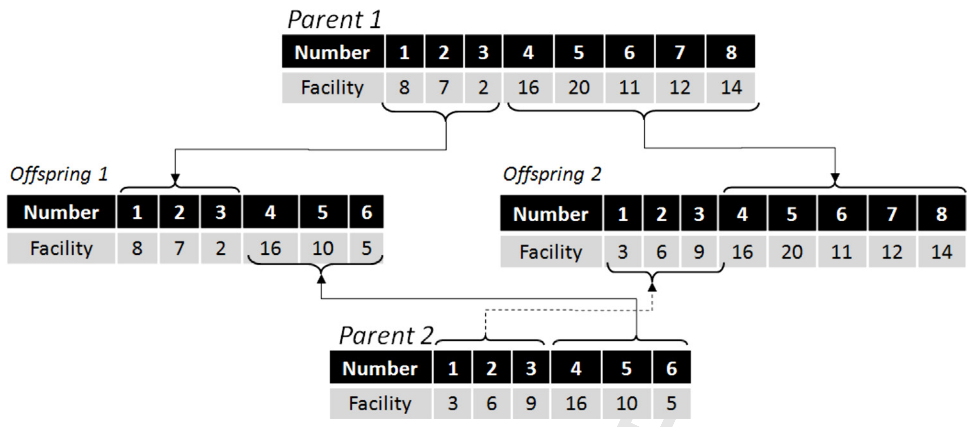

Fig. 4. An example of the crossover procedure.

θ facilities, the original extracted facility j is put back into its original place and the procedure continues by applying this process to the next routed facility. The general framework of the mutation procedure is provided in Algorithm 2.

3.1.8. Local search

In order to intensify the search, two kinds of move, namely *2-opt* and *drop-and-add* procedures, are developed. By applying the *2-opt* procedure, the goal is to improve the tour length by changing the order of the facilities visited by the tour. Within the *drop-and-add* procedure, a facility is extracted from its location in the tour. In case of feasibility and improving the objective function, the extracted facility will not be reinserted in the tour, otherwise, the algorithm searches for the best feasible substitution from the N facilities nearest to the extracted one with the minimum extra insertion cost. Essentially, all sequences consisting of one or two facilities are generated and the best feasible substitution leading to an improvement in the objective function will be implemented, otherwise, in case of having a worse solution, the extracted facility is put back into its original place. In a random order, both *2-opt* and *drop-and-procedures*, are applied until we have no improvement in the tour length.

Algorithm 2. Mutation procedure.

```
J:= The set of randomly ordered facilities visited in the (CurrentSolution);
j:= 0;
While ($j \le |J|$)
  rand:= Randomly generate a number in [0, 1];
  if (rand $\le \beta$) then
    Extract the jth facility of J;
    L:= The set of $\theta$ randomly ordered facilities nearest to j;
    l:= 1;
    NewSolution:= CurrentSolution;
    While ($l \le \theta$ or the solution is not feasible) do
      NewSolution:= Select facility l from the set L and add it to its best
  position in the NewSolution;
      l:= l + 1;
    end
    if (the new solution is feasible) then
      CurrentSolution:= NewSolution;
    end
  end
  j:= j + 1;
end
```

3.1.9. Survivors' selection

The set of N_p initial chromosomes and the new generated children are sorted in a non-decreasing order of the fitness value. To obtain the next generation, the set of both N_p initial and children chromosomes are divided into three groups. Essentially, the first group (A) contains 50% of those chromosomes having the best fitness value in the ordered lists. The second (B) and third (C) groups contain the next 30% and 20% of the chromosomes from the ordered lists, respectively. Upon this procedure, the group A of children chromosomes are substituted with those having the worst fitness value in group A of initial chromosomes. This process is repeated for the next two groups as well. Within this procedure, the substitutions are done in a way that the population size (N_p), will remain fixed.

3.2. Variable neighborhood search

Proposed by Mladenović and Hansen [32], VNS is a metaheuristic algorithm that explicitly applies a strategy based on dynamically changing neighborhood structures. The general framework of the VNS algorithm for the introduced problem is given in Algorithm 3. The algorithm starts to construct an initial solution (*CurrentSolution*) by applying the *construction rule 1* (see Section 3.1.2). It then improves upon this initial solution by applying the *2-Opt* and *drop-and-add* procedures (see Section 3.1.8). The improvement of the solution continues in a loop until the termination criteria is met which is a given number of iterations (ρ) without having an improvement in the cost of the best known solution. The loop contains a *shaking* procedure and the local search procedures. If a solution is improved by applying the shaking and local search procedures it will be accepted as the incumbent solution. We also utilize the *threshold accepting* criterion in updating the current solution at the end of each iteration of the VNS algorithm. Essentially, a worse solution is accepted as the current solution if it is not worse than a given percentage (λ) of the cost of the best known solution. At each iteration of the VNS algorithm we apply the *shaking* procedure in order to dynamically expand the neighborhood structure. In particular, the algorithm generates a solution which is in the neighborhood size k of the *CurrentSolution*, i.e., *Shaking*(*CurrentSolution*, k). Specifically, *Shaking*(*CurrentSolution*, k) is defined as the set of solutions that can be obtained from the *CurrentSolution* by calling the *addition* or *removal* procedures for k times. The details of these procedures are given in the following:

- *Removal procedure*: routed facilities on the tour of the solution are sorted in a non-increasing order of a score which is defined in (27). In this relation *RemovalCost*(i) is the cost saving occurred by removing facility i form the solution while *RemovalCover*(i) is the number of customers that are explicitly covered by facility i.

$$R_{remove}(i) = \frac{RemovalCost(i)}{(RemovalCover(i) + 1)} \quad \textit{for each routed facility } i \tag{27}$$

UNCORRECTED PROOF

Following this step, the first α facilities of the ordered set are considered as the candidate list. Finally, a facility is selected from the candidate list randomly and is removed from the solution.

- *Addition procedure*: unrouted facilities are sorted in a non-increasing order of their score values which is defined in (28). In this relation *AdditionCost*(i) is the cost imposed to the tour length by adding facility i into its best position in the tour, i.e., the location with the cheapest extra insertion cost. Moreover, *AdditionCover*(i) is the total number of uncovered facilities that will be covered by visiting facility i in the solution.

$$R_{add}(i) = \frac{AdditionCover(i)}{(AdditionCost(i)+1)} \quad \textit{for each unrouted facility } i \tag{28}$$

Following this step, the first δ facilities yielding the maximum score are considered in the candidate list from which one facility is selected randomly and added into its best position in the current tour.

Following this step the obtained solution could be infeasible by not covering the pre-specified number of customers. As long as the solution is infeasible, an unrouted facility which is able to cover at least one uncovered customer is selected randomly and added into its best position in the tour.

Algorithm 3. The general framework of the VNS algorithm.

```
CurrentSolution:= Initialization();
CurrentSolution:= 2-Opt(CurrentSolution);
CurrentSolution:= drop-and-add (CurrentSolution);
BestSolution:= CurrentSolution;
Iter:=1;
While (Iter $\leq \rho$) do
  k = 1;
  While ($k \leq \phi$) do
    CurrentSolution : = Shacking(CurrentSolution, k);
    CurrentSolution:= 2-Opt(CurrentSolution);
    CurrentSolution:= drop-and-add (CurrentSolution);
    if (Cost(CurrentSolution) < Cost(BestSolution)) then
      BestSolution = CurrentSolution;
      k:=1;
      Iter:=1;
      if (Iter > $\rho$) then
        break;
      end
    end
    else if (Cost(CurrentSolution) < $(1+\lambda)$Cost(BestSolution))then
      Iter:= Iter + 1;
      k:=k+1;
    end
    else
      CurrentSolution:= BestSolution;
      Iter:= Iter + 1;
      k:=k+1;
      if (Iter > $\rho$) then
      break;
      end
    end
  end
end
```

4. Computational experiments

4.1. Instances

To test the performance of the developed algorithms a set of 115 instances are designed, derived from TSP library [36]. The generated data ranging in size from 51 to 1000 vertices which are divided into small ($|V| \leq 76$), medium ($100 \leq |V| \leq 200$) and large size ($535 \leq |V| \leq 1000$) instances. According to the developed problem characteristics, the set of vertices is divided into three groups. The first vertex in each datum is considered to be the depot and the rest of the vertices are divided into facility and customer vertices. From each instance, we have generated three new data in which 30, 40 or 50 percent of the vertices (V) are considered to be the facility vertices, respectively. Moreover, we assume that each facility $j \in T$ to be able to cover its NC nearest customers. In our designed instances, NC is a random value taken from [1,10]. Finally, from each instances three new data are designed for which 50, 75 and 100 percent of the customers have to be covered, respectively. Moreover, a set of 27 test instances are generated having 100, 150 or 200 vertices. To generate this set of test instances, we follow the same rules already explained for the original data.

Table 1
Parameters tuning.

Algorithm	Parameter	Different tested values	Best value
MA	Np	{100, 120, 140}	140
	Nc	{20, 28}	28
	β	{0.05, 0.08, 0.11}	0.05
	θ	{4, 6, 8}	6
	N	{10, 12, 14}	14
	γ	{80, 100, 120}	100
VNS	ρ	{500, 1000, 2000}	1000
	ϕ	{30, 40, 50}	30
	α	{3, 6, 9}	9
	δ	{5, 10, 15}	5
	N	{10, 12, 14}	14
	λ	{0, 0.005, 0.01, 0.015}	0.015

4.2. Implementation and parameter tuning

The metaheuristic algorithms have been implemented in C++ and tested on PC with an Intel Core i5 processor running at 2.27 GHz with 4 GB RAM. A simple test has been conducted to reach a good combination of the parameters involved in the designed algorithms. For each test instance and each parameter combination, 5 independent runs of the algorithm have been done and by taking into account the average performance of the results, the best combination of the parameters has been driven. For both algorithms, Table 1 provides the range of the parameters and the best combination used to run the final experiments.

Regarding the MA, three and two different rules were proposed for the *selection* (Section 3.1.4) and *crossover* (Section 3.1.5), respectively. Here, we perform a simple test to select the best way of combining them in order to be used within the general framework of the MA. Table 2 reports the result of different combinations obtained by running the algorithm over the set of test instances. Preliminary tests show that the algorithm results are not so sensitive to different parameters' configuration. So, to run the instances we use the following set of parameters: $N_p = 100$, $N_c = 20$, $\beta = 0.05$, $\theta = 4$, $N = 10$ and $\gamma = 120$. Based on these results, on average the combination of the "Roulette wheel" in selecting the parents and "rule 1" for doing the *crossover*, leads to the best performance. Moreover, there is only 0.41% of gap between the best and worst performance of the available six different combinations over the set of 27 test instances.

4.3. Computational results

We have used ILOG Cplex 12.1 [23] to run the node-based and flow-based models to optimality. Particularly, 3600 s of CPU time is put on the maximum running time of Cplex and the results are reported in Tables 3–5 for the small, medium and large instances, respectively.

Table 2
Sensetivity analysis of different rules for *Crossover* and *Selection* procedures.

Crossover	*Crossover rule 1*			*Crossover rule 2*		
Selection	*Roulettewheel*	*Randomselection*	*Bestselection*	*Roulettewheel*	*Randomselection*	*Bestselection*
Average cost	**4636.69**	4644.06	4655.47	4640.58	4645.09	4639.30

Table 3 represents the results obtained by the proposed exact and metaheuristic methods for the small size instances. In this table, the first column gives the name of the instance, while the next four columns show the number of vertices (|*V*|), facilities (|*T*|), customers (|*W*|) and the minimum amount of prize to be collected (|*P*|), respectively. Columns six to eight, represent the results obtained by running the flow-based model over each of the instances and the objective function, gap from the optimality and running times are provided in the columns labeled by "Obj.", "Gap(%)" and "Time", respectively. The next three columns reports the same results obtained by running the node-based model. Finally, in the remaining columns each successive three columns represent, respectively, the results obtained by MA and VNS algorithms. In this table, the column represented by "Average Obj." gives the average objective function of the corresponding mataheuristic algorithm over five different runs, followed by its deviation ("Dev(%)") from the best solutions obtained by exact methods. Finally, the column labeled by "Time" is the average running time of the corresponding mataheuristic algorithm over five different runs. The last two rows of the table provide the number of the best solutions obtained by each method (No. Best) and their average performance over all the available instances (Average), respectively.

In almost all of the 32 small size instances, both exact formulations are able to reach the optimal solutions. On average, the running times are 519.01 and 335.90 s for the flow-based and node-based formulations, respectively. The MA and VNS algorithms perform very well on these instances in which all the optimal solutions have been obtained. In this table the best solutions found by each method is represented in bold and for all the available instances, MA and VNS are able to reach the optimal solutions in all five different runs. Finally, the average run time of the MA and VNS are 0.15 and 0.08 s over 32 available instances, respectively.

The results of different algorithms over the medium size instances are reported in Table 4. According to this table, by running the exact models, there are several instances for which the time threshold of Cplex has reached without proving the optimality of the best found solutions. Totally, there are 69 instances in the medium size group where in 9 and 14 instances the optimal solutions have been achieved by applying the node-based and flow-based models, respectively. Computational results over this class of instances show that almost all of the optimal solutions have been obtained by the metaheuristic algorithms as well. Essentially, there are only 1 and 3 instances for which the MA and VNS algorithms are not able to obtain the best solutions achieved by the exact models, respectively. There are also 32 instances for which the MA strictly

Table 3
Comparison of the results obtained by exact and metaheuristic algorithms for the small size instances.

Name	|*V*|	|*T*|	|*W*|	*P*	*Flow-based formulation*			*Node-based formulation*			*MA*			*VNS*		
					Obj.	Gap (%)	*Time*	Obj.	Gap (%)	*Time*	Average Obj.	Dev (%)	Time	Average Obj.	Dev (%)	Time
S1	51	15	35	17	**71.33**	0.00	3.00	**71.33**	0.00	0.75	**71.33**	0.00	0.13	**71.33**	0.00	0.05
S2	51	15	35	26	**105.98**	0.00	3.25	**105.98**	0.00	1.08	**105.98**	0.00	0.12	**105.98**	0.00	0.06
S3	51	15	35	35	**172.36**	0.00	3.25	**172.36**	0.00	0.08	**172.36**	0.00	0.09	**172.36**	0.00	0.07
S4	51	20	30	15	**75.29**	0.00	11.78	**75.29**	0.00	3.55	**75.29**	0.00	0.14	**75.29**	0.00	0.06
S5	51	20	30	22	**102.59**	0.00	37.83	**102.59**	0.00	9.47	**102.59**	0.00	0.17	**102.59**	0.00	0.07
S6	51	25	25	12	**38.49**	0.00	1.25	**38.49**	0.00	0.24	**38.49**	0.00	0.12	**38.49**	0.00	0.05
S7	51	25	25	18	**82.38**	0.00	131.23	**82.38**	0.00	25.28	**82.38**	0.00	0.17	**82.38**	0.00	0.07
S8	51	25	25	25	**140.62**	0.00	108.05	**140.62**	0.00	4.05	**140.62**	0.00	0.16	**140.62**	0.00	0.10
S9	52	16	35	17	**1378.45**	0.00	1.83	**1378.45**	0.00	0.67	**1378.45**	0.00	0.12	**1378.45**	0.00	0.05
S10	52	16	35	26	**2198.91**	0.00	4.05	**2198.91**	0.00	0.59	**2198.91**	0.00	0.11	**2198.91**	0.00	0.05
S11	52	21	30	15	**669.76**	0.00	0.56	**669.76**	0.00	0.19	**669.76**	0.00	0.13	**669.76**	0.00	0.06
S12	52	21	30	22	**1554.50**	0.00	60.91	**1554.50**	0.00	4.08	**1554.50**	0.00	0.15	**1554.50**	0.00	0.06
S13	52	21	30	30	**3910.04**	0.00	173.47	**3910.04**	0.00	13.94	**3910.04**	0.00	0.14	**3910.04**	0.00	0.10
S14	52	26	25	12	**572.13**	0.00	0.47	**572.13**	0.00	0.97	**572.13**	0.00	0.15	**572.13**	0.00	0.06
S15	52	26	25	18	**1240.67**	0.00	249.61	**1240.67**	0.00	53.17	**1240.67**	0.00	0.14	**1240.67**	0.00	0.07
S16	52	26	25	25	**2958.78**	0.00	1179.13	**2958.78**	0.00	746.22	**2958.78**	0.00	0.15	**2958.78**	0.00	0.12
S17	52	16	35	17	**1340.29**	0.00	2.19	**1340.29**	0.00	0.81	**1340.29**	0.00	0.15	**1340.29**	0.00	0.06
S18	52	16	35	26	**2291.58**	0.00	10.77	**2291.58**	0.00	2.11	**2291.58**	0.00	0.13	**2291.58**	0.00	0.06
S19	52	21	30	15	**878.30**	0.00	3.36	**878.30**	0.00	0.38	**878.30**	0.00	0.15	**878.30**	0.00	0.05
S20	52	21	30	22	**1521.64**	0.00	45.94	**1521.64**	0.00	6.88	**1521.64**	0.00	0.14	**1521.64**	0.00	0.06
S21	52	21	30	30	**3590.08**	0.00	38.06	**3590.08**	0.00	5.17	**3590.08**	0.00	0.13	**3590.08**	0.00	0.10
S22	52	26	25	12	**479.76**	0.00	0.42	**479.76**	0.00	0.81	**479.76**	0.00	0.13	**479.76**	0.00	0.05
S23	52	26	25	18	**1334.81**	0.00	432.09	**1334.81**	0.00	44.74	**1334.81**	0.00	0.15	**1334.81**	0.00	0.07
S24	52	26	25	25	**2894.96**	0.00	3484.84	**2894.96**	0.00	578.73	**2894.96**	0.00	0.19	**2894.96**	0.00	0.09
S25	76	23	52	26	**98.94**	0.00	130.92	**98.94**	0.00	30.03	**98.94**	0.00	0.16	**98.94**	0.00	0.07
S26	76	23	52	39	**147.89**	0.00	358.36	**147.89**	0.00	45.19	**147.89**	0.00	0.15	**147.89**	0.00	0.09
S27	76	30	45	22	**86.16**	0.00	78.69	**86.16**	0.00	50.33	**86.16**	0.00	0.17	**86.16**	0.00	0.08
S28	76	30	45	33	**119.54**	0.00	141.55	**119.54**	0.00	38.92	**119.54**	0.00	0.19	**119.54**	0.00	0.09
S29	76	38	37	18	**57.79**	0.00	57.94	**57.79**	0.00	34.25	**57.79**	0.00	0.14	**57.79**	0.00	0.08
S30	76	38	37	27	**106.39**	15.31	3600.00	**106.39**	16.07	3600.00	**106.39**	0.00	0.24	**106.39**	0.00	0.10
S31	76	38	37	37	**173.59**	16.03	3600.00	**173.59**	0.00	1846.03	**173.59**	0.00	0.27	**173.59**	0.00	0.21
S32	76	38	37	18	**20279.16**	0.00	2653.58	21638.57	33.99	3600.00	**20279.16**	0.00	0.15	**20279.16**	0.00	0.09
No. Best					**32**			31			**32**			**32**		
Average					**1583.54**	0.98	519.01	1626.02	1.56	335.90	**1583.54**	0.00	0.15	**1583.54**	0.00	0.08

Table 4
Comparison of the results obtained by exact and metaheuristic algorithms for the medium size instances.

Name	|V|	|T|	|W|	P	*Flow-based formulation*			*Node-based formulation*			*MA*			*VNS*		
					Obj.	Gap (%)	*Time*	Obj.	Gap (%)	*Time*	Average Obj.	Dev (%)	Time	Average Obj.	Dev (%)	Time
Ml	100	30	69	34	**6043.13**	15.58	3600.00	6061.44	12.34	3600.00	**6043.13**	0.00	0.23	**6043.13**	0.00	0.11
M2	100	30	69	51	**8471.25**	0.00	2763.55	**8471.25**	0.00	1358.34	**8471.25**	0.00	0.27	**8471.25**	0.00	0.12
M3	100	40	59	29	**3836.03**	0.00	3111.47	**3836.03**	0.00	2589.89	**3836.03**	0.00	0.28	**3836.03**	0.00	0.10
M4	100	40	59	44	6424.29	33.82	3600.00	6424.29	30.47	3600.00	**6288.73**	−2.11	0.33	**6288.73**	−2.11	0.13
M5	100	40	59	59	**8789.24**	13.04	3600.00	**8789.24**	0.00	1310.11	**8789.24**	0.00	0.36	**8789.24**	0.00	0.28
M6	100	50	49	24	**3148.77**	0.00	3285.84	**3148.77**	10.95	3600.00	**3148.77**	0.00	0.21	**3148.77**	0.00	0.12
M7	100	50	49	36	5783.34	45.20	3600.08	6056.26	46.18	3600.00	**5777.62**	−0.10	0.38	**5777.62**	−0.10	0.16
M8	100	50	49	49	9114.16	35.46	3600.00	**9094.20**	26.20	3600.00	**9094.20**	0.00	0.40	**9094.20**	0.00	0.43
M9	100	30	69	34	**4362.92**	0.00	2479.83	**4362.92**	0.00	251.84	**4362.92**	0.00	0.18	**4362.92**	0.00	0.10
M10	100	30	69	51	**8105.04**	13.70	3600.00	**8105.04**	14.85	3600.00	**8105.04**	0.00	0.25	**8105.04**	0.00	0.14
M11	100	40	59	29	**3592.77**	10.99	3600.00	**3592.77**	0.00	736.28	**3592.77**	0.00	0.23	**3592.77**	0.00	0.11
M12	100	40	59	44	7007.12	38.64	3600.00	7326.59	33.70	3600.00	**6948.96**	−0.83	0.33	**6948.96**	−0.83	0.15
M13	100	50	49	24	**2929.98**	0.00	2763.84	**2929.98**	0.00	2554.08	**2929.98**	0.00	0.30	**2929.98**	0.00	0.10
M14	100	50	49	36	**5446.84**	38.22	3600.00	**5446.84**	33.41	3600.00	**5446.84**	0.00	0.32	**5446.84**	0.00	0.14
M15	100	30	69	34	5274.42	23.29	3600.00	**5164.91**	0.00	1298.50	**5164.91**	0.00	0.29	**5164.91**	0.00	0.11
M16	100	30	69	51	**7298.66**	12.33	3600.00	**7298.66**	3.59	3600.00	**7298.66**	0.00	0.30	**7298.66**	0.00	0.14
M17	100	40	59	29	4662.59	32.54	3600.00	**4580.82**	19.13	3600.00	**4580.82**	0.00	0.25	**4580.82**	0.00	0.13
M18	100	40	59	44	**6880.86**	35.94	3600.00	6917.87	34.47	3600.00	**6880.86**	0.00	0.35	**6880.86**	0.00	0.15
M19	100	50	49	24	**3749.47**	34.96	3600.00	3753.16	21.93	3600.00	**3749.47**	0.00	0.24	**3749.47**	0.00	0.11
M20	100	50	49	36	6549.37	50.35	3600.00	6170.11	43.45	3600.00	**5936.53**	−3.79	0.40	**5936.53**	−3.79	0.17
M21	100	50	49	49	**8506.35**	17.83	3600.68	8643.74	16.03	3600.00	**8506.35**	0.00	0.46	**8506.35**	0.00	0.44
M22	100	30	69	34	5842.87	8.83	3600.00	5842.87	21.07	3600.00	**5704.71**	−2.36	0.27	**5704.71**	−2.36	0.10
M23	100	30	69	51	**7513.86**	0.00	334.38	**7513.86**	0.00	53.09	**7513.86**	0.00	0.26	**7513.86**	0.00	0.13
M24	100	40	59	29	**5029.98**	33.38	3600.00	**5029.98**	36.85	3600.00	5072.39	0.84	0.34	**5029.98**	0.00	0.11
M25	100	40	59	44	**6456.74**	20.15	3600.00	**6456.74**	4.61	3600.00	**6456.74**	0.00	0.34	**6456.74**	0.00	0.13
M26	100	50	49	24	4420.68	42.92	3600.00	4490.01	56.73	3600.00	**4280.76**	−3.17	0.25	**4280.76**	−3.17	0.11
M27	100	50	49	36	**5833.35**	41.01	3600.00	5976.48	42.91	3600.00	**5833.35**	0.00	0.35	**5833.35**	0.00	0.14
M28	100	50	49	49	**8143.74**	25.06	3600.00	**8143.74**	13.07	3600.00	**8143.74**	0.00	0.46	**8143.74**	0.00	0.42
M29	100	30	69	34	**4750.53**	0.00	1815.30	**4750.53**	0.00	82.46	**4750.53**	0.00	0.21	**4750.53**	0.00	0.09
M30	100	30	69	51	**7784.01**	14.81	3600.00	**7784.01**	0.00	835.74	**7784.01**	0.00	0.26	**7784.01**	0.00	0.12
M31	100	40	59	29	**3396.99**	6.23	3600.00	**3396.99**	0.00	171.93	**3396.99**	0.00	0.29	**3396.99**	0.00	0.12
M32	100	40	59	44	**6584.91**	33.59	3600.00	**6584.91**	24.36	3600.00	**6584.91**	0.00	0.37	**6584.91**	0.00	0.17
M33	100	40	59	59	**9949.88**	15.22	3600.00	**9949.88**	0.00	899.30	**9949.88**	0.00	0.32	**9949.88**	0.00	0.34
M34	100	50	49	24	**3066.63**	20.66	3600.00	**3066.63**	14.26	3600.00	**3066.63**	0.00	0.29	**3066.63**	0.00	0.10
M35	100	50	49	36	5750.13	44.72	3600.00	5750.13	46.67	3600.00	**5657.02**	−1.62	0.41	**5657.02**	−1.62	0.13
M36	100	50	49	49	**8009.42**	11.93	3600.00	**8009.42**	11.73	3600.00	**8009.42**	0.00	0.47	**8009.42**	0.00	0.51
M37	100	30	69	34	**1726.09**	0.00	1251.83	**1726.09**	0.00	493.80	**1726.09**	0.00	0.22	**1726.09**	0.00	0.09
M38	100	30	69	51	**2501.29**	0.00	104.81	**2501.29**	0.00	44.00	**2501.29**	0.00	0.22	**2501.29**	0.00	0.13
M39	100	40	59	29	**1518.85**	31.55	3600.00	1540.25	35.57	3600.00	**1518.85**	0.00	0.28	**1518.85**	0.00	0.11
M40	100	40	59	44	**2107.69**	20.41	3600.00	**2107.69**	26.42	3600.00	**2107.69**	0.00	0.34	**2107.69**	0.00	0.15
M41	100	50	49	24	1570.85	41.36	3600.00	1632.52	54.75	3600.00	**1499.04**	−4.57	0.23	**1499.04**	−4.57	0.11
M42	100	50	49	36	2238.72	41.59	3600.00	2558.96	50.54	3600.00	**2208.52**	−1.35	0.31	**2208.52**	−1.35	0.16
M43	100	50	49	49	**3228.85**	7.83	3600.00	**3228.85**	3.61	3600.00	**3228.85**	0.00	0.30	**3228.85**	0.00	0.61
M44	150	45	104	52	6800.55	36.43	3600.00	6611.13	30.72	3600.00	**6557.02**	−0.82	0.33	**6557.02**	−0.82	0.15
M45	150	45	104	78	9558.62	29.37	3600.00	9494.90	23.62	3600.00	**9489.22**	−0.06	0.43	9503.10	0.09	0.24
M46	150	60	89	44	5941.15	54.81	3600.00	5983.58	55.35	3600.00	**5393.39**	−9.22	0.36	**5393.39**	−9.22	0.17
M47	150	60	89	66	8557.71	52.88	3600.00	8171.50	48.29	3600.00	**8149.13**	−0.27	0.64	**8149.13**	−0.27	0.26
M48	150	75	74	37	4950.34	57.27	3600.00	**4583.03**	56.53	3600.00	**4583.03**	0.00	0.35	**4583.03**	0.00	0.19
M49	150	75	74	55	7388.02	59.08	3600.00	8329.72	66.61	3600.00	**6841.86**	−7.39	0.57	**6841.86**	−7.39	0.23
M50	150	75	74	74	11127.67	50.08	3600.00	10067.26	35.43	3600.00	**10061.50**	−0.06	0.70	10170.60	1.03	1.41

Table 4
(*Continued*)

Name	|V|	|T|	|W|	P	*Flow-based formulation*			*Node-based formulation*			*MA*			*VNS*		
					Obj.	Gap (%)	*Time*	Obj.	Gap (%)	*Time*	Average Obj.	Dev (%)	Time	Average Obj.	Dev (%)	Time
M51	150	45	104	52	6152.91	35.80	3600.00	6377.95	30.71	3600.00	**6133.27**	−0.32	0.37	**6133.27**	−0.32	0.19
M52	150	45	104	78	**8629.85**	25.70	3600.00	8730.54	16.41	3600.00	**8629.85**	0.00	0.45	**8629.85**	0.00	0.22
M53	150	60	89	44	5200.64	56.50	3600.00	5305.46	58.71	3600.00	**5199.33**	−0.03	0.39	**5199.33**	−0.03	0.16
M54	150	60	89	66	8712.97	55.90	3600.00	7624.55	48.94	3600.00	7471.30	−2.01	0.64	**7471.14**	−2.01	0.24
M55	150	75	74	37	5218.44	62.19	3600.00	7246.78	83.33	3600.00	**4898.98**	−6.12	0.38	**4898.98**	−6.12	0.18
M56	150	75	74	55	7573.78	60.09	3600.00	7959.61	73.71	3600.00	**7169.79**	−5.33	0.57	**7169.79**	−5.33	0.27
M57	200	60	139	69	6286.19	47.90	3600.00	6356.29	44.03	3600.00	6281.59	−0.08	0.61	**6281.11**	−0.08	0.27
M58	200	60	139	104	**9476.31**	30.88	3600.00	**9476.31**	20.52	3600.00	**9476.31**	0.00	0.64	9499.65	0.25	0.37
M59	200	80	119	59	6162.23	60.83	3600.00	6199.21	71.84	3600.00	**5620.09**	−8.80	0.66	**5620.09**	−8.80	0.26
M60	200	80	119	89	9241.57	63.05	3600.00	8827.56	57.22	3600.00	**8464.17**	−4.12	0.91	**8464.17**	−4.12	0.41
M61	200	100	99	49	5439.22	64.73	3600.00	5549.83	73.90	3600.00	**4895.78**	−9.99	0.57	**4895.78**	−9.99	0.28
M62	200	100	99	74	8388.30	66.49	3600.00	8346.32	71.59	3600.00	**7366.94**	−11.73	1.01	7383.64	−11.53	0.44
M63	200	60	139	69	6562.64	45.87	3600.00	6814.37	36.87	3600.00	**6257.61**	−4.65	0.49	6471.74	−1.39	0.25
M64	200	60	139	104	9626.72	34.50	3600.00	9642.84	19.73	3600.00	9588.34	−0.40	0.80	**9579.08**	−0.49	0.39
M65	200	80	119	59	6307.84	59.72	3600.00	6005.98	56.01	3600.00	**5442.88**	−9.38	0.63	**5442.88**	−9.38	0.24
M66	200	80	119	89	8854.88	54.51	3600.00	9175.32	51.43	3600.00	8368.61	−5.49	0.84	**8359.44**	−5.60	0.40
M67	200	100	99	49	5226.24	61.70	3600.00	5342.40	67.10	3600.00	4887.60	−6.48	0.57	**4872.50**	−6.77	0.23
M68	200	100	99	74	9338.61	64.42	3600.00	8593.37	63.76	3600.00	**7217.00**	−16.02	0.79	**7217.00**	−16.02	0.35
M69	200	100	99	99	11966.08	44.77	3600.00	11088.01	38.97	3600.00	**10754.88**	−3.00	1.03	10841.55	−2.22	4.32
No. Best					33			30			**63**			**63**		
Average					6262.62	30.31	3053.34	6262.20	32.59	3390.05	**6045.62**	**−1.90**	0.42	6051.23	−1.83	0.28

Table 5
Comparison of the results obtained by metaheuristic algorithms for the large size instances.

Name	30s				60s				120s			
	Best Obj.		Average Obj.		Best Obj.		Average Obj.		Best Obj.		Average Obj.	
	MA	*VNS*	*MA*	*VNS*	*MA*	*VNS*	*MA*	*VNS*	*MA*	*VNS*	*MA*	*VNS*
L1	**260.92**	261.32	265.66	**262.58**	**260.92**	261.32	265.66	**262.58**	**260.92**	261.32	**265.03**	262.58
L2	**568.82**	569.46	**571.59**	592.01	**568.82**	569.32	**571.59**	591.25	**568.82**	569.32	**570.56**	585.43
L3	221.78	**219.01**	224.43	**222.36**	221.78	**219.01**	224.43	**221.86**	221.78	**219.01**	224.43	**221.86**
L4	537.71	**521.77**	540.9	**534.94**	521.77	**521.62**	536.31	**534.36**	521.77	**521.62**	536.01	**531.29**
L5	**174.33**	175.9	**177.27**	178.07	**174.33**	175.9	**177.27**	178.07	**174.33**	175.9	**177.27**	178.07
L6	**409.69**	421.19	**413.59**	432.73	**409.69**	416.06	**413.59**	426.54	**409.69**	416.06	**413.59**	425.91
L7	**9659.42**	9860.09	**9783.33**	10262.22	**9659.42**	9860.09	**9775.89**	10236.86	**9659.42**	9860.09	**9746.22**	10222.04
L8	7445.34	**7428.56**	7475.27	**7444.14**	7445.34	**7428.56**	7475.27	**7434.01**	7445.34	**7410.18**	7474.78	**7430.33**
L9	**3406314.93**	3466420.1	**3451098.9**	3690228.26	**3363318.96**	3462163	**3433883.37**	3688099.4	**3359436.27**	3420977.4	**3428711.14**	3669406.3
L10	**5809962.9**	5956696.1	**5882573.7**	6204090.53	**5801249.02**	5944205.4	**5861252.04**	6195724	**5797974.84**	5940558.7	**5851822.69**	6169100.5
L11	2715609.38	**2698974.3**	2848941.4	**2830881.13**	2715609.38	**2698974.3**	2841862.11	**2825326.9**	2715609.38	**2698974.3**	2840534.15	**2776202.5**
L12	**4852148.78**	4969874.1	**4966469.1**	5354164.37	**4788332.03**	4954127.8	**4925313.33**	5348881.6	**4766605.34**	4954127.8	**4910558.61**	5330731.2
L13	2466741.98	**2402721.8**	**2540296.8**	2623601.18	2466741.98	**2402721.8**	**2531566.86**	2592871	2459972.67	**2399977.7**	**2525418.28**	2592322.2
L14	**4391043.77**	4485000.6	**4425225.1**	4729981.71	**4361460.06**	4481123.7	**4404383.57**	4652509	**4338690.35**	4479302.3	**4391780.16**	4643592.4
No. Best	**9**	5	**9**	5	**9**	5	**9**	5	**9**	5	**10**	4
Average	**1577407.25**	1599943.29	**1608937.74**	1696858.75	**1567732.17**	1597518.19	**1601180.69**	1688220.16	**1563837.33**	1594223.77	**1597882.86**	1680081.11

Table 6
Comparison of the MA with different algorithms available for the CSP.

Name	NC	*LS*2			*SN*			*MA*		
		Best Obj.	Average Obj.	*Time*	Best Obj.	Average Obj.	Time	Best Obj.	Average Obj.	Time
eil51	7	**164**	**164**	1	**164**	**164**	3	**164**	**164**	0
	9	**159**	**159**	1	**159**	**159**	2	**159**	**159**	0
	11	**147**	**147**	1	**147**	**147**	2	**147**	**147**	0
berlin52	7	**3887**	**3887**	1	**3887**	**3887**	2	**3887**	**3887**	0
	9	**3430**	**3430**	1	**3430**	**3430**	2	**3430**	**3430**	1
	11	**3262**	**3262**	1	**3262**	**3262**	2	**3262**	**3262**	1
st70	7	**288**	**288**	1	**288**	**288**	4	**288**	**288**	1
	9	**259**	**259**	2	**259**	**259**	4	**259**	**259**	1
	11	**247**	**247**	2	**247**	**247**	4	**247**	**247**	1
eil76	7	**207**	**207**	1	**207**	**207**	4	**207**	**207**	1
	9	186	186	1	**185**	**185**	4	**185**	**185**	0
	11	**170**	**170**	1	**170**	**170**	4	**170**	**170**	1
pr76	7	**50275**	**50275**	2	**50275**	**50275**	4	**50275**	**50275**	1
	9	**45348**	45462.2	2	**45348**	**45348**	4	**45348**	**45348**	1
	11	**43028**	**43028**	2	**43028**	**43028**	4	**43028**	**43028**	1
rat99	7	**486**	**486**	2	**486**	**486**	7	**486**	**486**	1
	9	**455**	**455**	2	**455**	**455**	7	**455**	**455**	1
	11	**444**	**444**	2	**444**	**444**	7	**444**	**444**	1
kroA100	7	**9674**	**9674**	3	**9674**	**9674**	7	**9674**	**9674**	1
	9	**9159**	**9159**	2	**9159**	**9159**	7	**9159**	**9159**	1
	11	**8901**	**8901**	2	**8901**	**8901**	7	**8901**	**8901**	1
kroB100	7	**9537**	**9537**	3	**9537**	**9537**	7	**9537**	**9537**	1
	9	**9240**	**9240**	3	**9240**	**9240**	7	**9240**	**9240**	1
	11	**8842**	**8842**	3	**8842**	**8842**	7	**8842**	**8842**	1
kroC100	7	**9723**	**9723**	3	**9723**	**9723**	7	**9723**	**9723**	1
	9	**9171**	**9171**	2	**9171**	**9171**	7	**9171**	**9171**	1
	11	**8632**	**8632**	2	**8632**	**8632**	7	**8632**	**8632**	1
kroD100	7	**9626**	**9626**	2	**9626**	**9626**	6	**9626**	**9626**	1
	9	**8885**	**8885**	3	**8885**	**8885**	7	**8885**	**8885**	1
	11	**8725**	**8725**	3	**8725**	**8725**	7	**8725**	**8725**	1
kroE100	7	**10150**	**10150**	2	**10150**	**10150**	6	**10150**	**10150**	1
	9	**8991**	**8991**	2	**8991**	**8991**	7	**8991**	**8991**	1
	11	**8450**	**8450**	2	**8450**	**8450**	7	**8450**	**8450**	1
rd100	7	**3461**	3485.6	2	**3461**	3493.8	6	**3461**	**3461**	1
	9	**3194**	**3194**	2	**3194**	**3194**	6	**3194**	**3194**	1
	11	**2922**	**2922**	2	**2922**	**2922**	6	**2922**	**2922**	1
kroA150	7	**11423**	11800	4	**11423**	**11423**	11	**11423**	**11423**	2
	9	**10056**	10062.4	3	**10056**	**10057.6**	10	**10056**	10062.4	3
	11	**9439**	**9439**	3	**9439**	**9439**	9	**9439**	**9439**	2
kroB150	7	**11457**	11491.2	4	**11457**	**11457**	10	**11457**	**11457**	3
	9	**10121**	**10121**	4	**10121**	**10121**	10	**10121**	**10121**	2
	11	**9611**	**9611**	4	**9611**	**9611**	10	**9611**	**9611**	2
kroA200	7	**13285**	13666.4	6	**13285**	13327	15	13289	**13292.8**	4
	9	**11708**	11716.8	5	**11708**	11731.6	14	**11708**	**11712.8**	4
	11	**10748**	10848.6	5	**10748**	10865.6	13	**10748**	**10809.6**	6
kroB200	7	13100	13511.6	5	**13051**	13181.2	15	**13051**	**13051**	3
	9	11900	11964.8	5	**11864**	**11878.4**	14	**11864**	**11878.4**	4
	11	10676	10809.6	5	**10644**	10656.8	13	**10644**	**10644**	4
No. Best		44	36		**48**	42		47	**47**	
Average		9026.02	9060.55	2.55	**9023.57**	9031.38	6.98	9023.65	**9025.54**	**1.59**

outperforms the best solutions obtained by the two exact models while this value is 30 for the case of VNS algorithm. The average running time of the MA algorithm over all of the instances is 0.42 s while this time is only 0.28 s for the VNS algorithm. Moreover, this time is 6262.62 and 6262.20 s for the flow-based and node-based models, respectively. The overall average deviation of the best solution obtained by the exact models with respect to results by MA and VNS algorithms clearly indicates the superiority of the metaheuristic methods where the overall deviations are −1.90% and −1.83% for the MA and VNS algorithms, respectively.

The results of the MA and VNS algorithms over the 14 large size instances are reported in Table 5. To have a fair comparison of the metaheuristic algorithms we have used CPU time as the termination criterion of the two algorithms. Essentially, both algorithms have been executed with three different time limits, namely 30, 60 and 120 s. In this table, the columns labeled by "Best Obj." and "Average Obj." give the best and average performance of the different algorithms over five different independent runs of corresponding algorithms. Moreover, the number of the best solutions obtained in each column and the correspondingaverage of different columns are given in the last two rows, respectively. Taking into account the overall average of the "Best Obj." column, the results show that on average the MA algorithm always outperform the VNS algorithm in different time limits. This is the same when considering the average performance of the algorithms over five independent runs. Taking into account the number of the best solutions obtained by each algorithm in different time limits, the results show that still MA outperform the VNS by obtaining larger number of the best known solutions.

The results of the metaheuristic algorithms over the large size instances, clearly show that the MA outperforms the VNS algorithm. To validate the quality of the proposed MA, we adapt this method to be used for the CSP. The CSP was introduced by Current and Schilling [11] with potential applications in relief transportation. The authors developed a simple heuristic method, where, at first the set covering problem (SCP) is solved over a given CSP instances to find the minimum number of vertices that are able to cover all the available demands. Following this step, the TSP tour is constructed over the resulted vertices from the first step of the algorithm. Since there could be multiple solutions with the same SCP objective function, the authors proposed to find all SCP optimal solutions and take the one having the minimum TSP tour. Golden et al. [18] developed two heuristics (called *LS1* and *LS2*) which applies swap, extraction-reinsertion and perturbation moves to try to find good quality solutions. Finally, an involved algorithm combining exact and heuristic ideas (called *SN*) was proposed in the literature [37] that outperforms the other available methods. Essentially, the *SN* method follows a destruct-and-repair paradigm, where, an initial feasible solution is destroyed by extracting some of the vertices visited on the tour and repaired by allocating the available vertices to the solution through solving an integer linear programming model to optimality [37].

Our introduced problem can be converted to the CSP. To do so, suppose $|W| = |T|$ and for each $i \in W$ the coordinates of i to be the same as at least one $j \in T$. Moreover, the covering distance of j is supposed to be the same as customer i in the CSP case. Letting $dis(i, j)$ to represent the distance between i and $j \in T$ and assuming $(0, i_1, i_2, \ldots, i_k, 0)$ to be a feasible tour, the depot could be easily eliminated from the calculation by letting $dis(i_k, 0) + dis(0, i_1) = dis(i_k, i_1)$.

The same set of parameters have been used to run the CSP instances. To compare the MA's performance over the CSP instances, we have used 16 benchmark instances available in the literature [18]. Moreover, to do the comparison we use the results obtained by *LS2* and *SN* which are the best available methods for the CSP. Table 6 provides the results of different algorithms for the CSP, in which the label of the columns have the same meaning as those explained for Table 5. Moreover, in these instances each customer could cover its NC = 7, 9, 11 nearest customers, resulting in 48 instances. Each algorithm has been executed five times and the best, worst and average performance are represented in the appropriate columns. Considering the best performance of the algorithms over 48 available instances, 47 of the best known solutions are obtained by the MA. The average run time of the MA is 1.59 while this is 2.55 and 6.98 for the *LS2* and *SN* methods, respectively. Taking into account the overall average performance of the three methods, the MA has the best performance. Particularly, it outperforms the other two methods by obtaining the most number of the best found solutions and the minimum average cost over all the 48 instances.

5. Conclusions and future directions

We proposed a variation of the traveling salesman problem with potential applications in humanitarian relief transportation problem. Given a set of vertices including the depot, facility and customer vertices, the goal of the introduced problem is to construct a minimum length cycle over a subset of facilities while covering a given number of customers. Two mathematical models and two metaheuristic algorithms including memetic algorithm and variable neighborhood search were proposed. Comparing the results obtained by the proposed metahuristic algorithms with those obtained by the exact models, clearly indicates the effectiveness of the developed heuristic methods. Moreover, the memetic algorithm was adapted to be used for the covering salesman problem and the results are comparable with the state of the art algorithms available in the literature.

uncited reference

Q2

[2]

References

[1] N. Altay, W.G. Green, OR/MS research in disaster operations management, Eur. J. Oper. Res. 175 (2006) 475–493.
[2] F. Angel-Bello, A. Alvarez, I. García, Two improved formulations for the minimum latency problem, Appl. Math. Model. 37 (4) (2013) 2257–2266.
[3] S. Arya, D.M. Mount, Algorithms for fast vector quantization, in: Proceedings of the Data Compression Conference 1993, IEEE Press, 1993.
[4] B. Awerbuch, Y. Azar, A. Blum, S. Vempala, Improved approximation guarantees for minimum weight k-trees and prize-collecting salesman, SIAM J. Comput. 28 (1) (1995) 254–262.
[5] E. Balas, The prize collecting traveling salesman problem, Networks 19 (1989) 621–636.
[6] X. Bao, Z. Liu, An improved approximation algorithm for the clustered traveling salesman problem, Inf. Process. Lett. 112 (23) (2012) 908–910.
[7] B. Bontouxa, C. Artiguesb, D. Feilletc, A memetic algorithm with a large neighborhood crossover operator for the generalized traveling salesman problem, Comput. Oper. Res. 37 (11) (2010) 1844–1852.
[8] L. Buriol, P.M. França, P. Moscato, A new memetic algorithm for the asymmetric traveling salesman problem, J. Heuristics 10 (5) (2004) 483–506.
[9] J.F.F. Bérubé, M. Gendreau, J.Y. Potvin, A branch-and-cut algorithm for the undirected prize collecting traveling salesman problem, Networks 54 (1) (2009) 56–67.
[10] A. Blum, R. Ravi, S. Vempala, A constant-factor approximation algorithm for the k-mst problem., in: ACM Symposium on Theory of Computing, 1996, pp. 442–448.
[11] J.R. Current, D.A. Schilling, The covering salesman problem, Transport. Sci. 23 (3) (1989) 208–213.
[12] D. Cvetkovic, I.C. Parmee, Preferences and their application in evolutionary multiobjective optimization, IEEE Trans. Evol. Comput. 6 (2002) 42–57.
[13] G.B. Dantzig, R. Fulkerson, S.M. Johnson, Solution of a large scale traveling salesman problem, Oper. Res. 2 (4) (1954) 393–410.
[14] M. Desrochers, G. Laporte, Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints, Oper. Res. Lett. 10 (1991) 27–36.
[15] N. Garg, A 3 factor approximation algorithm for the minimum tree spanning k vertices., in: Proceedings IEEE Foundations of Computer Science, 1996, pp. 302–309.
[16] M. Gendreau, G. Laporte, F. Semet, The covering tour problem, Oper. Res. 45 (1997) 568–576.
[17] S. Ghafurian, N. Javadian, An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesman problems, Appl. Soft Comput. 11 (1) (2011) 1256–1262.
[18] B. Golden, Z. Naji-Azimi, S. Raghavan, M. Salari, P. Toth, The generalized covering salesman problem, INFORMS J. Comput. 24 (4) (2012) 534–553.
[19] G. Gutin, A.P. Punnen (Eds.), The Traveling Salesman Problem and its Variations., Kluwer, Boston, 2002.
[20] G. Gutin, D. Karapetyan, A memetic algorithm for the generalized traveling salesman problem, Nat. Comput. 9 (1) (2010) 47–60.
[21] M.E. Hartmann, Ö. Özlük, Facets of the p-cycle polytope, Discrete Appl. Math. 112 (1–3) (2001) 147–178.
[22] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.
[23] IBM ILOG Cplex, http://www.ilog.com
[24] N. Javahar, N. Balaji, A genetic algorithm based heuristic to the multi-period fixed charge distribution problem, Appl. Soft Comput. 12 (2) (2012) 682–699.
[25] I. Kara, G. Laporte, T. Bektas, A note on the lifted Miller-Tucker-Zemlin subtour elimination constraints for the capacitated vehicle routing problem, Eur. J. Oper. Res. 158 (2004) 793–795.
[26] I. Karaoglan, F. Altiparmak, I. Kara, B. Dengiz, The location-routing problem with simultaneous pickup and delivery: formulations and a heuristic approach, Omega 40 (2012) 465–477.
[27] D. Karapetyan, G. Gutin, Efficient local search algorithms for known and new neighborhoods for the generalized traveling salesman problem, Eur. J. Oper. Res. 219 (2) (2012) 234–251.

[28] K.Y. Kim, K. Park, J. Ko, A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling, Comput. Oper. Res. 30 (2003) 1151–1171.
[29] G. Laporte, A. Asef-Vaziri, C. Sriskandarajah, Some applications of the generalized travelling salesman problem, J. Oper. Res. Soc. 47 (12) (1996) 1461–1467.
[30] J.F. Maurras, V.H. Nguyen, On the linear description of the k-cycle polytope, Int. Trans. Oper. Res 8 (2001) 99–109.
[31] C.E. Miller, A.W. Tucker, R.A. Zemlin, Integer programming formulations and traveling salesman problems, J. Assoc. Comput. Mach. 7 (1960) 326–329.
[32] N. Mladenović, P. Hansen, Variable neighborhood search, Comput. Oper. Res. 24 (1997) 1097–1100.
[33] P. Moscato, Memetic algorithms: a short introduction, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization., McGraw-Hill, New York, 1999, pp. 219–234.
[34] S.S. Radiah Shariff, N.H. Moin, M. Omar, Location allocation modeling for healthcare facility planning in Malaysia, Comput. Ind. Eng. 62 (4) (2012) 1000–1010.
[35] D.G. Reina, S.L. Toral Marín, N. Bessis, F. Barrero, E. Asimakopoulou, An evolutionary computation approach for optimizing connectivity in disaster response scenarios, Appl. Soft Comput. 13 (2) (2013) 833–845.
[36] G. Reinelt, A traveling salesman problem library, ORSA J. Comput. 3 (4) (1991) 376–384.
[37] M. Salari, Z. Naji Azimi, An integer programming-based local search for the covering salesman problem, Comput. Oper. Res. 39 (11) (2012) 2594–2602.
[38] X. Wen, Y. Xu, H. Zhang, Online traveling salesman problem with deadline and advanced information, Comput. Ind. Eng. 63 (4) (2012) 1048–1053.
[39] H. Yaman, Formulations and valid inequalities for the heterogeneous vehicle routing problem, Math. Program. 106 (2006) 365–390.
[40] K. Yang, Y.K. Liu, G.Q. Yang, Solving fuzzy p-hub center problem by genetic algorithm incorporating local search, Appl. Soft Comput. (2012).
[41] R. Zanjirani Farahani, N. Asgari, N. Heidari, M. Hosseininia, M. Goh, Covering problems in facility location: a review, Comput. Ind. Eng. 62 (1) (2012) 368–407.
[42] F. Zhao, S. Li, J. Sun, D. Mei, Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem, Comput. Ind. Eng. 56 (4) (2009) 1642–1648.