Solving Various Classes of Arc Routing Problems with a Memetic Algorithm-based Framework

Sasan Mahmoudinazlou[1], Changhyun Kwon[2], Hadi Charkhgard[1*]

[1]Department of Industrial and Management Systems Engineering, University of South Florida, Tampa, FL 33620.
[2]Department of Industrial and Systems Engineering, KAIST, Daejeon, South Korea.

*Corresponding author(s). E-mail(s): h.charkhgard@gmail.com;
Contributing authors: sasanm@usf.edu; chkwon@kaist.ac.kr;

Abstract

Arc routing problems are combinatorial optimization problems that have many real-world applications, such as mail delivery, snow plowing, and waste collection. Various variants of this problem are available, as well as algorithms intended to solve them heuristically or exactly. Presented here is a generic algorithmic framework that can be applied to a variety of arc routing problems where a fleet of vehicles is used to visit a predefined set of edges. The main characteristic of the problem that qualifies it for the proposed framework is that each edge should be visited no more than once. This proposed framework uses genetic algorithms, dynamic programming, and local searches in a systematic manner and provides guidelines for applying them to the arc routing problem of one's choice. We select two problems to test the effectiveness of our proposed framework: the min-max windy K-vehicle rural postman problem and the undirected capacitated arc routing problem. We implement our proposed framework and compare it with existing algorithms using an established benchmark set for each problem. We demonstrate that our generic proposed framework can outperform existing custom-built algorithms.

Keywords: arc routing, Chinese postman problem, memetic algorithm, metaheuristic, Dynamic programming

1 Introduction

Arc routing problems are fundamental in operations research and logistics, covering a wide range of real-world applications, including garbage collection, mail delivery, school bus route planning, and snow plowing [1, 2]. These problems involve determining optimal routes for a fleet of vehicles to traverse a network of arcs while satisfying various constraints. A considerable amount of research has been conducted over the years in order to address different aspects of arc routing problems. For a comprehensive overview of the existing literature, interested readers may refer to a review paper on arc routing problems by Corberán et al. [3] and a survey of arc routing problems under uncertainty by De Maio et al. [4]. In summary, the existing literature can be categorized from at least three perspectives: the number of vehicles, arc directions, and the number of visits.

A significant amount of the literature on arc routing has focused on single-vehicle routing problems, such as the well-known Chinese Postman Problem (CPP) introduced by a Chinese mathematician, Meigu Guan (Mei-Ko Kwan), and the Rural Postman Problem (RPP) proposed by Orloff [5]. CPP is characterized by the objective of finding the shortest tour that traverses each arc at least once while minimizing the total distance. The RPP differs slightly in that not all of the arcs are required for visiting. While these problems have received significant attention, they represent just one facet of the broader arc routing class. There is also a category of arc routing problems involving multiple vehicles, where a fleet of vehicles must service a set of arcs. As an example, the Capacitated Arc Routing Problem (CARP) introduced by Golden and Wong [6], is a well-studied multi-vehicle arc routing problem. CARP involves visiting a set of edges with specific demands. Vehicles are homogeneous and have a limited capacity. Some of the existing solutions approaches in this category are an exact approach by Bartolini et al. [7] that employs set partitioning, the transformation to the Capacitated Vehicle Routing Problem (CVRP) by [8], as well as heuristic methods such as Tabu Search [9], Genetic Algorithm [10], and Ant Colony Optimization [11].

Aside from the number of vehicles, arc routing studies can be classified based on the capability to traverse the arcs in one direction or in two directions. Thus, arc routing problems can be classified into three categories: directed, undirected, and mixed. The arcs in directed arc routing problems have a specific direction, whereas the arcs in undirected arc routing problems may be visited in any direction, and in mixed arc routing problems, the graph may contain both directed and undirected arcs. The problems of this category are also solvable by various optimization methods. Heuristic methods by Maniezzo and Roffilli [12] for directed CARP, exact methods by Belenguer et al. [13] for mixed CARP, and transformation to node-routing by Baldacci and Maniezzo [14] for undirected CARP are some examples of this category.

Lastly, another type of classification for arc routing problems is based on the number of visits to each arc. In the first category of arc routing problems (referred to as the "single-visit" category), each edge must only be visited once, e.g., CPP, RPP, and CARP. The second type of arc routing problem (referred to as the "at-most-one-visit" category) is one in which each edge should be visited at most once, meaning that some of the edges may be left unvisited. Problems in this category aim to maximize profit, for example, CARP with profit [15]. The third category (referred

to as the "multiple-visit" category) consists of problems in which we are permitted to visit edges more than once. The Profitable Arc Tour Problem [16] is an example of this type of problem, in which profits can be collected multiple times from arcs. Some of the existing solution approaches in this category are exact algorithms such as Branch-and-Price by Feillet et al. [16] and heuristic algorithms such as a hybrid Metahueristic by Euchi and Chabchoub [17] for the Profitable Arc Tour Problem.

From the existing body of literature, one can observe that the classes of arc routing problems mentioned above are typically investigated in isolation. As a result, the existing solution methods are often customized for specific classes, creating a gap in the development of adaptable algorithms capable of addressing a wide spectrum of problems. This study aims to bridge this gap by introducing a generic algorithmic framework designed to tackle a range of arc routing problems, accommodating any number of vehicles and arc direction formats within the single-visit or at-most-one-visit categories. The only limitation of the proposed framework is its applicability to the multiple-visit category.

Our proposed generic framework is based on Memetic Algorithms (MA). This framework leverages the power of genetic algorithms, dynamic programming, and local search techniques to provide efficient and effective solutions. To address an arc routing problem involving a fleet of vehicles, three levels of decision-making must be considered. The first level involves assigning each arc to a vehicle, the second level determines the order in which the arcs are visited during each tour, and the third level deals with the direction of the arcs, particularly in undirected and mixed graphs. The second and third decisions, namely the order and direction of arcs, are handled by genetic algorithms (GAs), using a direction-aware chromosome encoding. The first decision, which involves assigning arcs to vehicles, is addressed using dynamic programming. To evaluate the effectiveness of our algorithm, we have chosen two different arc routing problems with distinct objectives and constraints. We compare the results of our framework to those of existing methods by solving an established benchmark set for each problem.

The rest of this paper is organized as follows: In Section 2, we will formally introduce the class of problems to which our proposed framework is intended to solve. A discussion of the conditions to check to determine whether a problem falls within this category will also be provided. The details of our generic framework will be presented in Section 3. Section 4 presents the computational results for two benchmark sets. The paper is concluded and future directions are suggested in Section 5.

2 Problem Description

This section provides a brief description of the classes of arc routing problems suitable for our proposed algorithmic framework. Consider a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, n\}$ represents the set of vertices and $\mathcal{E}$ represents the set of edges. The depot, denoted by vertex V_0, serves as both the starting and ending point for each tour. The graph G can be directed, undirected, or mixed. Each edge $e \in \mathcal{E}$ is associated with a travel time t_e (or cost c_e), and these travel times adhere to the triangle inequality. In the presence of undirected edges, as seen in the windy arc routing problem [18], each

undirected edge will possess two different travel times (costs). Within this context, there exists a subset of edges, labeled as E^R, which must be traversed; these are referred to as the required edges. For certain problem classes, a distinct subset of edges is referred to as profitable edges, rather than required edges. In such cases, our objective is to visit the edges that yield the highest profit, rather than traversing all the edges in the set.

Our proposed framework is designed to solve arc routing problems in which edges in a set of required or profitable edges may only be traversed once. In some arc routing problems, multiple visits are permitted. Considering that our proposed framework encodes a solution in a fixed-size array that includes all of the required edges (or profitable edges), multi-service arc routing problems cannot be addressed by our proposed framework. We can name three such problems as follows: maximum benefit Chinese postman problem [19], profitable arc tour problem [17], and generalized maximum benefit multiple Chinese postman problem [20]. There is also a type of arc routing problem in which the goal is to provide service to nodes. A traversal of each arc, however, involves serving the customers of the nodes adjacent to it. The problem is referred to as the Close Enough Arc Routing Problem (CEARP) [21]. This is also a problem that cannot be solved by our proposed framework.

The proposed framework can accommodate any arbitrary number of vehicles, denoted as m. Furthermore, it can be tailored to accommodate various common constraints. Two such constraints, which we use for illustration in this study, are the maximum tour length and capacity restrictions. Assume that d_j represents the demand for edge e_j, and C represents the maximum capacity of each vehicle. The sum of d_j across all edges in a tour must not exceed C. In a similar manner, travel time constraints can be defined. Let L be the maximum duration allowed for each tour. The total travel time for each tour must be less than or equal to L. For problems without capacity constraints, we can simply set $C = \infty$ (similarly for L).

In terms of the objective function, the class of arc routing problems solvable by our proposed framework may vary from minimizing the total cost and minimizing the maximum tour length to maximizing the total profit. Section 3.2 provides details regarding the proper handling of the objective function. In light of all that, the proposed framework can solve a wide range of problems relating to arc routing, including the min-max windy k-rural postman problem [22], the capacitated arc routing problem [6], the undirected capacitated arc routing problem with profits [15], the team orienteering arc routing problem [23], among others.

In summary, the main limitation of our proposed framework arises when there are edges that require multi-service. Otherwise, our generic framework can be used. In Section 3, we will provide explanations about how the proposed framework can handle various constraints and objective functions for problem classes that can be solved by our intended framework. To demonstrate the effectiveness of the proposed framework, in Section 4, we introduce two different problems, including the min-max windy K-vehicle rural postman problem and the undirected capacitated arc routing problem, and illustrate the performance of our proposed framework.

3 A Memetic Algorithm-based Framework

The purpose of this section is to provide a detailed explanation of the architecture of our framework, beginning with the genetic algorithm. Algorithm 1 provides a detailed description of the MA. We begin by generating μ individuals (line 1) where μ is a user-defined parameter that indicates the minimum population size. Generating initial individuals can be performed randomly. However, the performance of evolutionary algorithms is generally improved when they start with a population that is well-fitted. In this regard, it is recommended to design a procedure that can generate high-quality initial solutions. Until the stopping condition is satisfied, each of the following steps is repeated. When It_{STOP} iterations have been completed without improvement, the algorithm terminates. The first step of the process is to sort the population according to fitness (objective function) and a diversity multiplier in order to measure differences between individuals (line 3). Diversification multipliers are specific to each problem and will be discussed in more detail. Afterwards, two individuals will be selected from the population as parents (line 4). To select the parents for this study, *Tournament Selection* was used. Tournament selection involves selecting a subset of individuals at random from a population. In this paper, $k_{\text{TOURNAMENT}}$ refers to the number of individuals within the subset, which is a parameter that is defined by the user. The individuals within this subset compete against each other, and the one with the best fitness value is selected to be a parent. The same process is followed for selecting the second parent. When crossover is applied to two parents, one child will be produced (line 5), for which the SPLIT algorithm is employed for fitness evaluation and tour determination (line 6). Afterward, the new offspring will be improved by applying neighborhood search functions (line 7). A survival plan is implemented once the population reaches $\mu + \lambda$. According to fitness value, the best μ individuals are retained and the rest are discarded (lines 9-11). If no improvement is observed after It_{DIV} iterations, the population will be diversified in order to help the algorithm avoid local optima. During this step, the top n_{BEST} individuals are kept, the rest are discarded, and new individuals are generated until the population size reaches μ. A similar process is used to create new individuals (lines 12-14) as was used to generate the initial population.

3.1 Chromosome representation

Based on the generic MA framework proposed in this study, any arc routing problem can be solved when the required arcs are assumed to be visited at most once. In this regard, chromosome encoding is assumed to include the order in which arcs must be arranged as well as their direction. Orders are incorporated by permuting the required arcs, while arc directions are handled by the signs of these numbers. For instance, $[2, -6, 5, 1, -3, 4]$ represents the chromosome encoding of an instance with 6 required arcs in an undirected graph (Figure 1). When representing an individual on a directed graph, all signs should be positive. In a mixed graph, the signs for directed arcs in the representation should always be positive, while the signs for undirected arcs can be either positive or negative. In this example, the arcs are ordered as their absolute values appear in the representation, i.e., $[E_2, E_6, E_5, E_1, E_3, E_4]$. It is the sign of each gene that determines the direction. Suppose edge E_1 was originally intended to connect

Algorithm 1 Memetic Algorithm

$\Omega =$ `initial_population()`
while Stopping condition is not met **do**
  `sort`(Ω) ▷ Based on fitness and diversification factor
  Select ω_1 and ω_2 from Ω
  $c \leftarrow$ `crossover`(ω_1, ω_2)
  $\omega =$ Split(c) ▷ Algorithm 2
  `educate`(ω) ▷ Neighborhood search
  $\Omega_{\text{F}} \leftarrow \Omega_{\text{F}} \cup \{\omega\}$
  if `size`$(\Omega) = \mu + \lambda$ **then**
    `select_survivors`(Ω)
  end if
  if `best`(Ω) not improved for It_{DIV} iterations **then**
    `diversify`(Ω)
  end if
end while
Return `best`(Ω)

vertex V_1 to vertex V_2, in which case $+1$ indicates $V_1 \rightarrow V_2$ and -1 indicates $V_2 \rightarrow V_1$. Let E_1, E_2, E_3, E_4, E_5 and E_6 represent $V_1 \rightarrow V_2, V_0 \rightarrow V_8, V_3 \rightarrow V_4, V_5 \rightarrow V_6, V_9 \rightarrow V_{10}$ and $V_7 \rightarrow V_8$ respectively. Assume there are two vehicles for visiting these arcs, then the sequence $[2, -6, 5, 1, -3, 4]$ can be split into two tours, each tour will be handled by one vehicle. Letting $[2, -6, 5]$ and $[1, -3, 4]$ represent tours for each of the two vehicles, they will respectively be equivalent to $V_0 \rightarrow V_8 \rightarrow V_7 \xrightarrow{s} V_9 \rightarrow V_{10} \xrightarrow{s} V_0$ and $V_0 \xrightarrow{s} V_1 \rightarrow V_2 \xrightarrow{s} V_4 \rightarrow V_3 \xrightarrow{s} V_5 \rightarrow V_6 \xrightarrow{s} V_0$, where $V_i \rightarrow V_j$ means traversing the required arc starting from V_i ending in V_j and $V_i \xrightarrow{s} V_j$ means the shortest path from V_i to V_j. Detailed information regarding the procedure for splitting the tour into several tours will be provided in the following section. Dijstra's algorithm [24] can be used to find the shortest path between all pairs of nodes at the beginning of the computation before running MA.

3.2 Individual evaluation by Split

This study proposes a generalized Split algorithm similar to what Prins [25] introduced for vehicle routing problems. We intend to design a dynamic programming-based framework for arc routing problems involving multiple vehicles. The algorithm must be able to take a sequence of edges and, without changing their sequence, optimally split them into multiple tours regardless of the objective function. Our Split has been formulated in a way that accommodates both vehicle capacity constraints and time limitations (travel distances) for each tour. The addition of any other constraint is possible as well. A constraint can easily be deactivated by setting the right hand side to a large number.

A simple illustration of our proposed split algorithm can be found in Figure 2. The algorithm takes a chromosome representation and determines the delimiters for dividing the sequence of edges into m tours, where m represents the number of vehicles.

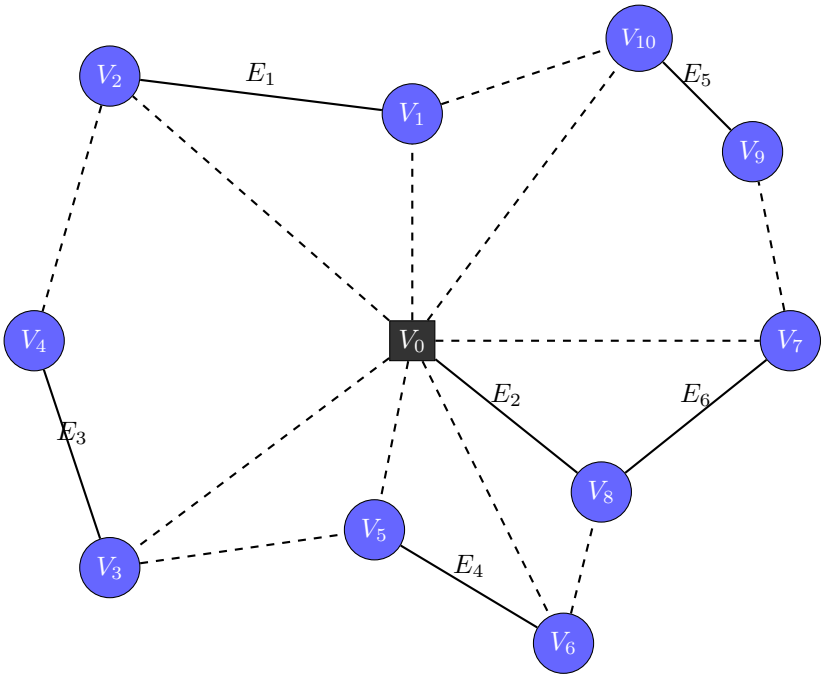


Fig. 1: An arc routing problem instance with six required edges.

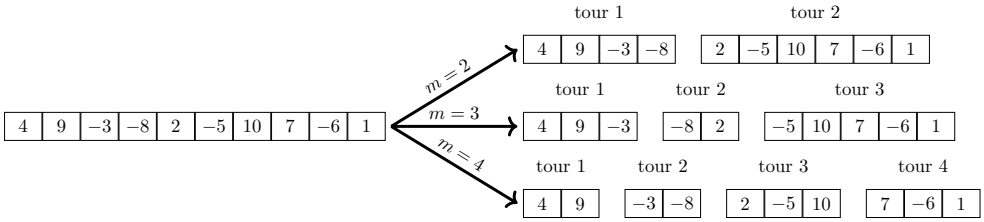


Fig. 2: Illustration of SPLIT algorithm.

Developing a dynamic programming algorithm begins with reducing the problem into smaller subproblems. Let $\alpha(k, r)$ be the problem of optimally splitting the first k required edges in the chromosome into r tours. In that case, the main problem would be equivalent to $\alpha(N_{\text{req}}, m)$, where N_{req} represents the number of all required edges and m represents the number of vehicles. Bellman's optimality equations will thus complete the dynamic programming algorithm which solves $\alpha(N_{\text{req}}, m)$ by solving the subproblems. Below is a list of notation used in the SPLIT algorithm for the convenience of the reader:

- $\boldsymbol{S} = [S_1, S_2, ..., S_{N_{\text{req}}}]$: The sequence of required edges in the chromosome. (e.g. $\boldsymbol{S} = [2, -6, 5, 1, -3, 4]$, explained in previous section)
- m: Number of vehicles; Obviously $m \leq N_{\text{req}}$.

- V_r^k: The optimal objective value of subproblem $\alpha(k, r)$, i.e., completing the subsequence $[S_1, ..., S_k]$ in r tours.
- $F(V_r^{k-1}, j)$: The objective value obtained by adding a tour containing $[S_k, ..., S_j]$ to the optimal solution of problem $\alpha(k-1, r)$. This value depends on the objective function of each specific problem.
- R_k: Set of all possible numbers of tours for visiting first k required edges, where

$$R_k = \begin{cases} \{0\} & \text{if } k = 0, \\ \{1, ..., \min(k, m)\} & \text{if } 1 \leq k \leq N_{\text{req}}, \end{cases}$$

  For example, having $m \geq 3$ vehicles, $R_3 = \{1, 2, 3\}$ since visiting three arcs needs at least one tour and it is infeasible to split them into more than three tours.
- t_{S_j}: The time (distance/cost) of traveling edge S_j
- τ_{S_i,S_j}: The time (distance/cost) of shortest path from the end point of edge S_i to the start point of edge S_j
  The start and end point of an edge depends on the sign of S_j. For example, if $S_j = +3$ and edge E_3 represents $V_1 \to V_2$, then the start and end point of this edge would be V_1 and V_2 respectively, and if $S_j = -3$ for the same edge E_3, then the start and end point would be V_2 and V_1 respectively. Note that if edge E_3 is directed, the corresponding S_j in the representation will always be positive.
- d_{S_j}: Demand of edge S_j
- L: The maximum time (distance/cost) allowed for a tour. If no such constraint exists, then $L = \infty$.
- C: The capacity of each vehicle. In case there is no capacity constraint, then $C = \infty$.

For each $r \in \{1, \ldots, m-1\}$ and $j \in \{r+1, \ldots, N_{\text{req}}\}$, the goal of the dynamic program is to establish an optimal solution of the problem $\alpha(j, r+1)$ by considering the possibility of adding one additional tour containing the subsequent $[S_k, \ldots, S_j]$ to the solution obtained for the problem $\alpha(k-1, r)$ for all $k \in \{r+1, \ldots, j\}$, assuming that they are all solved to optimality. More specifically, for each $r \in \{1, \ldots, m-1\}$ and $j \in \{r+1, \ldots, N_{\text{req}}\}$, by assuming that V_r^{k-1} is known for all $k \in \{r+1, \ldots, j\}$, the Bellman optimality equation can be stated as

$$V_{r+1}^j = \min_{k=r+1,\ldots,j} \text{or} \max_{k=r+1,\ldots,j} \left\{ F(V_r^{k-1}, j) \right\}.$$

The choice of "min" or "max" depends on the problem and the sense of the objective function being explored. For example, if the objective function is to maximize the total profit, then "max" should be selected and we can define

$$F(V_r^{k-1}, j) = V_r^{k-1} + \sum_{i=k}^{j} p_{S_i}.$$

where where p_{S_i} is the profit obtained by traversing edge S_i. Similarly, if the objective function is to minimize the maximum tour length, then "min" should be selected and

Algorithm 2 SPLIT for multi-vehicle arc routing (in minimization form)

1: $V_0^0 \leftarrow 0$
2: $V_r^k \leftarrow +\infty \quad \forall k = 1$ to N_{req} , $\forall r$ in R_k ▷ $V_r^k \leftarrow -\infty$ for maximization
3: **for** $k = 1$ to N_{req} **do**
4: **for** $r \in R_{k-1}$ **do**
5: **if** $V_r^{k-1} < +\infty$ **then** ▷ $V_r^{k-1} > 0$ for maximization
6: `time` $\leftarrow 0$; `load` $\leftarrow 0$; $j \leftarrow k$
7: **while** $j \leq N_{\text{req}}$ **do**
8: **if** $k = j$ **then**
9: `time` $\leftarrow \tau_{0,S_j} + t_{S_j} + \tau_{S_j,0}$ ▷ 0 represents depot.
10: **else**
11: `time` $\leftarrow$ `time` $- \tau_{S_{j-1},0} + \tau_{S_{j-1},S_j} + t_{S_j} + \tau_{S_j,0}$
12: **end if**
13: `load` $\leftarrow$ `load` $+ d_{S_j}$
14: **if** `time` $> L$ or `load` $> C$ **then**
15: `break`
16: **end if**
17: **if** $r + 1 \in R_j$ **then**
18: **if** $F(V_r^{k-1}, j)$ `better than` V_{r+1}^j **then**
19: $V_{r+1}^j \leftarrow F(V_r^{k-1}, j)$ ▷ F depends on the objective function
20: **end if**
21: **end if**
22: $j \leftarrow j + 1$
23: **end while**
24: **end if**
25: **end for**
26: **end for**
27: return `best`(V_m^i)

we can define

$$F(V_r^{k-1}, j) = \max(V_r^{k-1}, T_{k,j})$$

where $T_{k+1,j}$ is the length of the tour that contains the subsequent $[S_k, ..., S_j]$.

In light of the above, the proposed SPLIT algorithm begins by initializing V_0^0 with 0 and all V_r^k with an undesirable value. Depending on whether the objective function should be minimized or maximized, the value is $+\infty$ or $-\infty$. As a result of a forward propagation, we are aiming to solve $\alpha(N_{\text{req}}, m)$ by starting with $\alpha(0, 0)$. The condition in Line 5 ensures that the problem $\alpha(k-1, r)$ is feasible to begin with. This infeasibility might be due to violation of a constraint. Variables `time` and `load` are used to record the travel time (distance/cost) and the vehicle load for the current tour that is being added. These variables are intended to ensure that the travel time and capacity constraints are not violated (Lines 14-16). If users wish to impose additional constraints, they can define them in a similar manner to capacity and time within the proposed framework. The algorithm compares the new solution of the problem

$\alpha(j, r+1)$ with the existing one while adding each edge S_j to the current tour, and if it is more satisfactory, it replaces the existing solution.

As a final note on the SPLIT algorithm, it should be noted that in some arc routing problems, not all arcs can be traversed in the final solution. Specifically, the problems with the objective of maximizing the total profit involve selecting the most profitable arcs and ordering them in different tours. Consequently, it may not be possible to solve the $\alpha(N_{\text{req}}, m)$ problem in this case. We will therefore select $\alpha(k, m)$, which provides the best objective value. In the SPLIT algorithm, arcs are selected in the same order as they appear in S until forward propagation is not possible due to constraints being violated.

3.3 Crossover

Based on our chromosome encoding, any permutation-based crossover can be applied with some modifications to handle both the signs as well as the order. It is, however, recommended to use the Similar Tour Crossover (STX) described in Mahmoudinazlou and Kwon [26], which is specially designed to address problems associated with multiple tours. From the first parent, a random tour is selected, followed by the tour with the greatest number of mutual edges from the second parent. A simple two-point crossover is then implemented between selected tours, and the new tour is then added to the child. Until the child has m tours, this process should be repeated. It is possible that some edges will appear more than once in the tours at the end of the process. In addition to removing repeating edges, any edges that could violate the time or capacity constraint will also be removed from the child's tour. Ultimately, all remaining edges will be assigned to one of the child's tours based on a greedy approach, to the extent that constraints allow. It is necessary to examine all possible insertions for each edge, and then the edge should be placed according to the objective function in the position and direction (sign) that is most appropriate. Furthermore, the edges that are not able to be placed on any of the tours due to constraints must be inserted at the end in order to keep all the edges in play.

3.4 Chromosome education

In evolutionary algorithms, improving the offspring by employing local search methods tends to lead to a faster convergence to a solution of higher quality. The neighborhoods that can be used for the class of arc routing problems that we are addressing in this paper may be classified into three categories. Firstly, there are *intra-route* neighborhoods, which include movements within each tour. The *reinsert* neighborhood consists of taking an edge from a tour and repositioning it on the same tour. An *exchange* move alters two edges within a tour. A moving edge may be examined in both directions (signs, in our representation). A *two-opt* move reverses the order of a subsequence of a tour. A *one-reverse* move inverts the direction (changes the sign in our representation) of one edge in a tour. These four neighborhoods are examples of intra-route local searches. Considering other neighborhoods and selecting the best ones for the problem of one's choice is highly recommended.

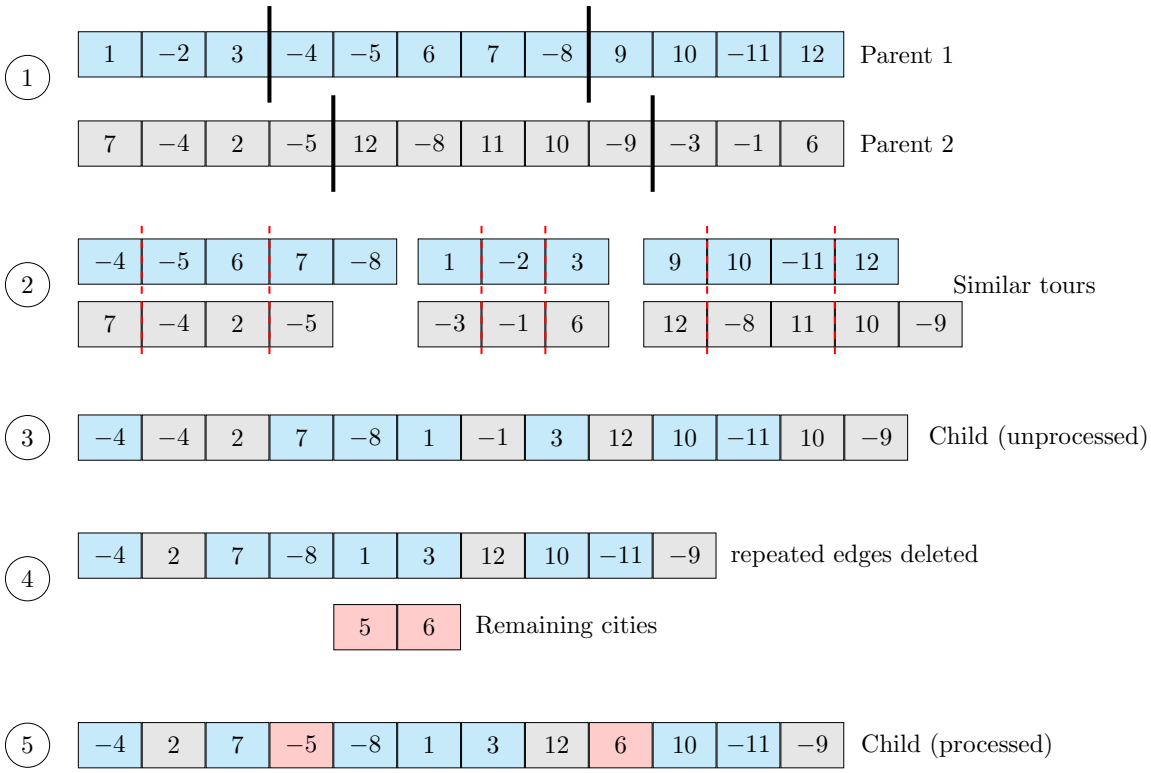
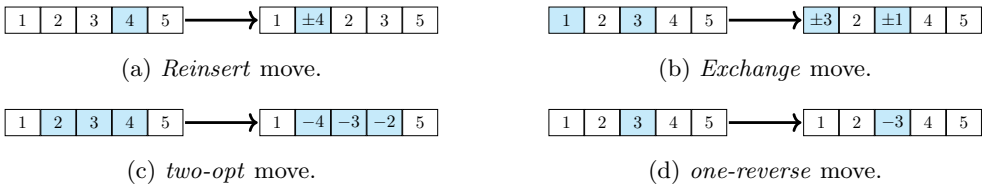


Fig. 3: Illustration of STX crossover for a problem with size $N_{\text{req}} = 12$ and $m = 3$.



(a) *Reinsert* move.

(b) *Exchange* move.

(c) *two-opt* move.

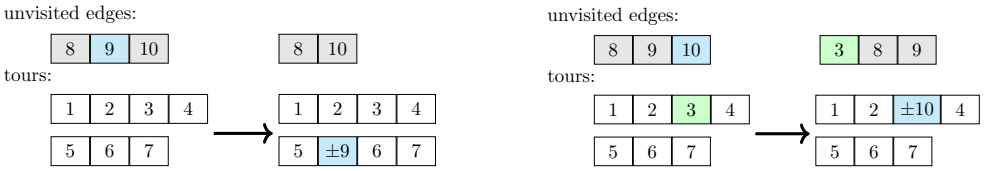
(d) *one-reverse* move.

Fig. 4: Intra-route neighborhoods.

(a) *shift(1,0)* move.

(b) *swap(1,1)* move.

Fig. 5: Inter-route neighborhoods.



(a) *add-one* move. (b) *substitute* move.

Fig. 6: Outer neighborhoods.

Another category is *inter-route* neighborhoods, which require two different tours. *Shift(1,0)* and *swap(1,1)* are examples of this type of operation. A shift(1,0) transfers one edge from one tour to another tour, while a swap(1,1) exchanges two edges from two different tours. One can also use other types of inter-route local searches, such as shift(2,0) or swap(2,1), etc. The third neighborhood category, which we refer to as the *outer* neighborhood, only applies to problems that do not necessarily traverse all edges. In problems where the objective is to maximize total profit, some of the edges may not be selected in the final solution. In this regard, adding one of the unvisited edges to one of the tours might be a potential improvement. Another option would be to swap one of the unvisited edges with an edge on one of the tours. We refer to theses moves as *add-one* and *substitute* moves respectively. Similarly, other movements are possible in the outer neighborhood category.

The next step is to determine how and in what order these local searches should be conducted. Local searches are sometimes performed at random in some studies. We recommend, however, using these local searches in a systematic manner in order to maximize their effectiveness. We begin by employing all intra-route neighborhoods one by one. Our method entails taking a chromosome and implementing all possible moves within one intra-route search until there is no further improvement. We repeat the process for the next intra-route neighborhood search. In addition, we keep track of the indices of tours that are being improved by any of the changes. The process is repeated until no further improvement is possible as a result of intra-route changes. A similar search is then conducted with inter-route movements, and the improved tours are recorded. Finally, moves within the outer neighborhood are similarly applied. Afterward, for tours that have been improved by any move, the whole process is repeated again. As soon as the entire iteration of the search does not result in an improved tour, the process is terminated.

3.5 Diversification

Typically, genetic algorithms have a tendency to generate populations that closely resemble the best individual, especially when the populations are small. This can result in the algorithm becoming stuck in local optima. Having a more diverse population increases the likelihood of reaching global optima because it provides more opportunities for exploration.

To further diversify the population, we evaluate the chromosome's fitness based on the solution's objective value, as well as its distance from its adjacent chromosomes

after sorting by fitness. As a first step, we convert the multi-vehicle arc routing solution into a rank-based representation in order to quantify the distance between the two individuals. Each rank-based representation of an individual consists of an array of size n representing the position of all required edges in any tour. If an edge is not included in any tour as a result of constraints, the corresponding value will be zero. Consider $[4, -3, 8], [-1, 5, -10, -7], [6, -2]$ as the tours of a multi-vehicle arc routing instance when $n = 10$ and $m = 3$. Therefore, the rank-based representation of this chromosome would be $R = [1, 2, 2, 1, 2, 1, 4, 3, 0, 3]$. A distance between two individuals P_1, P_2 with rank-based representations R_1, R_2 can be calculated as follows:

$$\delta(P_1, P_2) = \frac{1}{n} \sum_{i=1}^{N_{\text{req}}} \frac{|R_1[i] - R_2[i]|}{M_r},$$

where M_i is the maximum rank value in R_1 and R_2. The degree of dissimilarity between two representations is assessed on a scale ranging from zero to one, with values approaching zero indicating similarity and values nearing one indicating dissimilarity. To measure diversity, we compute a quantity called the diversity contribution $\Delta(P)$ by considering the average distance between an individual, denoted as P, and its two closest neighbors. To determine the fitness of each individual, we employ the following formula:

$$\texttt{fitness}(P) = \texttt{objective}(P) \times \gamma^{\Delta(P)},$$

where $\texttt{objective}(P)$ is the value of objective function for the given individual and γ represents the diversification factor, which is a hyper parameter for the algorithm. When solving a maximization problem, γ should be greater than one; otherwise, it should be lower than one.

4 Computational results

To evaluate the performance of our proposed framework, we selected two different problems, the min-max windy k-vehicle rural postman problem (MM K-WRPP) and the undirected capacitated arc routing problem with profits. Each problem is analyzed using a benchmark set and baseline algorithms derived from the literature. We developed the algorithm using the Julia programming language on a Mac computer with 16 GB of RAM and an Apple M1 processor. The parameters used in our MA are $\mu = 10$, $\lambda = 20$, $k_{\text{TOURNAMENT}} = 2$, $It_{\text{STOP}} = 5000$, $It_{\text{DIV}} = 1000$, $n_{\text{BEST}} = 0.2\mu$, $\gamma = 0.9$ (1.2) for minimization (maximization) problems. Considering evolutionary algorithms are stochastic optimizations, the results of each run may differ. As a result, we solve each instance ten times using MA and report the best and average results obtained from the ten runs.

4.1 Min-max windy k-vehicle rural postman problem

There is an important arc routing problem known as the windy rural postman problem. In WRPP, $G = (\mathcal{V}, \mathcal{E})$ is an undirected graph and $E_R \in \mathcal{E}$ is the set of all required edges. Every edge in E_R has two travel times (costs), one for traversing it in each

direction. i.e., for the edge e_{ij}, the travel times are t_{ij} and t_{ji}, where $t_{ij} \neq t_{ji}$. The objective is to find a tour in which the vehicle starts from the depot, visits all the edges in E_R, and returns to the depot with the least amount of travel time (cost). Here, we solve the WRPP using multiple vehicles with the objective of minimizing the longest tour length. We set C and L to ∞ since there is no constraint on vehicle capacity or tour length.

A benchmark data set derived from 24 RPP instances proposed by Christofides is used to test our algorithm on MM K-WRPP. Within this set, six different WRPP instances are generated for each 24 RPP instances. This benchmark set contains 144 instances, which are available at http://www.uv.es/corberan. Among the benchmark instances in this set, there are those with up to 50 vertices, 184 edges, and 78 required edges. Several algorithms have been proposed in the literature for solving this set. Examples of existing algorithms in the literature include a branch-and-cut algorithm by Benavent et al. [22], an Iterative Local Search (ILS) algorithm by Benavent et al. [27], and a branch-price-and-cut algorithm by Corberán et al. [19]. The results of our study are compared with those of ILS. The ILS algorithm has been tested in four different settings. To conduct the comparison, we have selected the setting with the best performance, which is referred to as ILS-H1.

Table 1 provides a summary of the results. Due to the speed of our processor, the computational time cannot be used as a reliable basis for comparison. For the purposes of comparing the quality of solutions, we take the best-known solutions as a baseline and calculate the gap for each algorithm. In each 24 groups of instances, six instances are included. Column "Opt." represents the number of instances for which the optimal solution is known. The presented gaps are the gaps between these results of algorithms (`Alg`) and the best known solutions (`BKS`). Following is the formula for calculating the gaps:

$$\text{gap}_{\texttt{Alg}} = \frac{\texttt{Alg} - \texttt{BKS}}{\texttt{BKS}} \times 100\%$$

Therefore, the negative gaps refer to the instances that MA has improved the `BKS`. For each algorithm, the gap and time are averaged over all six instances within the group. For MA, gap^b and gap^a represent the gaps for best and average results out of ten runs respectively. Average 1 indicates the average over all groups of instances, whereas Average 2 is over groups C20-C24 in order to emphasize the performance over large instances. The results show that ILS-H1 performs better than MA when there are two vehicles involved. There is a similar performance between the algorithms when there are three vehicles. However, in the case of four and five vehicles, MA is superior to ILS-H1. Having used an exact method for splitting and a heuristic method for sequence, we conclude that the exact part of our algorithm is more active with more tours to split, resulting in better performance. It is worth noting that MA has improved the best-known solution for $1, 13$, and 15 instances for the case of $3, 4$, and 5 vehicles, respectively.

4.2 Undirected capacitated arc routing problem with profits

As a second example of applying our proposed framework, we examine the undirected capacitated arc routing problem with profits (UCARPP). The graph $G(\mathcal{V}, \mathcal{E})$ in this

	Number of vehicles = 2						Number of vehicles = 3						Number of vehicles = 4						Number of vehicles = 5					
	Opt.	ILS-H1		MA			Opt.	ILS-H1		MA			Opt.	ILS-H1		MA			Opt.	ILS-H1		MA		
		gap	time	gap^b	gap^a	time		gap	time	gap^b	gap^a	time		gap	time	gap^b	gap^a	time		gap	time	gap^b	gap^a	time
C01	6	0.0	0.0	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.1
C02	6	0.0	0.2	0.0	0.0	0.2	6	0.0	0.2	0.0	0.0	0.2	6	0.0	0.2	0.0	0.0	0.2	6	0.0	0.2	0.0	0.0	0.2
C03	6	0.0	5.8	0.0	0.1	0.6	6	0.0	4.0	0.0	0.0	0.6	6	0.0	3.5	0.0	0.0	0.6	6	0.0	3.4	0.0	0.0	0.5
C04	6	0.1	4.0	0.0	0.1	0.5	6	0.0	2.7	0.0	0.1	0.5	6	0.0	2.6	0.0	0.0	0.5	5	0.3	2.6	0.0	0.0	0.4
C05	6	0.0	0.3	0.0	0.0	0.2	6	0.0	0.3	0.0	0.0	0.2	6	0.0	0.2	0.0	0.0	0.2	6	0.0	0.2	0.0	0.0	0.2
C06	6	0.0	2.3	0.0	0.1	0.4	6	0.0	1.9	0.0	0.0	0.4	6	0.0	1.7	0.0	0.0	0.4	5	1.2	1.7	0.0	0.0	0.3
C07	6	0.0	5.1	0.0	0.2	0.4	6	0.0	3.1	0.0	0.0	0.5	6	0.0	2.7	0.0	0.0	0.4	5	0.1	2.9	0.0	0.0	0.4
C08	6	0.0	6.6	0.0	0.1	0.6	6	0.0	5.0	0.0	0.0	0.6	5	0.0	4.7	0.0	0.1	0.6	4	1.5	4.5	0.0	0.0	0.5
C09	6	0.0	0.4	0.0	0.0	0.2	6	0.0	0.3	0.0	0.0	0.2	6	0.0	0.3	0.0	0.0	0.2	6	0.0	0.3	0.0	0.0	0.2
C10	6	0.0	0.1	0.0	0.0	0.1	6	0.0	0.1	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.1
C11	6	0.0	0.0	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.1
C12	6	0.0	0.0	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.1
C13	6	0.0	0.0	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.1	6	0.0	0.0	0.0	0.0	0.8
C14	6	0.0	18.3	0.0	0.1	1.0	6	0.0	10.5	0.0	0.5	1.0	5	0.1	9.0	0.0	0.1	1.1	4	0.9	8.4	0.1	0.1	1.0
C15	6	0.0	1.6	0.0	0.0	0.3	6	0.0	1.2	0.0	0.0	0.4	6	0.0	1.2	0.0	0.0	0.3	6	0.0	1.2	0.0	0.0	0.3
C16	6	0.1	17.8	0.0	0.5	1.1	6	0.1	10.2	0.0	0.3	1.2	3	2.1	8.8	0.1	0.4	1.1	1	4.0	8.4	0.1	0.2	1.1
C17	6	0.0	1.0	0.0	0.0	0.2	6	0.0	0.8	0.0	0.0	0.2	6	0.0	0.8	0.0	0.0	0.2	6	0.0	0.8	0.0	0.0	0.2
C18	6	0.0	1.2	0.0	0.0	0.3	6	0.0	1.1	0.0	0.0	0.3	6	0.0	1.0	0.0	0.0	0.2	6	0.0	1.0	0.0	0.0	0.2
C19	6	0.0	11.2	0.0	0.0	0.7	6	0.0	6.3	0.0	0.3	0.8	6	0.0	5.2	0.0	0.2	0.7	4	0.4	4.8	0.0	0.0	0.6
C20	6	0.3	175.0	0.3	0.9	4.2	4	0.8	79.4	0.3	0.8	4.7	1	1.3	56.7	-0.2	0.7	4.6	0	3.2	48.9	-0.3	0.4	4.7
C21	6	0.9	214.8	0.9	2.6	5.7	2	1.6	97.5	0.4	2.2	5.8	0	2.4	68.7	-0.3	0.9	6.0	0	5.4	58.2	-0.2	0.7	5.2
C22	6	0.3	351.6	0.6	1.6	9.2	0	1.5	147.1	0.2	1.7	8.9	0	3.9	95.4	-0.6	1.0	8.5	0	7.3	77.7	-0.3	0.6	8.8
C23	6	0.5	404.0	0.8	1.7	8.3	1	2.1	162.5	0.5	1.6	8.2	0	4.9	105.9	0.1	1.1	7.6	0	7.3	85.8	-0.9	0.4	7.5
C24	6	0.2	128.0	0.3	1.2	4.0	3	0.7	59.9	0.1	0.7	3.9	0	3.7	43.8	0.2	0.9	3.7	0	3.8	38.2	0.2	0.8	3.7
Average 1	0.1	56.2		0.1	0.4	1.6		0.3	24.7	0.1	0.3	1.6		0.8	17.2	0.0	0.2	1.6		1.5	14.6	-0.1	0.1	1.55
Average 2	0.4	254.7		0.6	1.6	6.3		1.3	109.3	0.3	1.4	6.3		3.3	74.1	-0.1	0.9	6.1		5.4	61.7	-0.3	0.6	5.97

Table 1: Comparison of MA and ILS-H1 in solving MM K-WRPP.

			VNS		Tabu A		MPA			ABC			MA		
	Ins.	size	slow	fast	slow	fast	gap^b	gap^a	time	gap^b	gap^a	time	gap^b	gap^a	time
Number of vehicles=2	val1	24-39	0.19	0.00	0.19	0.38	0.00	0.02	9.60	0.37	1.17	3.96	0.19	0.44	2.31
	val2	24-34	0.00	0.00	0.00	0.00	0.00	0.02	9.23	0.00	0.17	2.94	0.00	0.00	1.63
	val3	24-35	0.00	0.27	0.36	0.00	0.00	0.00	6.30	0.00	0.50	3.15	0.00	0.14	1.82
	val4	41-69	1.69	1.09	1.24	0.28	-0.93	-0.45	41.05	-0.76	1.47	14.69	-0.57	0.03	5.42
	val5	34-65	1.20	2.15	0.59	0.82	-0.38	0.12	53.48	-0.38	1.35	12.74	-0.38	0.34	3.85
	val6	31-50	0.00	0.19	1.73	0.91	0.00	0.60	79.87	0.00	1.18	6.56	-0.26	-0.05	2.52
	val7	40-66	0.00	0.60	1.26	2.47	-0.47	-0.18	167.73	-0.47	1.74	13.50	-0.58	0.66	3.87
	val8	30-63	1.19	0.94	0.60	1.03	0.00	0.84	316.60	0.35	2.31	11.85	-0.02	0.32	3.95
	val9	50-92	0.76	2.67	2.00	1.41	-0.47	0.29	198.88	-0.54	1.44	33.19	-1.08	-0.46	8.02
	val10	50-97	1.02	1.64	2.73	1.36	-0.91	-0.08	243.45	-1.35	0.79	37.78	-1.64	-1.05	11.57
	Average		0.61	0.95	1.07	0.87	-0.32	0.12	112.62	-0.28	1.21	14.04	-0.43	0.04	4.50
Number of vehicles=3	val1	24-39	0.00	0.15	0.41	0.42	0.00	0.00	32.90	0.00	0.92	4.41	0.00	0.09	2.91
	val2	24-34	0.00	0.00	0.16	0.16	0.00	0.00	9.57	0.00	0.18	3.36	0.00	0.00	2.07
	val3	24-35	0.00	0.00	0.00	0.00	0.00	0.00	7.47	0.00	0.10	3.64	0.00	0.00	2.33
	val4	41-69	0.47	1.16	0.79	1.00	-0.63	-0.05	26.30	0.23	1.51	16.65	-0.37	0.48	8.33
	val5	34-65	1.20	1.17	0.76	1.45	-0.66	0.08	171.55	-0.71	1.10	14.80	-0.84	-0.21	6.59
	val6	31-50	0.05	1.09	0.46	1.09	-0.05	0.31	59.67	-0.05	1.50	7.79	-0.24	0.18	3.99
	val7	40-66	0.09	1.63	1.78	2.91	-0.47	0.16	82.10	-0.47	1.42	15.68	-0.47	-0.07	6.70
	val8	30-63	0.58	1.08	1.20	1.22	-0.52	0.19	67.90	0.32	0.99	13.75	-0.47	0.04	6.29
	val9	50-92	0.29	3.11	0.51	1.54	-1.00	-0.24	357.03	-1.00	0.77	37.90	-1.68	-1.06	14.00
	val10	50-97	0.63	1.75	0.76	1.22	-1.01	-0.28	183.93	-1.27	-0.14	43.29	-2.35	-1.67	20.38
	Average		0.33	1.11	0.68	1.10	-0.43	0.02	99.84	-0.30	0.83	16.13	-0.64	-0.22	7.36
Number of vehicles=4	val1	24-39	0.00	0.20	0.41	0.41	0.00	0.00	6.77	0.41	1.13	5.11	0.10	0.17	3.71
	val2	24-34	0.00	0.00	0.00	0.00	0.00	0.00	11.10	0.00	0.00	3.81	0.00	0.00	2.44
	val3	24-35	0.00	0.00	0.00	0.00	0.00	0.00	7.10	0.00	0.00	4.08	0.00	0.00	2.79
	val4	41-69	0.32	0.57	0.42	0.11	-0.56	-0.15	83.90	-0.56	0.78	17.86	-0.81	-0.33	9.35
	val5	34-65	0.18	0.86	1.47	1.53	-0.95	-0.19	116.37	-0.99	0.67	15.35	-1.03	-0.56	8.89
	val6	31-50	0.00	0.50	0.60	0.96	0.00	0.31	43.43	0.00	0.80	8.57	0.00	0.03	4.37
	val7	40-66	0.06	0.86	0.68	2.03	-0.12	0.51	119.53	-0.12	0.76	17.15	-0.12	0.09	7.51
	val8	30-63	0.13	0.96	0.47	1.41	-0.03	0.29	126.47	0.51	1.11	14.85	-0.22	0.03	8.11
	val9	50-92	0.75	2.08	0.27	1.52	-0.68	-0.15	350.98	-0.41	0.94	40.26	-0.79	-0.37	17.18
	val10	50-97	0.00	1.85	2.38	1.67	-0.45	0.04	316.38	-0.53	0.96	47.18	-0.67	-0.31	22.54
	Average		0.14	0.79	0.67	0.96	-0.28	0.06	118.20	-0.17	0.72	17.42	-0.36	-0.12	8.69

Table 2: Comparison of MA with baseline algorithms in solving UCARPP.

problem is an undirected graph, with $\mathcal{V}$ representing all the vertex points and $\mathcal{E}$ representing all the edges. Rather than required edges, we have profitable edges represented by $E_p \in \mathcal{E}$, with different profits for each edge. There is also a demand associated with each edge in E_p. Each homogeneous vehicle has a capacity of C. Unlike WRPP, UCARPP has symmetric travel times. Each edge has only one travel time regardless of its direction ($t_{ij} = t_{ji}$). Furthermore, there is a constraint limiting the total travel time for each tour by L in addition to the capacity constraint for vehicles.

An arc routing benchmark set, originally developed by Belenguer and Benavent [28] and modified by Archetti et al. [15] for UCARPP, is used as the benchmark set. This dataset is available at http://users.ntua.gr/ezach/. x These instances are named in the format val$\{i\}\{s\}$, where i is an integer from $[1, 10]$ and s is a character from $[A, B, C, D]$. In each val$\{i\}$, the instances are identical graphs, with the only difference being the edge demands and profits. In each instance, the problem has been solved under the assumption of two, three, and four vehicles; therefore, there are 102 instances in total. Among the algorithms that have been proposed in the literature to solve this benchmark set are Variable Neighborhood Search (VNS) and Tabu Search (Tabu feasible and Tabu Admissible) by Archetti et al. [15], a Move Promise Algorithm (MPA) by Zachariadis and Kiranoudis [29], and an Artificial Bee Colony (ABC) by Cura [30].

An overview of the results can be found in Table 2. The results are presented as averages for the instances of each group. The group val1, for example, contains three instances with the names val1A, val1B, and val1C. Some groups contain three instances while others contain four instances. In the second column of the table, the name of the group of instances is displayed. The third column displays the graph size as $|V| - |E|$, where $|V|$ is the number of vertices and $|E|$, the number of profitable edges.

We have taken the best results out of VNS, Tabu feasible, and Tabu admissible as the baseline and calculated the gap between these results and the baseline for all algorithms. Following is the formula for calculating the gaps:

$$\text{gap}_{\texttt{Alg}} = \frac{\texttt{baseline} - \texttt{Alg}}{\texttt{baseline}} \times 100\%$$

A negative gap value indicates that the result was better than the baseline. Algorithms used for comparison include VNS, Tabu admissible (Tabu A), MPA, and ABC. VNS and Tabu A have been implemented wit two variants, slow and fast. MPA, ABC, and our MA were all run ten times for each instance. gap^b and gap^a represent the gap between best and average results and baseline, respectively. Similarly to the previous section, the comparison cannot be made based on the computation times since the processing speeds are different. Nevertheless, the solution quality obtained by MA in all three cases is clearly superior. Furthermore, it is noteworthy that MA has improved `BKS` in 30 out of 102 instances.

5 Conclusions

The purpose of this paper was to present a generic algorithmic framework that can be applied to a variety of arc routing problems. This category of problems includes those in which a set of edges should be visited at most once. To solve the problems,

we used genetic algorithms in conjunction with dynamic programming. The genetic algorithm is responsible for determining the order and direction of arcs, while the dynamic programming is responsible for assigning edges to different vehicles. In order to facilitate algorithm convergence, local searches are used. A systematic guideline is provided for applying these methods. This will enable the reader to apply them to the arc routing problem of their choice. This study utilized Dijstra's algorithm to determine the shortest path between the nodes.

The min-max windy K-vehicle rural postman problem and the undirected capacitated arc routing problem were selected to test the effectiveness of our approach. Our proposed framework was implemented and compared with existing algorithms based on an established benchmark set for each problem. We show the efficacy of our proposed framework by showing that it can outperform existing algorithm. In addition, our proposed framework found new best solutions for 29 out of 144 instances and 30 out of 102 instances of these benchmark sets, respectively. We hope that this work will encourage researchers to develop generic approaches able to address a variety of problems instead of merely focusing on one problem. There may be a potential research direction in developing an algorithm that can be applied to a class of node routing problems.

Acknowledgements. This work was partially supported by the National Science Foundation via CMMI-2032458.

Conflict of interest. The authors declare that they have no conflict of interest.

References

[1] Assad, A.A., Golden, B.L.: Arc routing methods and applications. Handbooks in Operations Research and Management Science **8**, 375–483 (1995)

[2] Del Pia, A., Filippi, C.: A variable neighborhood descent algorithm for a real waste collection problem with mobile depots. International Transactions in Operational Research **13**(2), 125–141 (2006)

[3] Corberán, Á., Eglese, R., Hasle, G., Plana, I., Sanchis, J.M.: Arc routing problems: A review of the past, present, and future. Networks **77**(1), 88–115 (2021)

[4] De Maio, A., Laganà, D., Musmanno, R., Vocaturo, F.: Arc routing under uncertainty: Introduction and literature review. Computers & Operations Research **135**, 105442 (2021)

[5] Orloff, C.S.: A fundamental problem in vehicle routing. Networks **4**(1), 35–64 (1974)

[6] Golden, B.L., Wong, R.T.: Capacitated arc routing problems. Networks **11**(3), 305–315 (1981)

[7] Bartolini, E., Cordeau, J.-F., Laporte, G.: Improved lower bounds and exact algorithm for the capacitated arc routing problem. Mathematical Programming **137**, 409–452 (2013)

[8] Longo, H., De Aragao, M.P., Uchoa, E.: Solving capacitated arc routing problems using a transformation to the cvrp. Computers & Operations Research **33**(6), 1823–1837 (2006)

[9] Hertz, A., Laporte, G., Mittaz, M.: A tabu search heuristic for the capacitated arc routing problem. Operations research **48**(1), 129–135 (2000)

[10] Lacomme, P., Prins, C., Ramdane-Chérif, W.: A genetic algorithm for the capacitated arc routing problem and its extensions. In: Workshops on Applications of Evolutionary Computation, pp. 473–483 (2001). Springer

[11] Santos, L., Coutinho-Rodrigues, J., Current, J.R.: An improved ant colony optimization based algorithm for the capacitated arc routing problem. Transportation Research Part B: Methodological **44**(2), 246–266 (2010)

[12] Maniezzo, V., Roffilli, M.: Algorithms for large directed capacitated arc routing problem instances: Urban solid waste collection operational support. Recent advances in evolutionary computation for combinatorial optimization, 259–274 (2008)

[13] Belenguer, J.-M., Benavent, E., Lacomme, P., Prins, C.: Lower and upper bounds for the mixed capacitated arc routing problem. Computers & Operations Research **33**(12), 3363–3383 (2006)

[14] Baldacci, R., Maniezzo, V.: Exact methods based on node-routing formulations for undirected arc-routing problems. Networks **47**(1), 52–60 (2006)

[15] Archetti, C., Feillet, D., Hertz, A., Speranza, M.G.: The undirected capacitated arc routing problem with profits. Computers & Operations Research **37**(11), 1860–1869 (2010)

[16] Feillet, D., Dejax, P., Gendreau, M.: The profitable arc tour problem: Solution with a branch-and-price algorithm. Transportation Science **39**(4), 539–552 (2005)

[17] Euchi, J., Chabchoub, H.: Hybrid metaheuristics for the profitable arc tour problem. Journal of the Operational Research Society **62**(11), 2013–2022 (2011)

[18] Minieka, E.: The chinese postman problem for mixed networks. Management science **25**(7), 643–648 (1979)

[19] Corberán, Á., Plana, I., Rodríguez-Chía, A.M., Sanchis, J.M.: A branch-and-cut algorithm for the maximum benefit chinese postman problem. Mathematical Programming **141**, 21–48 (2013)

[20] Shafahi, A., Haghani, A.: Generalized maximum benefit multiple chinese postman problem. Transportation Research Part C: Emerging Technologies **55**, 261–272 (2015)

[21] Gulczynski, D.J., Heath, J.W., Price, C.C.: The close enough traveling salesman problem: A discussion of several heuristics. Perspectives in Operations Research: Papers in Honor of Saul Gass' 80 th Birthday, 271–283 (2006)

[22] Benavent, E., Corberán, A., Plana, I., Sanchis, J.M.: Min-max k-vehicles windy rural postman problem. Networks: An International Journal **54**(4), 216–226 (2009)

[23] Archetti, C., Speranza, M.G., Corberán, Á., Sanchis, J.M., Plana, I.: The team orienteering arc routing problem. Transportation Science **48**(3), 442–457 (2014)

[24] Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1**(1), 269–271 (1956)

[25] Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & operations research **31**(12), 1985–2002 (2004)

[26] Mahmoudinazlou, S., Kwon, C.: A hybrid genetic algorithm for the min–max multiple traveling salesman problem. Computers & Operations Research **162**, 106455 (2024)

[27] Benavent, E., Corberán, Á., Sanchis, J.M.: A metaheuristic for the min–max windy rural postman problem with k vehicles. Computational Management Science **7**, 269–287 (2010)

[28] Belenguer, J.M., Benavent, E.: A cutting plane algorithm for the capacitated arc routing problem. Computers & Operations Research **30**(5), 705–728 (2003)

[29] Zachariadis, E.E., Kiranoudis, C.T.: Local search for the undirected capacitated arc routing problem with profits. European Journal of Operational Research **210**(2), 358–367 (2011)

[30] Cura, T.: An artificial bee colony approach for the undirected capacitated arc routing problem with profits. International Journal of Operational Research **17**(4), 483–508 (2013)