International Journal of Society Systems Science

ISSN online: 1756-252X - ISSN print: 1756-2511
https://www.inderscience.com/ijsss

An improved hybrid genetic algorithm to solve the multi-vehicle covering tour problem with restriction on the number of vertices

Manel Kammoun

DOI: 10.1504/IJSSS.2023.10058186

Article History:

Received:	30 January 2021
Last revised:	29 December 2021
Accepted:	15 April 2022
Published online:	08 August 2023

An improved hybrid genetic algorithm to solve the multi-vehicle covering tour problem with restriction on the number of vertices

Manel Kammoun

Department of Quantitative Methods,
Faculty of Economics and Management,
MODILS,
Sfax University,
Sfax, Tunisia
Email: kamounemanel@gmail.com

Abstract: In this paper, we address the multi-vehicle covering tour problem where only the restriction on the number of vertices in each route (m-CTP-p). The objective of the m-CTP is to minimise the total routing cost and fulfill the demand of all customers such that each customer which is not included in any route must be covered. Each covered vertex must be within a given distance of at least a visited vertex and the number of vertices on a route does not exceed a pre-defined number p. We propose two approaches to solve this variant. First, we develop a genetic algorithm (GA) using an iterative improvement mechanism. Then, an effective hybrid genetic algorithm (HGA) is developed in addition to a local search heuristic based on variable neighborhood descent method to improve the solution. Extensive computational results based on benchmark instances on the m-CTP-p problem show the performance of our methods.

Keywords: covering; genetic algorithm; variable neighbourhood descent; VND; hybrid approach.

Reference to this paper should be made as follows: Kammoun, M. (2023) 'An improved hybrid genetic algorithm to solve the multi-vehicle covering tour problem with restriction on the number of vertices', *Int. J. Society Systems Science*, Vol. 14, No. 3, pp.247–267.

Biographical notes: Manel Kammoun was obtained her MS in Operational Research and Production Management and her PhD in Operational Research and Decision Aide from the Sfax university, Tunisia. Her main research interests concern combinatorial optimisation and heuristic search with applications to logistics and routing transportation systems. She is a member of Modelling and Optimisation for Decisional, Industrial and Logistic Systems Laboratory (MODILS).

1 Introduction

The VRP is one of the most famous combinatorial problems. It is a classical logistics problem that aims at servicing a given set of customers using a set of vehicles located at a central depot. Many applications defined on the real world need to provide vehicle routing strategies. For a detailed literature review about the VRP, we refer to the paper of Braekers et al. (2016).

The CTP has been motivated by the humanitarian logistics that aims at minimising damage and losses in human lives after a disaster where appear some restrictions on some resources (time, budget, ...). These restrictions prevent the healthcare organisations to provide relief as soon as possible to the victims and to supply the affected populations with food and medecines.

The covering issue is considered in a large number of real world applications including but not limited to disaster management, problems arising in emergency situations and busines sector. For example the covering issue is involved to solve the problem of disaster relief (De La Torre et al., 2012). Besides, Doerner and Hartl (2008) deal with the unavailability of roads to reach a specific customer by applying the m-CTP. The restriction of time can be appear in the problem of vaccination campaigns where the service duration to vaccinate the people is more important than the travelling time. In addition, we have the problem of resource availability like the number or the capacity of the vehicles such as dairy practice problem in the case of milk collection points (Simms, 1989).

In this paper, we address a generalisation of the covering tour problem. The studied problem can be described as follows: find a good design of a set of tours to supply demands to cities with the aim of minimising the total length of all tours while respecting a covering distance. This distance is defined as the largest distance that any customer must travel to reach the nearest supplied city or visited vertex. In this work we consider the constraint on the number of vertices on each tour.

The paper is organised as follows. In Section 2, we present a literature review of the CTP. In Section 3, we describe the studied problem. Then in Section 4, we present the different steps of our proposed approches. Finally, in Section 5 we present computational results and a comparative study was provided.

2 Literature review

In this section and before we proceed with our study, we briefly review the related problems of the CTP and their variants.

Over the last decades, several authors have adopted exact methods, heuristics and metaheuristics to solve different variants of the VRP. But, few works focus on the CTP despite his importance in the real world. The CTP was first introduced by Current (1981) where he divides the set of vertices into two groups. The first one includes the vertices that can be visited and contains a set of vertices that must be visited and the second group includes the vertices that must be covered by the tour. The CTP seeks to find a minimum-length Hamiltonian cycle along a set of vertices where another set of vertices lies within a given distance of at least a vertex of a route. For instance, Gendreau et al. (1997) solve the one vehicle version (1-CTP) exactly by branch and cut algorithm. The CTP was extended by Hachicha et al. (2000) to include multiple

routes. They introduce a generalised variant of the CTP called m-CTP which aims to find a minimum length set of vehicle routes with respecting the constraints related to the length of each route and the number of vertices that it contains and solve the problem heuristically. Lopes et al. (2013) develop a branch-and price algorithm to solve the same problem. Jozefowiez (2011) solve the multi-version by an exact algorithm based on a column generation approach where they formulate the sub problem similarly to the 1-CTP model proposed in Gendreau et al. (1997) and consider the master problem as a simple set covering problem. Jozefowiez et al. (2007) added a new objective to the m-CTP aiming at minimising the greatest distance between the vertices of another set and the nearest visited vertex. They solve the proposed bi-objective covering tour problem (BOCTP) with two-phases cooperative strategy that combines a multi-objectives evolutionary algorithm with a branch-and-cut algorithm initially designed to solve a single objective covering tour problem. Jozefowiez (2015) solve the m-CTP exacty using a branch-and-price algorithm where the resulting subproblem is a variant of the profitable tour problem. It is reduced to a ring star problem and solved by a branch-and-cut algorithm. Há et al. (2013) study a variant of the m-CTP with no tour length constraints and develop exact and metaheuristic methodologies to solve the problem. They develop a branch-and-cut algorithm and a two-phase metaheuristics derived from evolutionary local search (ELS). Kammoun et al. (2015) show the efficiency of a variable neighbourhood search (VNS) algorithm for solving the m-CTP with only the restriction on the number of vertices in each route (m-CTP-p) and improving results in terms of quality and solution time compared with Há et al. (2013). Vargas Suarez (2012) analyse the single-vehicle CTP, the m-CTP and the orienteering problem (OP) and developed a m-Selector operator which proves their effectiveness to solve these problems.

More recently, the multi-vehicle multi-covering tour problem was introduced and studied by Pham et al. (2017). An integer linear program, a branch and cut algorithm and a genetic algorithm (GA) are developed to solve the problem. Margolis et al. (2019) study the multi-vehicle covering tour problem with time windows (MCTPTW) that aims to determine a set of maximal coverage routes that serve a secondary set of sites under a fixed time schedule, coverage requirements and energy restrictions. They extend both the m-CTP and the VRP with time-windows (VRPTW) under energy constraints to formulate a deterministic mixed-integer second-order cone programming (MISOCP).

Furthermore, Current and Schilling (1994) solve the maximum covering tour problem (MCTP) as a multi-objective CTP. Another m-CTP variant, which considers multiple objectives is introduced by Tricoire et al. (2012).The authors deal with stochastic demand and consider multiple objectives including the cost and expected uncovered demand. They formulate the multi-objective problem using the ε – constraint method and solve the problem exactly by a branch-and-cut approach. Flores-Garza et al. (2017) introduced the cumulative m-CTP, which aims to find a set of tours that must be followed by a fleet of vehicles in order to minimise the sum of arrival times (latency) at each visited location. They formulate a mixed integer linear programming model and develop a greedy randomised adaptive search procedure (GRASP) to solve it. Another variant of the m-CTP named multi-vehicle multi-covering tour problem (mm-CTP) is introduced by Pham et al. (2017) in which the customer vertices must be covered several times rather than once. Kammoun et al. (2019) develop a general variable neighbourhood search with mixed VND (GVNS) to solve the same problem. The results of the GVNS show an appreciable competitive behaviour compared with

Pham et al. (2017). They develop also two hybrid metaheuristics that combine GA and variable neighbourhood descent (VND) method and a general variable neighbourhood search (GVNS) algorithm (Kammoun et al., 2020). The results show that these hybrid approaches are competitive with the ELS and GA proposed in the literature.

Several methods are used to solve different variants of the CTP problem. We note that the population-based metaheuristic search method (GA) hybridised with LS technique have been successfully applied to hundreds of real world problems in a wide range of domains and represent a trade-off between global searching and local searching in terms of quality of the solution and the time to converge to a global optimum (Ochelska-Mierzejewska et al., 2021; Viana et al., 2020; Barkaoui and Berger, 2020). The main reason behind the hybridisation of GA and VND algorithms is to exploit the complementary features of each optimisation techniques. In this context, we propose an hybridisation to the GA and we enhance it with a VND algorithm.

3 Problem description

The m-CTP-p is defined by Baldacci et al. (2005) as an undirected graph $G = (V \cup W, E)$, where $V \cup W$ is the vertex set and E is the edge set. They consider different kinds of locations: V, T and W. V is the set of vertices that can be visited and W is the set of vertices that must be covered by up to m vehicles. $T \subseteq V$ is the set of vertices that must be visited including the depot where identical vehicles are located. They consider that each vertex of V has an unit demand and each vehicle has a capacity of p. In this problem, we do not need to visit all vertices of V with the exception of the vertices of T to satisfy the demand of each customer. The demand of each customer could be satisfied in two different ways: either by visiting the customer along the tour or by covering it. Covered vertices must be within a predefined distance d from the tour. The m-CTP is an NP-hard problem as it reduces to a travelling salesman problem (TSP) when $d = 0$ and $V = W$, to a VRP with unit demand when $T = V$ and $W = \varnothing$; or simply to a CTP when there are no capacity constraints. The difference between these routing problems is illustrated by the Figure 1. In the TSP solution the route can be started with any vertex whereas in the VRP solution all the vehicles start from the depot and all the vertices must be visited.

The goal of the m-CTP is to find a minimum length set of vehicle routes with respecting the number of vertices in each route and satisfying the following constraints:

1 each vehicle route starts and ends at the depot

2 each vertex of T belongs to exactly one route while each vertex of $V \setminus T$ belongs to at most one route

3 each vertex of W must be covered by a route, i.e., a covered vertex $v_l \in W$ can be covered by a visited vertex $v_k \in V$ when the distance between these two vertices small than a certain fixed value

4 the number of vertices on a route (excluding the depot) is less than a given value p

5 the length of each route does not exceed a fixed value q.

In the m-CTP-p, the last constraint is relaxed and the only considered restriction is related to the number of vertices on each route. Figure 2 shows an example of the m-CTP instance and their corresponding feasible solution where $|T| = 1$, $|V| = 10$, $|W| = 8$, $m = 3$ and the number of vertices in each route does not exceed $p = 3$.

Figure 1 An illustrative example of the different routing problems

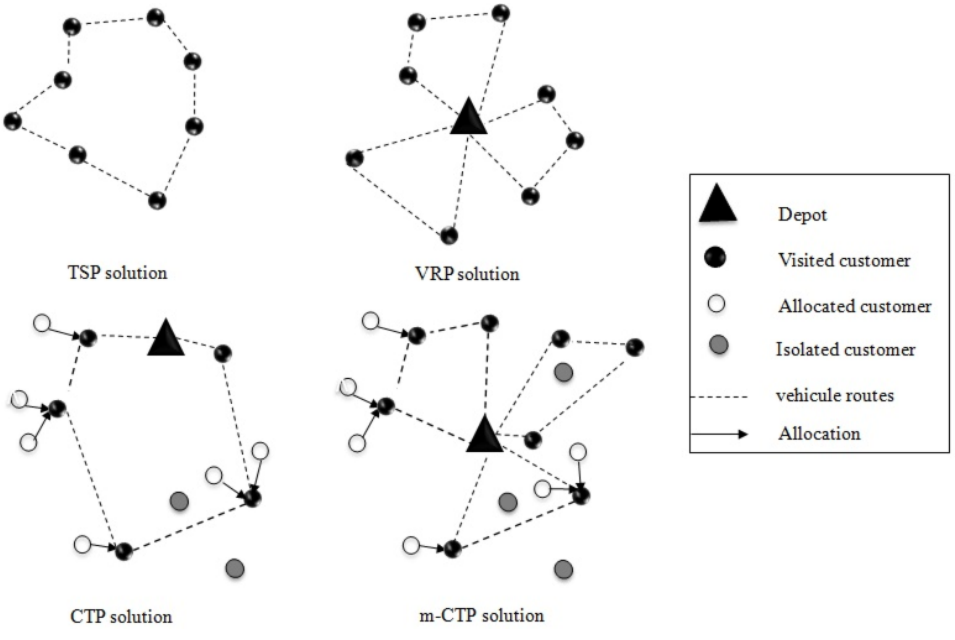
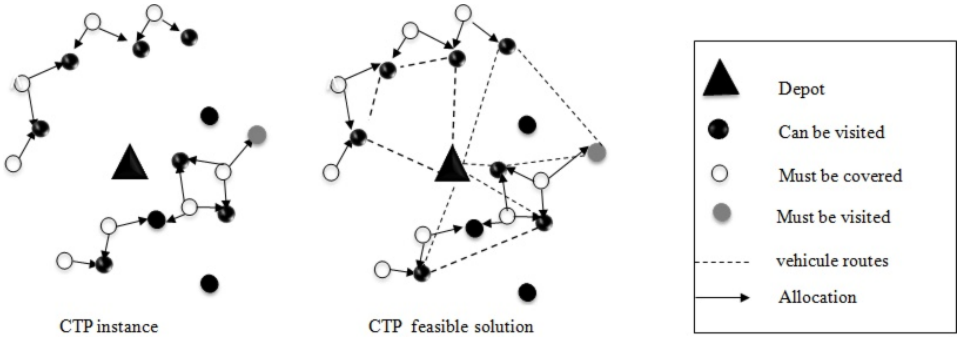


Figure 2 Example of feasible solution for a m-CTP instance



4 Proposed approches

4.1 A GA for m-CTP-p

The GA is considered as a robust algorithm with great flexibility as it follows a set of genetic operators which prevent the solution of being trapped in a local optimum. The

GA was shown to be competitive with other modern heuristic techniques since a wide variety of hard optimisation problems were solved successfully by GA (Bräysy et al., 2004; Oliveira da Costa et al., 2018).

In this work, we develop a GA to solve the m-CTP-p. To begin our algorithm, some parameters need to be initialised as population size, number of generations, mutation and crossover rates. As a second step, we define an initial population based on a set of random feasible solutions named chromosomes. Then we measure the fitness of each chromosome to select two parents. An offspring is generated from the selected parents using the crossover operator. In order to maintain a diversity in the population, we mutate the offspring, then we use an iterative improvement procedure (IIP) that apply a swap move between a gene from the chromosome and a selected element from the remaining set of vertices and find the best position of this element in the offspring. Finally, a new population is formed by applying the replacement phase in which we reject the worst parent from the population and replace it with the obtained offspring. After the predefined number of generation is performed we obtain the best offspring. The different steps of our approach are summarised in Algorithm 1.

Algorithm 1 GA for m-CTP-p

Initialise: $psize$ = 10, P_m = 0.1, P_c = 0.9, nbg = 100; /*Global parameters*/
Generate Pop; /*First generation of individuals*/
Calculate the evaluation function $FEVAL$ of each individual S_i in Pop;
Let S_{best} the individual with the best fitness;
Let S_{worst} the individual with the worst fitness;
while number of generations is not reached **do**
  $S_1 \leftarrow SELECT(Pop)$;
  $S_2 \leftarrow SELECT(Pop)$;
  $S_{new} \leftarrow CROSSOVER(S_1, S_2)$;
  $S'_{new} \leftarrow MUTATION(S_{new})$;
  $S''_{new} \leftarrow IIP(S'_{new})$;
  if $FEVAL(S''_{new}) < FEVAL(S_{worst})$ **then**
    $S_{best} \leftarrow REPLACEMENT(S''_{new}, S_{worst}, Pop)$;
  end if
end while

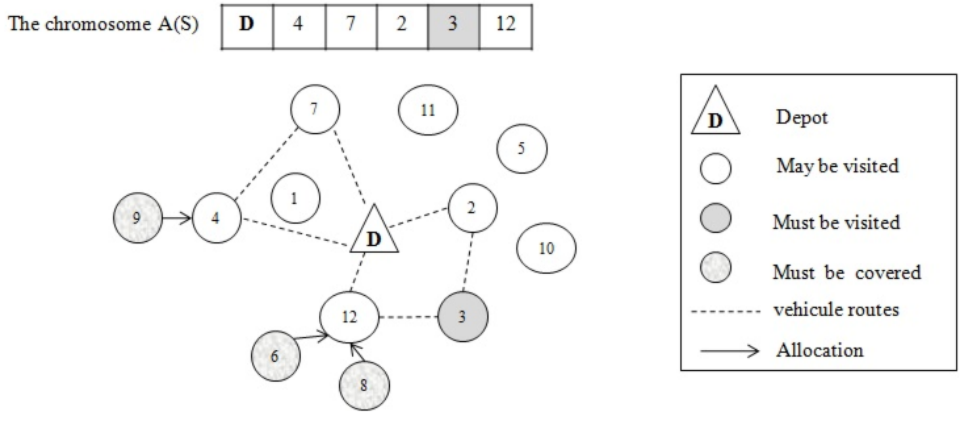
In the following we describe with more details each step of our approach.

4.1.1 Solution representation

In this work, the initial population was built by generating an initial set of feasible solutions. Here we describe a constructive heuristic designed to derive good initial feasible solution. The heuristic constructs a vehicle route that guarantee the covering constraint. Starting at the depot, in each iteration, the next visit (vertices from V) is chosen in order to maximise the number of covered vertices. Each solution contains a set of vertices that must be visited (T) and some vertices from V to cover the total vertices in W. The obtained feasible solutions represent the set of individuals of the population. We measure the fitness function of each individual from the population and initialise the best one. The representation of each individual in the population have a big influence on the choice of efficient and appropriate GA operators. In our approach, we use a vector $A(S)$ to present a solution S. This vector always started by the depot

(D) and the remaining elements of the vector give the set of customers which form the different tours in the solution. Notice that each chromosome must contain all the vertices of T since they represent the vertices that must be visited while the vertices of W should not be present in the solution. To illustrate the representation of a solution S, we consider an example with single depot, 12 customers and 2 vehicles (see Figure 3).

Figure 3 An example of m-CTP solution representation



In this example two routes are presented where each one started from the same depot. In the first route customer 4 is followed by customers 7 while in the second route three customers (2, 3, 12) are visited where one of them belongs to T. As we see in this example customers 6 and 8 are covered by the visited customer 12 while customer 9 was covered by customer 4. In the initial population, a set of individuals was randomly generated. An initial individual S is initially constructed by the insertion of all customers that must be visited (T) at random in a vector A, then by generating a random insertion of the remaining set of customer until all customers of W will be covered.

4.1.2 Selection operator

The second step in a standard GA is how to select parents from the initial population. In this step of the algorithm we do not create a new chromosome, we only pick good parents from the initial population that will be used in the next step to create new offsprings. Various traditional selection mechanisms are used in the literature: roulette wheel selection, rank selection, tournament selection and elitism selection. In our work we applied the roulette wheel selection to select two parents from the initial population to survive and create new offspring. We calculate the fitness and the probability distribution for each individual. Then, we calculate the cumulative probability for the population. The parents are selected according to their probability distribution. Our selection procedure is described in Algorithm 2.

Algorithm 2 $SELECT(Pop)$

for $i = 0; i < psize$ **do**
  Calculate $FEVAL(S_i)$;
end for
Let $M = \max(FEVAL(S_1), FEVAL(S_2), ..., FEVAL(S_{psize}))$;
for $i = 0; i < psize$ **do**
  $P_i = 1 + (M - FEVAL(S_i))$; /*the selection probability*/
end for
for $i = 0; i < psize$ **do**
  $P_c = P_c + P_i$; /*the cumulative probability*/
end for
Generate a random number r in the range $[0, P_c]$;
$t \leftarrow P_0$;
for $i = 0; i < psize$ **do**
  if $r < t$ **then**
    $S \leftarrow S_i$;
  else
    $t = t + p_{i+1}$;
  end if
end for

4.1.3 Crossover operator

After selecting two parents, the crossover takes place to create the new offspring. The crossover operator consists of recombining the parents to form the new chromosome. In this study we apply the one point crossover (cp) and we copy the part of the chromosome before the cp from the first parent into the new offspring. Then, we clean up the second parent by removing the genes formed the new offspring from them. Finally, we insert the gene from the second parent into the offspring and repeat this step until a feasible offspring is reached. The crossover procedure is given with more details in Algorithm 3 and then illustrated in Figure 4(a).

Algorithm 3 $CROSSOVER(S_1, S_2)$

Select randomly a cp from the first parent S_1; /*crossing point*/
for $i = 0; i < cp$ **do**
  $S' \leftarrow S_1$;
end for
clean up S_2 /*delete gene in S' from S_2*/
repeat
  Insert genes from S_2 into S';
until a feasible offspring S' is reached;

4.1.4 Mutation operator

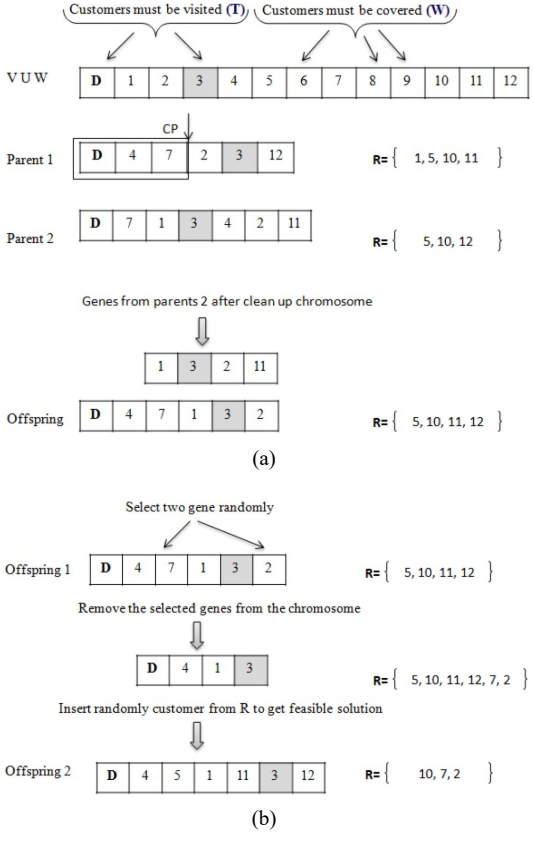
The mutation operator preserves the diversification in the population since it produces a neighbouring chromosome by changing some genes of the original one and prevent the algorithm to trapped in local optima. This operator consist of selecting randomly a set of customers and removing them from the solution. The mutation procedure developed

in our work is summarised in Algorithm 4 and then illustrated in Figure 4(b) where $cp = 3$ and one depot and 12 customers are considered.

Algorithm 4 $MUTATION(S_{new})$

1: Let I be the set of genes in the chromosome S_{new} and $R = V \backslash I$;
2: Remove randomly l genes from S_{new};
3: **repeat**
4: Select randomly a customer i from$R$;
5: Insert i randomly in S_{new};
6: **until** the chromosome S_{new} become feasible;

Figure 4 An example of a genetic operators, (a) an example of a crossover operator (b) an example of a mutation operator (see online version for colours)



4.1.5 Fitness function

The evaluation function gives us an idea about the quality of the solution which helps on comparing between individuals and gives the individual with best fitness more chance to survive. In this work, we define the evaluation function of an individual S_i by $FEVAL(S_i)$. In our relevant problem (m-CTP-p) the objective is to minimise

the total travel cost with respecting the capacity constraint under a restriction on the number of vertices in the solution. The cost of the individual S_i is represented by $Cost(S_i)$. In addition if the capacity constraint is violated a penalty will be added to our evaluation function. Let $PENALTY(S_i)$ be the penalty on the violation of the capacity constraints. Therefore, the evaluation function $FEVAL(S_i)$ of an individual S_i is calculated as follow:

$$FEVAL(S_i) = Cost(S_i) + PENALTY(S_i) \tag{1}$$

According to the value of the penalty the algorithm visit unfeasible solutions or visit only the feasible one. In this work, the $PENALTY(S_i)$ is computed as follows:

$$PENALTY(S_i) = \sum \alpha \max\{0, \textit{demand of customers} - \textit{capacity of the vehicles}\} \tag{2}$$

where α is a constant parameter that affects on the value of the penalty and thereafter on the algorithm. In this work, α is a constant taken as a big number to consider only feasible solutions.

4.1.6 The IIP method

In the following, we present the IIP hopping for an efficient update of the set of neighbouring solutions. This procedure has a role of diversification in our algorithm. It consist of relocating vretices and move from from one feasible solution to neighbouring solution. The neighbour detaied solutions are performed by swapping a customer from the solution with another one from the remaining set of customers. All the customers of W must be covered after any swap move. The IIP procedure is described in Algorithm 5 and then illustrated in Figure 5. After GA have been performed, we develop a LS method to improve the resulting offspring. In the sequel, we detail the local search (LS) method.

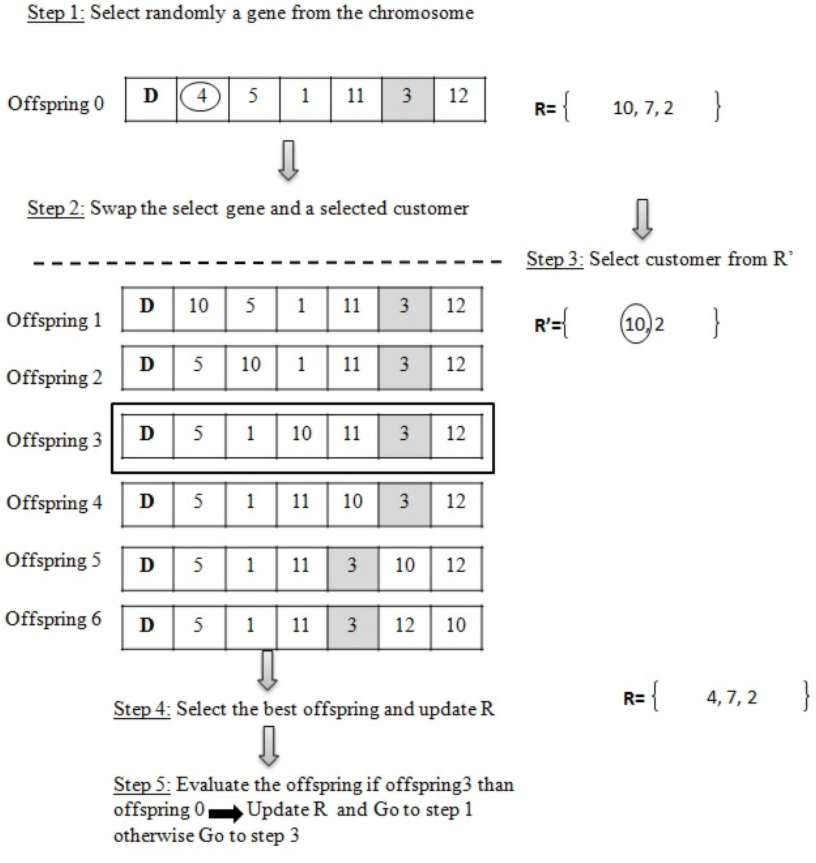
Algorithm 5 $IIP(S)$

repeat
  Select a gene i from the offspring S;
  let g be the set of gene in the offspring S and $R = V \backslash g$;
  Determine $R' \subset R$ that guarantee the feasibility of S by applying a swap move between a gene i and any element from R' to get a new offspring S'
  Select customer j from R'
  Swap i and j and find the best position of j in the chromosome S_i
  if $FEVAL(S_i') < FEVAL(S_i)$ **then**
    $S \leftarrow S'$;
    Update S and go back to step 2
  else
    Go back to step 6
  end if
until no possible improvement

In this example, customer 4 was selected from the chromosome and to maintain a feasible solution it can be replaced only by two customers from the remaining set

(customers 10 and 2). We select customer 10 from R' and then a swap move between customers 4 and 10 was performed. We notice that customer 10 try to find the best position in the chromosome. In step 4 we select the offspring 3 as the best chromosome. In the last step we compare the resulting offspring with the first one and if we have an improvement we update the remaining set by adding customer 4 in R and repeat the process until no possible improvement otherwise go to step 3 and swap customer 4 with another customer from R'.

Figure 5 The swap improvement method



4.1.7 Replacement phase

Finally we finish our GA by a replacement phase which is used to maintain good solutions in the population. Many methods in the literature are used to replace one individual from a population with another one. This can be done randomly or by removing a specific individual from the population. In this paper, our replacement procedure consists in selecting the individual with the worst fitness function and remove it from the population. In this phase we compare the worst individual with the new obtained offspring and keep the best one inside the population. The replacement procedure is presented in Algorithm 6.

Algorithm 6 $REPLACEMENT(S_{new1}, S_{worst}, Pop)$

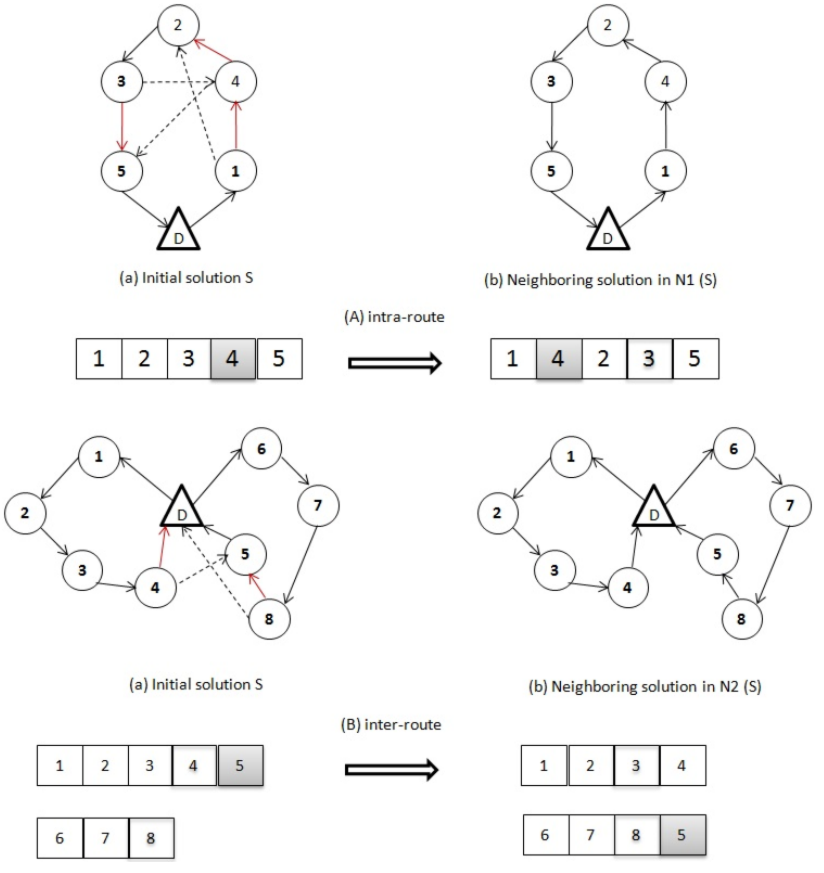
1: $S_{worst} \leftarrow S_{new1}$;
2: Update Pop;
3: Update S_{worst};
4: Update S_{best};

In the sequel, we propose an hybrid approach to improve the solution using a LS method.

4.2 Hybrid GA for m-CTP-p

The idea of combining the GA with a LS heuristic is initially used to intensify the search since GA guarantee only the diversification due to the use of many genetics operators. We propose to hybridise GA with a VND method that explores several neighbourhood structures in a deterministic way. In this section the hybridisation (HGA) is discussed.

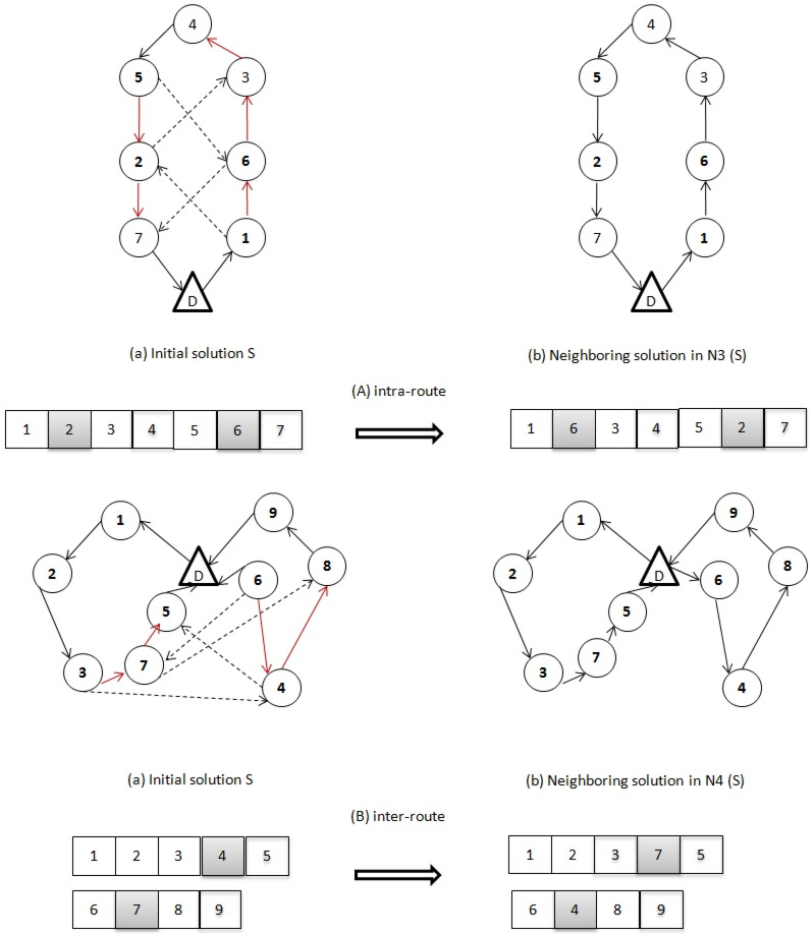
Figure 6 $N_1 - N_2$ insertion (see online version for colours)



Generally, a LS algorithm starts from an initial solution and then moves to a neighbour one by modifying some components of a given solution. Therefore, we cannot design a

VND method without defining the neighbourhood structure which aim at investigating a large number of solutions in a short time. The choice of the different neighbourhood structures is performed according to the studied problem. Relocating vertices between two routes are tested: neighbourhoods structures that removes one customer from a route and insert it in a different tour was used besides other neighbourhood that swaps the position of two customers from two different routes.

Figure 7 $N_3 - N_4$ swap (see online version for colours)



Let N_k, $k = 1, ..., k_{\max}$ be the set of neighbourhoods used in our VND algorithm and $N_k(S)$ be the neighbours of a solution S, w.r.t a neighbourhood structure N_k. In this work, we aim at generating a new neighbouring solution S' of S by exploring four neighbourhood structures: N_1, N_2, N_3 and N_4 based on two classical neighbourhoods. The classical insertion and swap moves are applied inside the same route and between two different routes.

Neighbourhood N_1 and N_2 use the INSERT operator which is employed either intra-route and inter route to get the first neighbouring solution S_1. This move changes the order of customers in those routes. For the neighbourhood N_1, the insertion

is carried out inside the same route by removing a customer and inserting it in a new position as shown in Figure 6(a). For neighbourhood N_2, the insertion move is performed between two different routes in which one customer is removed from its position and inserted into another route. Neighbourhood N_2 modify the corresponding routes not only the order of customers but also changes the number of customers in those routes. Figure 6(b) represents such move.

Algorithm 7 $VND(S)$

Input: initial solution S, neighbourhood structures N_k, $k = \{1, ..., k_{\max}\}$;
Begin
$k = 1$;
repeat
  Find S' the best neighbour of S using the neighbourhood structure N_k;
  if $FEVAL(S') < FEVAL(S)$ **then**
    $S' \leftarrow S$;
    $k = 1$;
  else
    $k = k + 1$;
  end if
until ($k > k_{\max}$);
End;

Algorithm 8 HGA for m-CTP-p

Initialise: $psize$ = 10, P_m = 0.1, P_c = 0.9, nbg = 100; /*Global parameters*/
Generate Pop; /*First generation of individuals*/
Calculate the evaluation function $FEVAL$ of each individual S_i in Pop;
Let S_{best} the individual with the best fitness;
Let S_{worst} the individual with the worst fitness;
while number of generations is not reached **do**
  $S_1 \leftarrow SELECT(Pop)$;
  $S_2 \leftarrow SELECT(Pop)$;
  $S_{new} \leftarrow CROSSOVER(S_1, S_2)$;
  $S'_{new} \leftarrow MUTATION(S_{new})$;
  $S''_{new} \leftarrow IIP(S'_{new})$;
  $S_{new1} \leftarrow VND(S''_{new})$;
  if $FEVAL(S_{new1}) < FEVAL(S_{worst})$ **then**
    $S_{best} \leftarrow REPLACEMENT(S_{new1}, S_{worst}, Pop)$;
  end if
end while

For neighbourhoods N_3 and N_4, a SWAP move is applied to $S_1$to get a second neighbouring solution S_2. This move is aimed at creating new solutions by interchanging customers in the same route or between different routes. For neighbourhood N_3, two customers inside the same route can be swapped while we consider one route and swap the positions of two customers inside the same route as shown in Figure 7(a). But for the neighbourhoods N_4, we consider two different routes and the swap move is performed by randomly selecting one customer from each route and exchanging them as shown in Figure 7(b).

The outline of our VND procedure is summarised in Algorithm 7. In both Figures 6 and 7 we first present the initial solution (a) in which a neighbourhood was applied by

modifying some components of this solution and then the new neighbouring solution (b) was created. The initial solution is represented by black lines while dashed lines correspond to the arcs removed from the original routes by applying one neighbourhood and red ones present the new arcs obtained after a move is applied.

Our resulting hybrid approach is summarised in Algorithm 8.

5 Computational results

A set of numerical experiments were performed using two benchmark sets of instances available in the literature. These instances was proposed by Há et al. (2013) and consists of 64 small instances and 32 large instances where the total number of customers is respectevely 100 and 200. Each of these instances was generated based on benchmark instances $KroA100$, $KroB100$, $KroC100$ and $KroD100$ of $TSPLIB$. These instances are first used to create a set of $nb_{total} = |V| + |W| = 100$ vertices. Tests are run for $n = [0.25\ nb_{total}]$ and $n = [0.5\ nb_{total}]$ and $|T| = 1$ and $|T| = [0.20\ n]$ and W is defined by taking the remaining point ($|W| = 100 - |V|$). As a second step, we use instances $KroA200$ and $KroB200$ with nb_{total} = 200 vertices to generate another set of instances. The instances are labelled $X - T - n - W - p$, where X is the name of $TSPLIB$ instance. To better understand, we give an example of a large instance which is derived from $KroB200$ of $TSPLIB$. Let B2-1-50-150-4 a generated instance with only one required vertex ($|T|$ = 1), 50 vertices that can be visited ($|V|$ = 50), 150 vertices that must be covered ($|W|$ = 150) and each tour must contain less than four vertices (p = 4). For further information about the considered instances, see Gendreau et al. (1997), Jozefowiez (2011) and Há et al. (2013).

Our algorithms described in the previous section was coded in C++ programming language and was run on a computer with an Intel Core i5-4200U and 2.3 Ghz processor and 6 GB memory.

A computational study is performed to evaluate the performance of our algorithms compared with the optimal solution and the solutions of Murakami (2018). In each benchmark instance, we show the results obtained using the GA, HGA and the results of other algorithms, as well as their computation time (in seconds). We also show the gap between our results and the results of the optimal solution and the results obtained using the ICG algorithm proposed by Murakami (2018) in each benchmark instance.

Let HA be the cost obtained by our HGA algorithm and BNC be the best khnown cost, i.e., the cost of the optimal solution or the value of the solution of the metaheuristic if optimal solutions could not be found. The percentage deviation of our method, Gap is computed as follows:

$$Gap = 100 \cdot (HA - BNC)/BNC \tag{3}$$

After a set of tests We also show the gap between the cost of our solution and the cost obtained using the ICG algorithm in each benchmark instance.The Gap is computed as follows:

$$Gap = 100 \cdot (HA - ICG)/ICG \tag{4}$$

We use the following genetic parameters in our experimentation. First we fixed the population size $psize$ = 10, i.e., at each population we have ten individuals (solutions

represented by chromosomes). Then we define the probability of genetic operations. We consider P_m = 0.1 the probability of mutation operator and P_c = 0.9 the probability of crossover operator. The algorithm was executed until 100 generations. At each generation we force our algorithm to visit only feasible solutions by fixing a large value of the parameter α, α = 1,000 in the $PENALTY(S)$.

Figure 8 Computation time for the GA, HGA and ICG algorithms

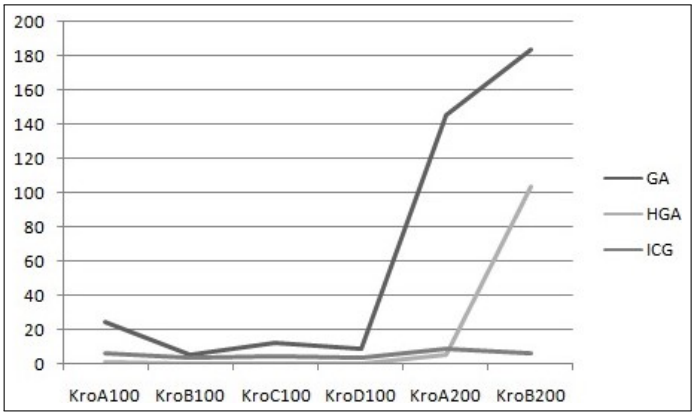
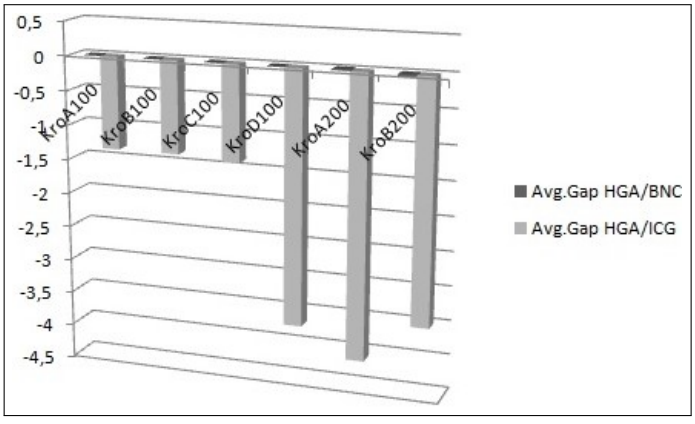


Figure 9 The average gap between the HGA and the results of Há et al. (2013) and Murakami (2018) algorithms



The results are summarised in Tables 1 and 2 for the test problems. Table 1 presents the results for 64 instances with 100 vertices and Table 2 presents the results for 32 instances with 200 vertices. The column headings are as follows: column *gap* shows the deviation between the cost of our solution and the cost obtained by Há et al. (2013) and Murakami respectively. Column $time$ is the running time in seconds. The italic numbers indicate the solution generated by GA that cannot be improved by the proposed hybrid approach. The numbers in bold indicate the optimal solution found by our HGA. The numbers marked with an asterisk correspond to feasible solutions obtained using Há's metaheuristic that not proved optimal. The underline numbers indicate improved results using HGA algorithm compared with feasible solutions of Há's metaheuristic.

Table 1 Computational results with 100 vertices of experiments based on HGA

Data	*GA*		*HGA*		*Optimal solution*		*ICG*		
	Result	*Time*	*Result*	*Time*	*Result*	*Gap*	*Result*	*Time*	*Gap*
A1-1-25-75-4	*8,479*	0.016	**8,479**	0.003	8,479	0	8,479	6.28	0
A1-1-25-75-5	*8,479*	0.025	**8,479**	0.01	8,479	0	8,481	5.09	–0.024
A1-1-25-75-6	*8,479*	0.06	**8,479**	0.001	8,479	0	8,481	3.39	–0.024
A1-1-25-75-8	*7,985*	0.104	**7,985**	0.003	7,985	0	7,985	3.05	0.000
A1-5-25-75-4	11,184	63.851	**10,827**	0.003	10,827	0	10,827	12.24	0.000
A1-5-25-75-5	9,341	2.409	**8,659**	0.002	8,659	0	8,659	11.69	0.000
A1-5-25-75-6	9,341	3.448	**8,659**	0.001	8,659	0	8,659	9.09	0.000
A1-5-25-75-8	9,341	1.674	**8,265**	0.003	8,265	0	8,265	2.6	0.000
A1-1-50-50-4	*10,271*	0.182	**10,271**	0.002	10,271	0	10,519	3.87	–2.358
A1-1-50-50-5	*9,220*	0.472	**9,220**	0.008	9,220	0	9,637	6.81	–4.327
A1-1-50-50-6	*9,130*	0.385	**9,130**	0.013	9,130	0	9,508	2.83	–3.976
A1-1-50-50-8	*9,130*	0.06	**9,130**	0.004	9,130	0	9,372	2.31	–2.582
A1-10-50-50-4	20,343	238.644	**17,953**	2.001	17,953	0	18,396	8.23	–2.408
A1-10-50-50-5	20,810	34.311	**15,440**	0.008	15,440	0	15,595	6.27	–0.994
A1-10-50-50-6	20,152	38.278	**14,064**	6.379	14,064	0	14,458	3.09	–2.725
A1-10-50-50-8	19,756	11.142	13,369	2.008	13,369*	0	13,641	4.43	–1.994
B1-1-25-75-4	*7,146*	0.067	**7,146**	0.005	7,146	0	7,146	6.02	0.000
B1-1-25-75-5	*6,901*	0.093	*6,901*	0.003	6,901	0	6,901	4.4	0.000
B1-1-25-75-6	*6,450*	0.038	**6,450**	0.001	6,450	0	6,450	1.48	0.000
B1-1-25-75-8	*6,450*	0.077	**6,450**	0.002	6,450	0	6,450	1.53	0.000
B1-5-25-75-4	*9,465*	1.497	**9,465**	0.002	9,465	0	9,465	3.73	0.000
B1-5-25-75-5	*9,460*	1.722	**9,460**	0.002	9,460	0	9,460	1.62	0.000
B1-5-25-75-6	9,460	19.356	**9,148**	0.001	9,148	0	9,287	5.6	–1.497
B1-5-25-75-8	9,156	5.196	**8,306**	0.002	8,306	0	8,599	3.97	–3.407
B1-1-50-50-4	*10,107*	0.455	**10,107**	0.023	10,107	0	10,255	4.28	–1.443
B1-1-50-50-5	*9,723*	0.002	**9,723**	0.007	9,723	0	9,729	3.19	–0.062
B1-1-50-50-6	*9,382*	0.441	**9,382**	0.017	9,382	0	9,580	2.42	–2.067
B1-1-50-50-8	*8,348*	37.078	**8,348**	0.048	8,348	0	9,075	2.62	–8.011
B1-10-50-50-4	18,350	0.969	**15,209**	0.003	15,209	0	15,283	2.11	–0.484
B1-10-50-50-5	17,847	0.948	**13,535**	0.076	13,535	0	13,684	2.8	–1.089
B1-10-50-50-6	17,870	11.469	**12,067**	0.029	12,067	0	12,403	4.19	–2.709
B1-10-50-50-8	16,308	3.324	**10,344**	1.443	10,344	0	10,409	3.7	–0.624
C1-1-25-75-4	*6,161*	0.002	**6,161**	0.001	6,161	0	6,283	1.24	–1.942
C1-1-25-75-5	*6,161*	0.001	**6,161**	0.001	6,161	0	6,288	6.58	–2.020
C1-1-25-75-6	*6,161*	0.001	**6,161**	0.001	6,161	0	6,161	1.52	0.000
C1-1-25-75-8	*6,161*	0.001	**6,161**	0.002	6,161	0	6,165	1.24	–0.065
C1-5-25-75-4	*9,898*	2.321	**9,898**	0.002	9,898	0	9,898	3.71	0.000
C1-5-25-75-5	9,898	11.882	**9,707**	0.002	9,707	0	9,707	4.82	0.000
C1-5-25-75-6	10,026	12.124	**9,321**	0.002	9,321	0	9,363	5.3	–0.449
C1-5-25-75-8	9,898	3.292	**7,474**	0.001	7,474	0	7,474	5.41	0.000
C1-1-50-50-4	*11,372*	0.093	**11,372**	0.015	11,372	0	11,626	5.68	–2.185
C1-1-50-50-5	*9,900*	0.425	**9,900**	0.002	9,900	0	9,914	3.69	–0.141

Table 1 Computational results with 100 vertices of experiments based on HGA (continued)

Data	*GA*		*HGA*		*Optimal solution*		*ICG*		
	Result	*Time*	*Result*	*Time*	*Result*	*Gap*	*Result*	*Time*	*Gap*
C1-1-50-50-6	*9,895*	0.008	**9,895**	0.002	9,895	0	9,903	2.1	–0.081
C1-1-50-50-8	*8,699*	29.058	**8,699**	0.005	8,699	0	9,355	1.83	–7.012
C1-10-50-50-4	22,247	27.174	**18,212**	1.179	18,212	0	18,288	6.89	–0.416
C1-10-50-50-5	22,519	1.824	**16,362**	2.005	16,362	0	16,485	7.14	–0.746
C1-10-50-50-6	20,809	18.132	**14,749**	0.078	14,749	0	14,927	4.55	–1.192
C1-10-50-50-8	21,205	84.688	**12,394**	2.03	12,394	0	13,183	3.91	–5.985
D1-1-25-75-4	*7,671*	0.023	**7,671**	0.004	7,671	0	7,729	6.03	–0.750
D1-1-25-75-5	*7,465*	0.311	**7,465**	0.013	7,465	0	7,922	1.56	–5.769
D1-1-25-75-6	*6,651*	0.063	**6,651**	0.005	6,651	0	6,709	2.93	–0.865
D1-1-25-75-8	*6,651*	0.16	**6,651**	0.003	6,651	0	6,730	2.23	–1.174
D1-5-25-75-4	13,279	9.74	**11,820**	0.001	11,820	0	12,498	2.24	–5.425
D1-5-25-75-5	11,943	0.172	**10,982**	0.001	10,982	0	11,526	4.59	–4.720
D1-5-25-75-6	12,040	27.412	**9,669**	0.003	9,669	0	9,676	4.93	–0.072
D1-5-25-75-8	11,569	12.533	**8,200**	0.002	8,200	0	8,334	4.02	–1.608
D1-1-50-50-4	*11,606*	15.53	**11,606**	0.018	11,606	0	11,606	3.13	0.000
D1-1-50-50-5	*10,770*	3.637	**10,770**	0.036	10,770	0	11,289	1.54	–4.597
D1-1-50-50-6	10,710	0.937	**10,525**	0.011	10,525	0	10,636	1.91	–1.044
D1-1-50-50-8	9,459	11.591	**9,361**	0.022	9,361	0	11,004	1.43	–14.931
D1-10-50-50-4	28,418	4.008	**20,982**	0.024	20,982	0	21,137	6.31	–0.733
D1-10-50-50-5	26,447	2.025	**18,576**	0.478	18,576	0	18,765	5	–1.007
D1-10-50-50-6	28,174	49.215	**16,330**	0.027	16,330	0	17,152	4.34	–4.792
D1-10-50-50-8	30,739	3.154	**14,204**	1.611	14,204	0	16,061	5.12	–11.562

Our results clearly show the performance of our approach. We noted that the GA takes a long execution time and cannot give all best known solutions whereas the hybridisation with VND improves the solution and accelerates the running time. In Figure 8, the average computation times of the GA algorithm become significantly larger when the scale of instance is increased (KroA200 and KroB200). On the other hand, the computation times of the HGA were not much larger, even with larger instances. This is one of the advantages of the LS heuristic.

It is clear that our hybrid algorithm performs better than our proposed GA. Our HGA can find all optimal solutions except one instance (A2-20-100-100-8) with small optimally gap 0.21. Our HGA perform better than the metaheuristic based on the ELS proposed by Há et al. (2013) and improves the solution in three instances whose optimal solution is unknown: B2-1-100-100-6, B2-20-100-100-4 and B2-20-100-100-8 with respectively gaps 0.39, 0.03 and 0.33. According to the computational experiments, our HGA out performs the ICG algorithm proposed by Murakami (2018) and competitive compared with the metaheuristic proposed by Há et al. (2013).

The comparison is performed for benchmark sets of instances. In Table 3, we report the results average values obtained over all the tested instances. The first and the second columns report the average gap (Avg.Gap) between our hybrid approach and the results of the solution obtained by by Hà et al. and Murakami respectively. The average gap between the cost of the optimal solution and the cost obtained using the HGA is smaller

than 5%. However, the Avg.Gap between the cost of our hybrid algorithm and the cost obtained using the ICG are more than 30%. Again, the average gaps show that for the easiest instances (with 100 nodes), the performance of the ICG algorithm provide worst solutions than our HGA. For the large instances (with 200 nodes), the gap was increased (see Figure 9). The ICG does not perform well in almost all the tested instances. We believe that the ICG is absolutely inferior to our hybrid algorithm.

Table 2 Computational results with 200 vertices of experiments based on HGA

Data	*GA*		*HGA*		*Optimal solution*		*ICG*		
	Result	*Time*	*Result*	*Time*	*Result*	*Gap*	*Result*	*Time*	*Gap*
A2-1-50-150-4	*11,550*	9.167	**11,550**	0.099	11,550	0	11,550	5.99	0.000
A2-1-50-150-5	*10,407*	0.223	**10,407**	0.012	10,407	0	10,491	7.25	–0.801
A2-1-50-150-6	*10,068*	3.72	**10,068**	2.318	10,068	0	10,483	7.43	–3.959
A2-1-50-150-8	8,995	14.945	**8,896**	0.03	8,896	0	10,695	6.32	–16.821
A2-10-50-150-4	23,189	5.157	**17,083**	0.006	17,083	0	17,230	30.04	–0.853
A2-10-50-150-5	23,705	25.876	**14,977**	0.023	14,977	0	15,108	10.42	–0.867
A2-10-50-150-6	22,209	31.714	**13,894**	0.447	13,894	0	14,245	5.52	–2.464
A2-10-50-150-8	21,650	18.732	**11,942**	0.305	11,942	0	11,963	6.44	–0.176
A2-1-100-100-4	*11,885*	0.365	**11,885**	0.065	11,885	0	12,533	2.52	–5.170
A2-1-100-100-5	*10,234*	0.874	**10,234**	0.183	10,234	0	10,937	7.41	–6.428
A2-1-100-100-6	*10,020*	7.144	10,020	0.045	10,020*	0	10,565	3.17	–5.159
A2-1-100-100-8	*9,093*	0.574	9,093	0.436	9,093*	0	10,494	4.97	–13.350
A2-20-100-100-4	46,326	35.051	26,594	4.155	26,594*	0	26,796	7.97	–0.754
A2-20-100-100-5	45,530	13.318	23,419	7.796	23,419*	0	23,882	6.17	–1.939
A2-20-100-100-6	44,976	1,078.364	20,966	38.258	20,966*	0	21,382	10.3	–1.946
A2-20-100-100-8	44,676	1,078.364	18,458	35.001	18,418*	0.217	19,413	21.21	–4.919
B2-1-50-150-4	*11,175*	24.738	**11,175**	0.088	11,175	0	11,322	3.69	–1.298
B2-1-50-150-5	*10,502*	26.111	**10,502**	0.359	10,502	0	10,974	4.54	–4.301
B2-1-50-150-6	*9,799*	12.578	**9,799**	0.045	9,799	0	10,734	3.49	–8.711
B2-1-50-150-8	*8,846*	52.701	**8,846**	0.145	8,846	0	9,196	3.95	–3.806
B2-10-50-150-4	19,181	278.648	**16,667**	0.117	16,667	0	16,849	5.93	–1.080
B2-10-50-150-5	18,391	119.237	**14,188**	0.131	14,188	0	14,299	8.01	–0.776
B2-10-50-150-6	17,804	154.208	**12,954**	0.131	12,954	0	13,021	6.73	–0.515
B2-10-50-150-8	18,102	110.802	**11,495**	0.379	11,495	0	11,684	5.24	–1.618
B2-1-100-100-4	19,170	486.064	**18,370**	1.857	18,370	0	18,560	10.32	–1.024
B2-1-100-100-5	16,519	44.806	**15,876**	9.203	15,876	0	16,651	6.04	–4.654
B2-1-100-100-6	15,629	197.587	14,867	0.6263	14,926*	–0.395	16,048	2.76	–7.359
B2-1-100-100-8	13,427	63.906	13,137	1.029	13,137*	0	14,592	4.68	–9.971
B2-20-100-100-4	51,497	320.829	34,062	12.178	34,073*	–0.032	34,301	7.97	–0.697
B2-20-100-100-5	51,228	818.119	29,412	365.325	29,412*	0	29,850	10.5	–1.467
B2-20-100-100-6	51,843	42.374	25,960	926.01	25,960*	0	26,488	8.03	–1.993
B2-20-100-100-8	50,986	185.05	22,082	342.01	22,156*	–0.333	23,697	11.35	–6.815

Table 3 The average gap of HGA with other algorithms on benchmark instances

Avg.Gap	*HGA/BNC*	*HGA/ICG*
KroA100	0	–1.338
KroB100	0	–1.337
KroC100	0	–1.390
KroD100	0	–3.691
KroA200	0.013	–4.100
KroB200	–0.047	–3.505

6 Conclusions

This paper deals with a challenging routing problem where the demands of all customers must be satisfied without visiting all of them. In this work, we solve a special case of this covering tour problem where a restriction on the number of vertices in each route was considered. We propose two approaches to solve the m-CTP-p. The main component of the proposed algorithms is an iterative improvement mechanism and a LS method that have two important features. The first feature is the possibility of improving the assignment decisions of customers to routes. The second feature is the ability to explore a set of neighbourhoods structures that substantially reduces the execution time of our algorithms. Computational results show that our proposed algorithms are competitive with existing meta-heuristics in the literature. Furthermore, the extensive experiments show the impact of the hybridisation to improves the results. The neighbourhood structures used in this algorithms give a promising results which encourage us to develop another approach based on other neighbourhood structures.

References

Baldacci, R., Boschetti, M.A., Maniezzo, V. and Zamboni, M. (2005) 'Scatter search methods for the covering tour problem', *Operations Research/Computer Science Interfaces*, Vol. 30.

Barkaoui, M. and Berger, J. (2020) 'A new hybrid genetic algorithm for the collection scheduling problem for a satellite constellation', *Journal of the Operational Research Society*, Vol. 71, No. 9, pp.1390–1410.

Braekers, K., Ramaekers, K. and Van Nieuwenhuyse, I. (2016) 'The vehicle routing problem: state of the art classification and review', *Computers and Industrial Engineering*, Vol. 99, pp.300–313.

Bräysy, O., Dullaert, W. and Gendreau, M. (2004) 'Evolutionary algorithms for the vehicle routing problem with time windows', *Journal of Heuristics*, Vol. 10, No. 6, pp.587–611.

Current, J.R. and Schilling, D.A. (1994) 'The median tour and maximal covering problems', *European Journal of Operational Research*, Vol. 73, No. 1, pp.114–126.

Current, J.R. (1981) *Multi Objective Design of Transportation Networks*, Department of Geography and Environmental Engineering, The Johns Hopkins University, Baltimore, MD.

De La Torre, L.E., Dolinskaya, I.S. and Smilowitz, K.R. (2012) 'Disaster relief routing: integrating research and practice', *Socio-Economic Planning Sciences, Special Issue: Disaster Planning and Logistics: Part 1*, Vol. 46, No. 1, pp.88–97.

Doerner, K.F. and Hartl, R.F. (2008) 'Health care logistics, emergency preparedness, and disaster relief: new challenges for routing problems with a focus on the Austrian situation', *Operations Research/Computer Science Interfaces*, Vol. 43, Springer, Boston, MA.

Flores-Garza, D.A., Salazar-Aguilar, M.A., Ngueveu, S.U. and Laporte, G. (2017) 'The multi-vehicle cumulative covering tour problem', *Annals of Operations Research*, Vol. 258, No. 2, pp.761–780.

Gendreau, M., Laporte, G. and Semet, F. (1997) 'The covering tour problem', *Operations Research*, Vol. 45, pp.568–576.

Há, M.H., Bostel, N., Langevin, A. and Rousseau, L-M. (2013) 'An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices', *European Journal of Operational Research*, Vol. 226, No. 2, pp.211–220.

Hachicha, M., Hodgson, M.J., Laporte, G. and Semet, F. (2000) 'Heuristics for the multi-vehicle covering tour problem', *Computers & Operations Research*, Vol. 27, No. 1, pp.29–42.

Jozefowiez, N., Semet, F. and El-Ghazali, T. (2007) 'The bi-objective covering tour problem', *Computers & Operations Research*, Vol. 34, pp.1929–1942.

Jozefowiez, N. (2011) 'A column generation approach for the multi-vehicle covering tour problem', *Proceedings of the 12th ROADEF Conference*.

Jozefowiez, N. (2015) 'A branch-and-price algorithm for the multivehicle covering tour problem', *Networks*, Vol. 64, No. 3, pp.160–168.

Kammoun, M., Derbel, H., Ratli, M. and Jarboui, B. (2015) 'A variable neighborhood search for solving the multi-vehicle covering tour problem', *Electronic Notes in Discrete Mathematics*, Vol. 47, pp.285–292.

Kammoun, M., Derbel, H. and Jarboui, B. (2019) 'A general variable neighborhood search with mixed VND for the multi-vehicle multi-covering tour problem', in Sifaleras, A., Salhi, S. and Brimberg, J. (Eds.): *Variable Neighborhood Search, ICVNS 2018, Lecture Notes in Computer Science*, Springer, Cham, Vol. 11328, pp.59–273.

Kammoun, M., Derbel, H. and Jarboui, B. (2020) 'Two meta-heuristics for solving the multi-vehicle multi-covering tour problem with a constraint on the number of vertices', *Yugoslav Journal of Operations Research*, Vol. 31, No. 3, pp.299–318.

Lopes, R., Souza, V.A.A. and da Cunha, A.S. (2013) 'A branch-and-price algorithm for the multivehicle covering tour problem', *Electronic Notes in Discrete Mathematics*, Vol. 44, pp.61–66.

Margolis, J.T., Song, Y. and Mason, S.J. (2019) *A Multi-Vehicle Covering Tour Problem with Speed Optimization*, arXiv preprint arXiv,1909.12435.

Murakami, K. (2018) 'Iterative column generation algorithm for generalized multi-vehicle covering tour problem', *Asia-Pacific Journal of Operational Research*, Vol. 35, No. 4, pp.1–22.

Ochelska-Mierzejewska, J., Poniszewska-Marańda, A. and Marańda, W. (2021) 'Selected genetic algorithms for vehicle routing problem solving', *Electronics*, Vol. 10, No. 24, p.3147.

Oliveira da Costa, P.R., Mauceri, S., Carroll, P. and Pallonetto, F. (2018) 'A genetic algorithm for a green vehicle routing problem', *Electronic Notes in Discrete Mathematics*, Vol. 64, pp.65–74.

Pham, T.A., Ha, M.H. and Nguyen, X.H. (2017) 'Solving the multi-vehicle multi-covering tour problem', *Computers and Operations Research*, Vol. 88, pp.258–278.

Simms, J.C. (1989) 'Fixed and mobile facilities in dairy practice. The veterinary clinics of North America', *Food Animal Practice*, Vol. 5, No. 3, pp.591–601.

Tricoire, F., Graf, A. and Gutjahr, W.J. (2012) 'The bi-objective stochastic covering tour problem', *Computers & Operations Research*, Vol. 39, No. 7, pp.1582–1592.

Vargas Suarez, L.G. (2012) *A Dynamic Programming Operator for Metaheuristics to Solve Vehicle Routing Problems with Optional Visits*, INSA de Toulouse, Theses.

Viana, M.S., Morandin Jr., O. and Contreras, R.C. (2020) 'A modified genetic algorithm with local search strategies and multi-crossover operator for job shop scheduling problem', *Sensors*, Vol. 20, No. 18, p.5440.