Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 2435002, 20 pages
https://doi.org/10.1155/2022/2435002

*Research Article*

# Hybrid Genetic Algorithms for the Asymmetric Distance-Constrained Vehicle Routing Problem

**Zakir Hussain Ahmed**,[1] **Asaad Shakir Hameed**,[2,3] **and Modhi Lafta Mutar**[2,4]

[1]*Department of Mathematics and Statistics, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia*
[2]*Department of Mathematics, General Directorate of Thi-Qar Education, Ministry of Education, Al-Haboubi str., Nasiriyah 64001, Iraq*
[3]*Data Mining and Optimization Group, Centre of Artificial Intelligence, Faculty of Information Science and Technology, National University of Malaysia, B. B. Bangi 43600, Selangor, Malaysia*
[4]*Department of Medical Instruments Engineering Techniques, Al-Turath University College, Mansour str., Baghdad 12013, Iraq*

Correspondence should be addressed to Zakir Hussain Ahmed; zaahmed@imamu.edu.sa

Received 22 March 2022; Revised 15 May 2022; Accepted 19 May 2022; Published 22 June 2022

Academic Editor: Shimin Wang



We aim to suggest a simple genetic algorithm (GA) and other four hybrid GAs (HGAs) for solving the asymmetric distance-constrained vehicle routing problem (ADVRP), a variant of vehicle routing problem (VRP). The VRP is a difficult NP-hard optimization problem that has numerous real-life applications. The VRP aims to find an optimal tour that has least total distance (or cost) to provide service to $n$ customers (or nodes or cities) utilizing $m$ vehicles so that every vehicle starts journey from and ends journey at a depot (headquarters) and visits every customer only once. The problem has many variations, and we consider the ADVRP for this study, where distance traveled by every vehicle must not exceed a predefined maximum distance. The proposed GA uses random initial population followed by sequential constructive crossover and swap mutation. The HGAs enhance the initial solution using 2-opt search method and incorporate a local search technique along with an immigration procedure to obtain effective solution to the ADVRP. Experiments have been conducted among the suggested GAs by solving several restricted and unrestricted ADVRP instances on asymmetric TSPLIB utilizing several vehicles. Our experiments claim that the suggested HGAs using local search methods are very effective. Finally, we reported a comparative study between our best HGA and a state-of-the-art algorithm on asymmetric capacitated VRP and found that our algorithm is better than the state-of-the-art algorithm for the instances.

## 1. Introduction

The vehicle routing problem (VRP) is very complicated traditional NP-hard combinatorial optimization problem (COP) that was presented by Dantzig and Ramser [1]. The problem determines minimum distance (cost or time) route for a vehicle set to serve a customer set. It has several real-life applications such as shipments delivery, transportation networks, and street cleaning. The VRP is a widely studied problem that has several variants such as the VRP with Backhauls (VRPB), the VRP with pickup and delivery (VRPPD), the split delivery VRP (SDVRP), the VRP with time window (VRPTW), and the multi-depot VRP (MDVRP) [2].

We consider another variant of the VRP, called distance-constrained VRP (DVRP) in which total distance toured by every vehicle in the tour is constrained by a predefined maximum distance. The problem determines minimum cost route for a vehicle set so that every customer is provided service once by exactly one vehicle, every vehicle starts journey and ends journey at the same depot, and the entire distance traveled by each vehicle must not exceed the predefined maximum distance. Methods that have been used to solve the DVRP as well as COPs are categorized as exact and heuristic methods [3]. Branch and bound, branch and price, branch and cut, and lexisearch are some examples of exact methods which obtain exact solutions [4]. However, these methods take lots of computational effort. However, heuristic

and metaheuristic methods obtain near exact solutions quickly, so they are normally used in large-scale DVRP instances. Metaheuristic methods are more advanced than the heuristic methods. Genetic algorithm (GA), simulated annealing (SA), differential evolution algorithm (DEA), tabu search (TS), artificial bee colony (ABC), ant colony optimization (ACO), and particle swarm optimization (PSO) are some examples of metaheuristic methods. They can obtain suitable solutions to various kinds of optimization problems in a realistic time [5]. Among them, GA is commonly applied to find effective solution to the COPs in a reasonable time.

GA is a popular metaheuristic algorithm, which was first introduced by John Holland [6]. The major assumption of GA is that just the stronger individuals/chromosomes can live longer. Normally, a random population of chromosomes is generated first, and then using possibly three operators—selection, crossover, and mutation, (hopefully) new population is created in each generation. The process is replicated till the stopping criterion is reached. The purpose is to find solution with higher fitness value that is close to the optimal solution.

A common problem with GAs is premature convergence to obtain optimal solution which is due to the population diversity loss. If it is low, the convergence will be fast; otherwise, convergence will be time-consuming and sometimes it is a wastage of computational efforts. So, it is important to balance between exploitation and exploration of search area. In general, the effectiveness of GAs extremely depends on genetic operators. Among them, crossover operator plays a very important role and accordingly many researchers used/developed different crossovers for the VRP. Usually, crossover techniques that were used/developed for the usual traveling salesman problem (TSP) are used in other COPs also. Among crossover operators, sequential constructive crossover (SCX) was found very good for some COPs [7, 8]. Though simple GA using SCX is very good, sometimes it gets stuck in local optima. So, one can go for hybrid GA that merges simple GA with a local search (or heuristic) method.

The main contribution of this paper is to propose a simple GA and four hybrid GAs (HGAs) for the ADVRP. In our proposed HGAs, initial population is generated randomly that is further enhanced by 2-opt local search, offspring are created by SCX, random alteration of two genes by swap mutation, solutions are improved by one of three different local search methods, and stagnation/premature convergence is removed by immigration method. Experiments have been conducted among the suggested GAs by solving several restricted and unrestricted ADVRP instances on asymmetric TSPLIB utilizing several vehicles. Our experiments claim that the suggested HGAs using different local search methods are very effective. Finally, we did a comparative study between our best HGA and a state-of-the-art algorithm [9] on some asymmetric capacitated VRP (ACVRP) and found that our algorithm is better than the competing algorithm for the instances.

This paper is arranged as follows: Section 2 defines the problem, Section 3 provides a literature survey for the problem, Section 4 develops the simple GA and hybrid GAs for the problem, Section 5 introduces results of experiments, and finally Section 6 introduces discussion and conclusion.

## 2. Problem Definition

The ADVRP determines the minimum cost route to serve a customer set. The cost is defined by total traveling distance. Customers are scattered across several locations, and each of them is to be visited only once by a single vehicle. Generally, the vehicles have the same distance constraints.

*2.1. Assumptions.* Following are the assumptions for defining the problem:

(i) Each customer is visited exactly once by exactly one vehicle

(ii) Each vehicle route starts and ends at the same depot

(iii) Each vehicle's route can only pass through one depot exactly once

(iv) A non-negative distance-constrained for all vehicle is defined, and the distance traveled by each vehicle cannot exceed the distance-constrained

(v) The sum of route of all vehicles must be minimum

*2.2. Notation.* Following is the list of notations that will be used in this study (Table 1).

The objective of the ADVRP is to find a least cost optimal tour set that visit all cities using all vehicles, every vehicle starts journey from and ends journey at the same headquarters, each city is visited exactly once, and the distance traveled by each vehicle must not exceed $D_{\max}$. If $d_{ij} = d_{ji}$, the matrix $D$ is symmetric, otherwise, asymmetric. The mathematical model of the ADVRP is given below [10].

The objective function:

$$f(S) = \min \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} d_{ij} x_{ij}. \tag{1}$$

Subject to

$$\sum_{i=0}^{n-1} x_{ij} = 1 \quad j = 1, 2, \ldots, n-1, \tag{2}$$

$$\sum_{j=0}^{n-1} x_{ij} = 1 \quad j = 1, 2, \ldots, n-1, \tag{3}$$

$$\sum_{j=0}^{n-1} x_{0j} = m, \tag{4}$$

$$\sum_{i=0}^{n-1} x_{i0} = m, \tag{5}$$

Table 1: List of notations that will be used in this study.

| Notation | Description |
|---|---|
| $n$ | Number of cities (or customers or nodes) |
| $N = \{0, 1, 2, \ldots, n\}$ | Set of cities, where "city 0" is the depot |
| $M$ | Number of vehicles |
| $D = [d_{ij}]$ | Distance matrix |
| $d_{ij}$ | The traveled distance from city $i$ to city $j$ |
| $D_{\max}$ | Maximum traveled distance-constrained |
| $x_{ij}$ | The decision binary variable that is equal to 1 if a vehicle travels from city $i$ to city $j$ and 0 otherwise |
| $z_{ij}$ | The variable that presents the shortest distance traveled from the depot to city $j$, where $i$ is the predecessor of $j$ |

$$\sum_{j=0, j\neq i}^{n-1} z_{ij} - \sum_{j=0, j\neq i}^{n-1} z_{ji} - \sum_{j=0}^{n-1} d_{ij} x_{ij} = 0 \quad i = 1, 2, \ldots, n-1, \tag{6}$$

$$z_{ij} \leq \left(D_{\max} - d_{j0}\right) x_{ij} \quad j \neq 0, \tag{7}$$

$$z_{ij} \leq (D_{\max}) x_{i0} \quad i = 1, 2, \ldots, n-1, \tag{8}$$

$$z_{ij} \geq \left(d_{ij} + d_{0i}\right) x_{ij} \quad i \neq 0, \tag{9}$$

$$z_{0i} = d_{0i} x_{0i} \quad i = 1, 2, \ldots, n-1, \tag{10}$$

$$x_{ij} \in \{0, 1\}. \tag{11}$$

In this formulation, the constraint (1) shows the objective function that minimizes the total routes' distance. The constraints (2) and (3) are the constraints that ensure that each node (or customer) is visited exactly once, whereas the constraints (4) and (5) ensure that only $m$ vehicles are allowed. The constraint (6) is a flow constraint that is identified as a flow-based model which states that the distance from *city i* to another *city j* on a tour must be same as the difference between the distance from headquarters (depot) to *city i* and the distance from headquarters to *city j*. The constraint (7) claims that the distance from headquarters to *city j* must not exceed the difference between the predefined maximum distance ($D_{\max}$) and the distance from *city j* to depot. The constraint (8) verifies that the distance traveled up to the depot must not exceed the predefined maximum distance. Additionally, the constraint (9) states that the total distance from depot to *city j* must not be less than the distance from the depot to *city i* plus the distance from *city i* to *city j*. The constraint (10) shows the initial value of $z_{0i}$ that equals the distance from the depot to *city i*. The constraint (11) states that the decision variables $x_{ij}$ are binary variables.

## 3. Literature Review

There is enough literature for the CVRP, but very few literature are available for the DVRP as it is not a common variant [2]. A branch and bound (B&B) method is developed in [10] for finding exact solution to the ADVRP. A multistart B&B method is developed in [11] for solving the ADVRP. Computational results show that the algorithm can provide exact solutions for some instances. But, for some instances, it could not find a feasible solution. Additionally, when distance restriction is tight, solving the problem instance becomes very hard, and the method is terminated before it might find any feasible solution. A lexisearch algorithm is developed in [4] for the DVRP and applied on various problem instance types. The results show that as the number of vehicles increases the computational time and optimal solution value also increase. Further, for some instances, the algorithm failed to prove the optimality of the solutions within restricted time limit. In general, exact algorithms cannot provide exact solutions for large problem instances, and hence many heuristic algorithms are developed for solving large problem instances.

Rachid et al. [12] compared some crossover operators for the VRP and found that partially mapped crossover (PMX) is better than ordered crossover (OX), and OX is better than merge #2. The PMX arbitrarily chooses two crossover points, copies the sub-chromosome between the points from any parent into one offspring, and then creates the full offspring by adding remaining cities from other parent in the mapped process. The OX arbitrarily chooses two crossover points, copies the sub-chromosome between the points from any parent into one offspring, and then creates the full offspring by adding remaining cities from other parent in the same order as they appear therein. The merge #2 operator is based on the global precedence among the genes and is independent of any of the chromosomes.

Krunoslav and Robert [13] compared eight crossover operators for the VRP and showed that alternate edge crossover (AEX) is best among them. The AEX chooses edges subsequently from the parents or arbitrarily chooses a legal edge if an illegal edge exists, for creating offspring.

Alabdulkareem and Ahmed [7] conducted a comparative study among four crossover methods—cycle crossover (CX), SCX, AEX, and PMX, for the DVRP and observed that SCX is the best. The CX takes positions and values from any parent so that the cities are reproduced from every parent in alternative cycles for creating offspring. The SCX creates an offspring using better links (edges) from the parents. Sometimes, it introduces better new edges which are not consistent in any parent. So, the chance of creating better offspring is very high [8].

Simple heuristic procedures have some drawbacks, such as stagnation and premature convergence. Hybrid techniques are used to overcome such drawbacks. Hybridization

can be done by combining the better sides of various exact methods or heuristic methods [14]. Several hybridization methods have been described in the literature for the VRP.

A hybrid swarm-based method (PSO-VNS) is proposed for the distance-constrained CVRP in [15], by combining a variable neighborhood search (VNS) within the PSO. As reported, the algorithm shows high-quality solutions compared to the existing algorithms.

The variable neighborhood SA (VNSA) algorithm is proposed for the CVRP in [16] by combining a modified VNS and SA. The algorithm is tested on 39 CVRP instances and then is compared against some existing algorithms. As reported, the algorithm could solve some large and very large instances efficiently.

A hybrid algorithm (LNS-ACO) is proposed for the capacitated VRP (CVRP) in [5] by embedding the solution by the ACO into the large neighborhood search (LNS) algorithm. The performance of the algorithm is tested on 88 CVRP instances and then is compared against other LNS algorithms. As reported, the algorithm has a suitable performance in solving the instances.

Four hybrid algorithms—improved intelligent water drops (IIWD), advanced cuckoo search (ACS), local search hybrid algorithm (LSHA), and post-optimization hybrid algorithm (POHA)—are proposed for the CVRP in [17]. Experimental results on some instances are compared to the best known solutions and found that LSHA and POHA algorithms could obtain best known solutions for most of the instances.

An enhanced perturbation-based VNS with adaptive selection mechanism method (PVNS-ASM) is developed in [18] by combining perturbation-based VNS (PVNS) with an adaptive selection mechanism (ASM). The algorithm is tested on 21 CVRP instances and then is compared against existing heuristics. The computational results show the efficiency of the algorithm.

A hybrid firefly algorithm (CVRP-FA) is proposed for the CVRP in [19] by integrating 2 h-opt and improved 2-opt algorithms for improving solution quality obtained by PMX and two mutation operators, and then tested on 82 instances. The computational results show that the algorithm has faster convergence rate and higher computational accuracy.

An improved SA (ISA) algorithm with crossover operator (ISA-CO) is developed for the CVRP in [20] where a population-based SA algorithm is applied. Further, the solutions are improved using four local search methods—swap, scramble, insertion, and reversion—and two crossover operators—PMX and OX operators. The algorithm was applied on 91 instances. The computational results show that the algorithm has a better performance compared to other algorithms.

A hybrid algorithm that combines the randomized VNS (RVNS) and TS is proposed in [9] to solve the ADVRP. In addition, the intensification and diversification stages are also incorporated to find optimal solutions. Computational results show that the algorithm is competitive in finding quality solutions.

There is some literature available for other VRP variants. A hybrid GA is proposed to solve the VRP with drones (VRPD) [21]. Experiments were carried out on different instances and found good performance of the algorithm. A novel hybrid algorithm by combining the GA and modified VNS (MVNS) for the VRP with cross-docking (VRPCD) is proposed in [22]. To prove the usefulness of the hybrid algorithm, a comparative study is carried out on some problem instances. It is found from the computational study that the proposed algorithm is more efficient than other algorithms to find best solutions in less computational time. A hybrid multi-objective genetic local search (HGLS) algorithm is proposed for the prize-collecting VRP (PCVRP) in [23]. Experiments on some instances are performed to evaluate the performance of the algorithm that shows the superiority of the algorithm.

## 4. The HGAs for the ADVRP

In this present section, a simple GA and four HGAs are proposed for the ADVRP. Following is the list of notations that will be used in our algorithms (Table 2).

*4.1. The Solution Encoding and Initial Population.* For applying GA to solve any problem, a way to represent (encode) a solution as chromosome (individual) must be defined first. In our GAs, a solution is encoded by an integer chromosome called path representation whose length is $n + m\text{-}1$, where $n$ is the number of cities and $m$ is the number of vehicles. In this representation, there are $m\text{-}1$ extra cities that represent duplicate depot cities to show the beginning of new vehicles [24]. A chromosome consisting of all routes of the vehicles is created randomly such that distance constraint is not violated. An initial population of size $P_s$ is created using Algorithm 1.

An example of a chromosome with $n = 10$ cities and $m = 3$ vehicles is given in Figure 1(a), where the integer 1 and integers bigger than 10 are the depot and the others are intermediate cities. The routes of the vehicles are shown in the VRP version in Figure 1(b), while the graphical interpretation of the routes is given in Figure 1(c). Thus, the given distance matrix is to be augmented to show the duplicate depot cities. For this, $m\text{-}1$ copy of the depot (city 1) row and column (i.e., 1st row and 1st column) is added to the given original matrix.

*4.2. Fitness Function and Selection Operator.* The objective function value of a chromosome (solution) is the total traveled distance of the routes by all vehicles. The distance of every route is computed by adding the distances between the cities. Since the ADVRP is minimization problem, so the fitness function is the inverted objective function. In the selection procedure, a subpopulation (some chromosomes) is chosen from the current population for forming the next population. The performance of GA is affected by choosing a better selection operator without which GA is a like random sampling that gives various results in the generations. Several selection procedures are present in the literature. We implement the fitness proportional selection (FPS) [25] for our GAs, which is very popular operator where the fitness

Table 2: List of notations that will be used in our algorithms.

| Notation | Description |
|---|---|
| $P_s$ | Population size |
| $P_c$ | Crossover probability |
| $P_{\text{mut}}$ | Mutation probability |
| MaxGen | Maximum generation allowed |
| $G_i$ | Population in $i$th generation |
| $f_i$ | Fitness of $i$th chromosome |
| $\text{prob}_i$ | Selection probability of $i$th chromosome in a generation |
| $cp_i$ | Cumulative probability of $i$th chromosome in a generation |
| $C_j$ | The $j$th chromosome |
| $B_i$ | Best solution in $i$th generation |
| $D_{\text{route}}$ | Distance of the route of a vehicle |
| BS | Best solution |
| BT | Best tour |

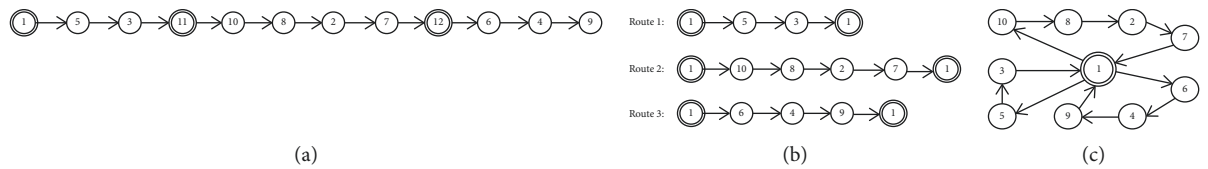


Figure 1: (a) An example chromosome, (b) the VRP routes, and (c) the graphical interpretation.

**Input**: $n$, $D_{\max}$, $P_s$.
**Output**: Population of chromosomes.
**for** $i = 1$ **to** $P_s$ **do**
  **Set** first city $p = 1$.
  Current chromosome contains only "city 1."
  Prepare a list of remaining cities except dummy cities.
  **for** $j = 2$ **to** $n$ **do**
    Select a city (suppose city $q$) randomly from the list of remaining cities.
    If the distance of the route of present vehicle is less than or equal to $D_{\max}$, then add it to the current chromosome and then remove it from the list to make sure that it is not repeated.
    If the distance of the route greater than $D_{\max}$ then add a dummy depot (city $q$).
    Rename the "city $q$" as "city $p$" and continue.
  **end for**
**end for**
Improve the population by 2-opt local search
**Return** the population

Algorithm 1: Initial population creation algorithm.

value of every chromosome in a population relates the area of roulette-wheel portions. Then, a chromosome is pointed by the roulette-wheel pointer after it is rotated. Depending on the fitness value of each chromosome, a probability $prob_i$ of selection is calculated as follows:

$$\text{prob}_i = \frac{f_i}{\sum_{j=1}^{P_s} f_j}; \quad i \in \{1, 2, \ldots, P_S\}, \tag{12}$$

where $P_s$ is the population size and $f_i$ is the fitness function value for the chromosome $i$. Thus, better fitness value chromosomes have higher chance of being selected as parents. There is no variation of the segment size and selection probability during the selection process. This process is very simple to implement, and it gives unbiased distributed probabilities to the chromosomes and assigns a high probability to the best chromosome. This procedure is called roulette-wheel selection procedure [6] that is presented in Algorithm 2.

*4.3. The Crossover Operator.* The selection procedure gives a trade-off between exploration and exploitation of search area. The crossover is a major procedure in GAs that is

**Input**: $P_s$, Population of chromosomes.
**Output**: New population of chromosomes.
Calculate the fitness $f_i$, probability $\text{prob}_i$, and then cumulative probability $\text{cp}_i$ of each chromosome ($1 \le i \le P_s$) of the population. Note that $cp_0 = 0$.
**for** $i = 1$ **to** $P_s$ **do**
  Generate a random number $r \in [0, 1]$.
  **if** ($cp_{j-1} < r \le cp_j$) (for any $j, 1 \le j \le P_s$) **then**
    Copy the chromosome $j$ to the population.
  **end if**
**end for**
**Return** the new population

Algorithm 2: Roulette-wheel selection algorithm.

Table 3: The given distance matrix.

| City | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 99999 | 2 | 11 | 10 | 8 | 7 | 6 |
| 2 | 6 | 99999 | 1 | 8 | 8 | 4 | 6 |
| 3 | 5 | 12 | 99999 | 11 | 8 | 12 | 3 |
| 4 | 11 | 9 | 10 | 99999 | 1 | 9 | 8 |
| 5 | 11 | 11 | 9 | 4 | 99999 | 2 | 10 |
| 6 | 12 | 8 | 5 | 2 | 11 | 99999 | 11 |
| 7 | 10 | 11 | 12 | 10 | 9 | 12 | 99999 |

employed on a chromosome pair to generate offspring(s) within a subspace restricted by the parents. Combinedly, selection and crossover operators are very strong operators that accelerate the convergence of GAs. The basic one-point or multi-point crossover operators do not work with respect to our encoding. The crossover operators which are valid for the TSP can be applied to the VRP and its variants. Several crossover operators are present in the literature for the TSP, and we are using the SCX; as it is observed to be one of the best crossovers for the DVRP [7], we apply this SCX with some modifications. The SCX algorithm is presented in Algorithm 3.

We demonstrate the SCX applying on a 7-city ($n = 7$) and 2-vehicle ($m = 2$) instance together with distance matrix given in Table 3. Further, suppose that maximum allowed distance is 60.

We modify the given distance matrix by combining one copy of the depot (city 1) row and column (i.e., 1st row and 1st column) to the matrix [14] that is provided in Table 4.

Let $P_1$: (1, 2, 4, 8, 3, 6, 5, 7) and $P_2$: (1, 3, 8, 5, 2, 7, 4, 6) be parent chromosomes. The objective function value of a chromosome is determined by summing the tour distances of all vehicles. The objective function value (total distance) of the 1st parent chromosome is 75 with the 1st and 2nd vehicle distances 54 and 21, respectively. The objective function value of the 2nd parent chromosome is 72 with the 1st and 2nd vehicle distances 56 and 16, respectively.

The calculation is begun from the city 1 (depot). After city 1, cities 2 in $P_1$ and 3 in $P_2$ are un-visited cities with distances $d_{12} = 2$ and $d_{13} = 11$. Since $d_{12} < d_{13}$, city 2 is combined that generates the offspring as (1, 2). Since $2 = D_{\text{route}} < D_{\max} = 60$, continue to build offspring. After city 2, cities 4 in $P_1$ and 7 in $P_2$ are legitimate cities with distances $d_{24} = 8$ and $d_{27} = 6$. Since $d_{27} < d_{24}$, city 7 is combined that generates the offspring as (1, 2, 7). Since $8 = D_{\text{route}} < D_{\max} = 60$, continue to build offspring. After city 7, city 4 is in $P_2$ with distances $d_{74} = 10$, but no city in $P_1$. So, for $P_1$, search from the starting and find legitimate city 4 with $d_{74} = 10$. Since both are same cities, city 4 is combined that generates the offspring as (1, 2, 7, 4). Since $18 = D_{\text{route}} < D_{\max} = 60$, continue to build offspring. After city 4, cities 8 in $P_1$ and 6 in $P_2$ are legitimate cities with distances $d_{48} = 11$ and $d_{46} = 9$. Since $d_{46} < d_{48}$, city 6 is combined that generates the offspring as (1, 2, 7, 4, 6). Since $27 = D_{\text{route}} < D_{\max} = 60$, continue to build offspring. After city 6, cities 5 are in $P_1$ with distances $d_{65} = 11$, but there is no city in $P_2$. So, for $P_2$, search from the starting and city 3 with $d_{63} = 5$ is found. Since $d_{63} < d_{65}$, city 3 is combined that generates the offspring as (1, 2, 7, 4, 6, 3). Since $32 = D_{\text{route}} < D_{\max} = 60$, continue to build offspring. After city 3, cities 5 in $P_1$ and 8 in $P_2$ are legitimate cities with distances $d_{35} = 8$ and $d_{38} = 5$. Since $d_{38} < d_{35}$, city 8 is combined that generates the offspring as (1, 2, 7, 4, 6, 3, 8). This completes route for the first vehicle whose distance is 37. Continue to build route for the next vehicle as well as the offspring. After city 8, the un-visited city 5 is in both parents, with distance $d_{85} = 8$. So, city 5 is added that produces the offspring as (1, 2, 7, 4, 6, 3, 8, 5). Since $8 = D_{\text{route}} < D_{\max} = 60$, continue to build offspring. However, this is the complete offspring chromosome, and so, we stop. The distance of the route of the 2nd vehicle is 19, and total distance of the offspring is 37 + 19 = 56 which is less than the distance of the parents. For this example, the SCX obtains an offspring that has value better than the values of both parent chromosomes. Figure 2(a) shows parent chromosomes ($P_1$ and $P_2$), Figure 2(b) shows the offspring chromosome (O),

**Input**: $D, P_c, D_{\max}$, Pair of parent chromosomes.
**Output**: Offspring chromosome.
Generate a random number $r \in [0, 1]$.
**if** $(r \leq P_c)$ then **do**
  **Set** $p = 1$.
  The offspring chromosome contains only "city 1."
  **for** $i = 2$ **to** $n$ **do**
    In each chromosome consider the first "legitimate" (un-visited) city existed after "city $p$."
    **if** no legitimate city is existed in a parent, **then**
      Examine from starting of the parent and choose the first legitimate city existed after "city $p$."
    **end if**
    Assume that "city $\alpha$" and "city $\beta$" are selected from 1st and 2nd parents, respectively.
    **if** $(d_{p\alpha} < d_{p\beta})$ then **do**
      Add "city $\alpha$" to the offspring chromosome.
    **else**
      Add "city $\beta$" to the offspring chromosome.
    **end if**
    **If** after combining the current city, $D_{\text{route}} > D_{\max}$
    **then**
      Drop the current city and add a dummy depot in the route as the end city of the route.
    **end if**
    Rename the present city as "city $p$" and continue.
  **end for**
**end if**
**Return** the offspring chromosome

Algorithm 3: Sequential constructive crossover algorithm.

Table 4: The modified distance matrix.

| City | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 9999 | 2 | 11 | 10 | 8 | 7 | 6 | 9999 |
| 2 | 6 | 9999 | 1 | 8 | 8 | 4 | 6 | 6 |
| 3 | 5 | 12 | 99999 | 11 | 8 | 12 | 3 | 5 |
| 4 | 11 | 9 | 10 | 9999 | 1 | 9 | 8 | 11 |
| 5 | 11 | 11 | 9 | 4 | 9999 | 2 | 10 | 11 |
| 6 | 12 | 8 | 5 | 2 | 11 | 9999 | 11 | 12 |
| 7 | 10 | 11 | 12 | 10 | 9 | 12 | 9999 | 10 |
| 8 | 9999 | 2 | 11 | 10 | 8 | 7 | 6 | 9999 |

Figure 2(c) shows ADVRP routes of the offspring, and Figure 2(d) shows the graphical interpretation of the offspring chromosome. In general, the crossover operator that maintains better attributes of parents in their offspring(s) is supposed to be better crossover, and SCX is supposed to be better in this respect. In Figure 2(b), six boldface edges are from either parent chromosome.

This SCX obtains only one offspring chromosome. The parent chromosomes are chosen based on the predefined crossover probability. If the offspring has better fitness value than the parent, the first parent is substituted by the offspring in the new population.

*4.4. Mutation Operator.* To diversify the population, mutation operator is applied with a prespecified probability. Generally, mutation probability is set very low compared to crossover probability. The exchange mutation that chooses randomly two places in a chromosome and exchanges their values, if neither of them is dummy depot, is applied here. The exchange mutation is presented in Algorithm 4.

For example, let the chromosome: (1, 2, 7, 4, **6**, 3, 8, **5**) with distance 56 be allowed for the mutation, and the 5th and 8th positions with their values are swapped. Then, the muted chromosome will be muted: (1, 2, 7, 4, **5**, 3, 8, **6**) with distance of 1st and 2nd vehicles 33 and 19, respectively, and with total distance equal to 33 + 19 = 52 which is less than the distance of the original chromosome. Figure 3 shows this mutation process. However, we do not see whether the value of muted chromosome is better than the original chromosome, we only see whether the distance constraint is valid, and if it is not valid, then the mutated chromosome is not accepted.

*4.5. Local Search Approach.* Local search approaches are used to hybridize the simple GA that improve the solution quality and convergence level of the simple GA. In this study, the local

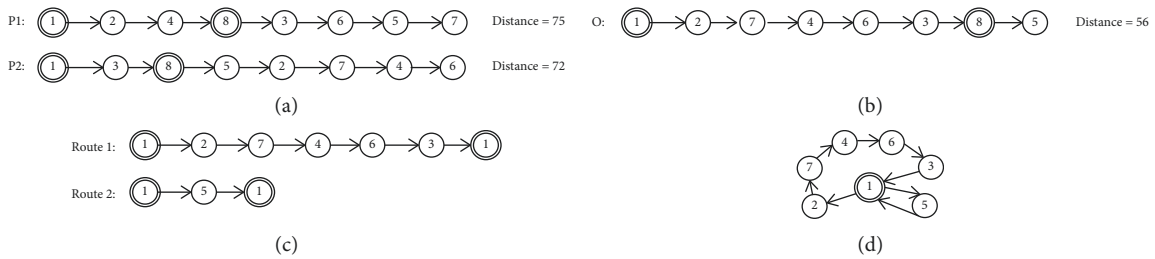


Figure 2: (a) Parent chromosomes, (b) the offspring, (c) the ADVRP routes of the offspring, and (d) the graphical interpretation of the offspring.

***Input*** *: A chromosome,* $P_{\text{mut}}$.
Output: Muted chromosome.
Generate a random number $r \in [0, 1]$.
**if** $(r \leq P_{\text{mut}})$ then **do**
*Select randomly two different cities except dummy depots, suppose "city α" and "city β" in the chromosome.*
"city $\alpha$" $\leftrightarrow$ "city $\beta$," provided that they do not violate the distance constraint.
**end if**
Return the mutated chromosome

Algorithm 4: Exchange mutation algorithm.

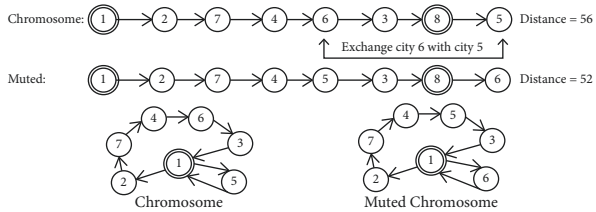


Figure 3: Exchange mutation process.

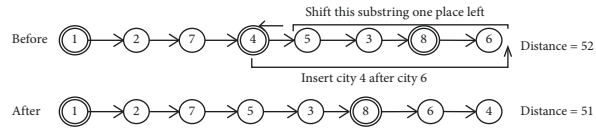


Figure 4: Implementation of the insertion search.

search approaches based on swap, insertion, and inversion mutations are used. Swap search chooses two cities (genes) randomly and swaps them. Insertion search inserts a randomly chosen city into a position in a chromosome randomly. Inversion search inverts the sub-chromosome between two randomly chosen places in a chromosome. Let ($\alpha 1$, $\alpha 2$, $\alpha 3$, ..., $\alpha n$) be a chromosome, then we define these three mutations as local search techniques in our HGAs as follows.

*4.5.1. Insertion Search.* The insertion search is presented in Algorithm 5. Figure 4 shows the implementation of the insertion search approach.

*4.5.2. Inversion Search.* The inversion search is presented in Algorithm 6. Figure 5 shows the implementation of the inversion search approach.

*4.5.3. Swap Search.* The swap search is presented in Algorithm 7. Figure 6 shows the implementation of the swap search approach.

In the proposed local search technique, one of these three local searches is chosen for the first three HGAs. For the fourth HGA, we choose any of the above local search approaches randomly with a probability of 1/3.

*4.6. Immigration Method.* Although GAs are robust approaches, however, occasionally they get trapped in local optima. It might be caused by similar population, and so, the population should be diversified to escape from the local optima. The immigrant procedure increases population diversity by substituting some chromosomes of the current population with newly generated chromosomes every generation. We use the following immigration procedure. If there is no improvement of solution within last 10% generations of maximum predefined generations, then 10% of population is replaced by random chromosomes which is further improved by 2-opt local search approach.

*4.7. The Algorithms.* We propose one simple GA and four HGAs for the ADVRP. The GA begins with randomly generated initial population and goes repeatedly through roulette-wheel selection, sequential constructive crossover, and exchange mutation procedures to enhance the population gradually, until a predefined maximum number of generations is reached, hoping that a near-optimal solution is obtained. In addition to the operators in GA, one of the

**Input**: A chromosome.
**Output**: New chromosome.
**for** $i = 2$ **to** $n - 1$ **do**
  **for** $j = i + 1$ **to** $n$ **d**
    If inserting city $\alpha i$ after city $\alpha j$ reduces the distance of the chromosome and does not violate distance constraint, then insert the city $\alpha i$ after the city $\alpha j$.
  end for
**end for**
**Return** the new chromosome

Algorithm 5: Insertion search algorithm.

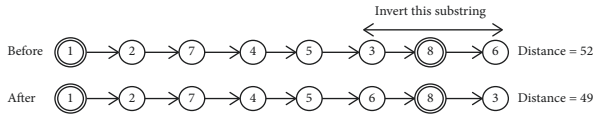


Figure 5: Implementation of the inversion search.

following local search approaches and above defined immigration approach are incorporated in the HGAs.

GA-INS : GA + insertion search + immigration approach.

GA-INV : GA + inversion search + immigration approach.

GA-SWP : GA + swap search + immigration approach.

GA-ADP : GA + adaptive search that randomly selects one of three local searches—insertion, inversion, and swap search + immigration approach.

The algorithm of the proposed HGAs is presented in Algorithm 8.

## 5. Experimental Results

The proposed GA and HGAs are encoded in Visual *C*++ and run on a Laptop with i7-1065G7 CPU@1.30 GHz and 8 GB RAM under MS Windows 10. The proposed GAs are executed for different parameter settings on some TSPLIB instances [26]. For setting parameters, ftv70 with 2 vehicles and infinite maximum distance constraint are used for the pilot runs. As the higher crossover probability can produce (hopefully) better solutions, we kept crossover probability fixed at 1.00 and run all algorithms for all combinations of $P_s = 20$, 30, 40, 50, 60, 70, 80, 90, and 100 and $P_{\text{mut}} = 0.05$, 0.06, 0.07, 0.08, 0.09, 0.10, 0.11, 0.12, 0.13, 0.14, and 0.15. We observed that for $P_s = 50$ and $P_{\text{mut}} = 0.10$, almost all algorithms could obtain better solutions; hence, these values are considered for the study. However, looking at the computational time and solution improvement in the successive generations, for termination condition, we considered 2000 generations for GA and 200 generations for HGAs. The parameter values are reported in Table 5.

We compare the performance of GA and four HGAs on asymmetric TSPLIB instances of various sizes with various numbers of vehicles.

In Figure 7, each GA is represented by a curve that indicates improvement of the solution in successive generations. The curve for simple GA shows that it starts the search process with the worst solutions compared to the HGAs at the initial stage. It shows variation in solutions within first 25 generations, and after that it shows no variation. So, it gets stuck in local minimum quickly and is found to be the worst one. Among HGAs, the curve for GA-INV shows that it starts the search process with the worst solutions at the initial stage, and shows variation in solutions within only first 10 generations. So, it gets stuck in local minimum very quickly and is found to be the worst one among HGAs. However, compared to simple GA, it is far better. The curve for GA-INS shows that it starts the search process with the best solutions compared to other HGAs at the initial stage and shows variation in solutions within first 30 generations. However, after 30 generations, it shows no variation. So, it gets stuck in local minimum quickly and is not the best one. The curves for GA-SWP and GA-ADP show that they start the search process with better solutions and are competing within first few generations. However, GA-SWP shows no variation in solutions after first 20 generations. The variation of solutions by GA-ADP continues up to 35 out of 50 generations, and it obtains best solution. So, GA-ADP is positioned in 1st position and GA-SWP is positioned in 2nd position.

We report relative studies among GA and HGAs on fifteen asymmetric TSPLIB instances of various sizes with 2 and 3 vehicles. Note that we suppose br17 with 2 vehicles is one instance and br17 with 3 vehicles is another instance. So, the total number of tested instances is thirty. The descriptions of the different column titles are as follows (Table 6).

Table 7 reports the results for 30 unrestricted ADVRP instances where $D_{\max}$ = Inf (infinity). The formula for AI is as follows:

$$\text{AI} = \frac{100\left(\text{AS}_1 - \text{AS}_2\right)}{\text{AS}_2}, \tag{13}$$

where $\text{AS}_1$ and $\text{AS}_2$ are average solutions found by the GA and a HGA, respectively.

The results are evaluated based on average solution, and SD and average improvement (%) of the HGAs over simple GA. From Table 7, it is noticed that all algorithms could find best average solutions for the instance br17 with both 2 and 3 vehicles. The algorithms GA-INS, GA-INV, GA-SWP, and GA-ADP could obtain best average solutions for 6, 5, 10, and

**Input**: A chromosome.
**Output**: New chromosome.
**for** $i = 2$ **to** $n - 1$ **do**
**for** $j = i + 1$ **to** $n$ **do**
If inverting substring between the cities $\alpha i$ and $\alpha j$ reduces the distance of the chromosome and does not violate distance constraint, then invert the substring
**end for**
**end for**
**Return** the new chromosome

Algorithm 6: Inversion search algorithm.

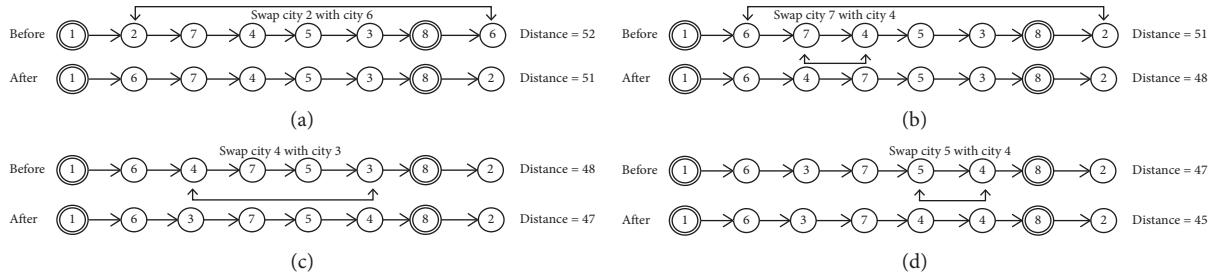


Figure 6: Implementation of the swap search.

**Input**: A chromosome.
**Output**: New chromosome.
**for** $i = 2$ **to** $n - 1$ **do**
**for** $j = i + 1$ **to** $n$ **do**
If swapping cities $\alpha$i and $\alpha$j reduces the distance of the chromosome and does not violate distance constraint, then swap them
**end for**
**end for**
**Return** the new chromosome

Algorithm 7: Swap search algorithm.

18 instances, respectively. On average, GA-ADP, GA-SWP, GA-INS, and GA-INV have average improvement (%) as 7.75, 7.51, 7.13, and 5.08, respectively. It shows that the average improvement of GA-ADP is the largest, GA-SWP is the second largest, GA-INS is the third largest, and GA-INV shows the smallest average improvement. From these results, we can tell that GA-ADP is the best one, GA-SWP is the second best, GA-INS is the third best, and GA-INV is positioned in fourth position. Further, by looking at SD, we can say that results by GA-ADP are stable because its obtained solutions have lowest SD. Figure 8 shows the average improvements (%) that also signifies the appropriateness of the HGAs, especially GA-ADP and GA-SWP. Note that b17.2 means the instance br17 with 2 vehicles. So, for these asymmetric unrestricted instances GA-ADP is the best method and GA-SWP is the second best method. Regarding the computational time, almost all HGAs are taking same time. However, simple GA takes less time. We further can see in this table that a number of vehicles have significant effect on the solution; i.e., as the number of vehicles increases, solution also increases.

From the above outcomes on the asymmetric unrestricted instances, we can see that HGAs have showed very good enhancements in the solutions over GA, and GA-ADP and GA are the best and worst algorithms, respectively. To confirm whether average solutions obtained by GA-ADP are statistically and significantly distinct from the average solutions found by other HGAs, we conducted Student's $t$-test applying the (14) below [27]. The $t$-test is utilized to measure not only improvement of an algorithm over another, but significant performance by the better algorithm.

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\left(\mathrm{SD}_1^2/n_1 - 1\right) + \left(\mathrm{SD}_2^2/n_2 - 1\right)}}, \tag{14}$$

where $\overline{X}_1$ is average of first sample, $\mathrm{SD}_1$ is standard deviation of first sample, $\overline{X}_2$ is average of second sample, $\mathrm{SD}_2$ is

**Input**: $n$, MaxGen.
**Output**: BS and BT
$G_0$ = Generate initial population using Algorithm 1
Evaluate ($G_0$)
BS = Find the best solution in this population
$i = 0$
**while** ($i \leq$ MaxGen) **do**
  $i = i + 1$
  $G_i$ = Population after selection using Algorithm 2
  **for** $j = 1$ **to** $P_s$ **do**
    $C_j$ = offspring chromosome using crossover Algorithm 3
    $C_j$ = mutated chromosome using mutation Algorithm 4
    $C_j$ = improved chromosome using a local search Algorithm 5, 6, or 7
  **end for**
  $G_i$ = New population
  Evaluate ($G_i$)
  $B_i$ = Find the best solution in this generation
  **if** ($B_i <$ BS) **then**
    BS = $B_i$
    BT = Best tour;
  **else if** (number of generation till last update $> 0.10 *$ MaxGen) **then**
      Apply immigration
  **end if**
  **end if**
**end while**
**Print** BS and BT

Algorithm 8: Hybrid genetic algorithm.

Table 5: Parameter settings for the GAs.

| Parameters | Values |
|---|---|
| $P_s$ | 50 |
| $P_c$ | 100% |
| $P_{\text{mut}}$ | 10% |
| Termination condition | For GA, 2000 generations<br>For HGAs, 200 generations |
| No. of runs for each instance | 20 times |

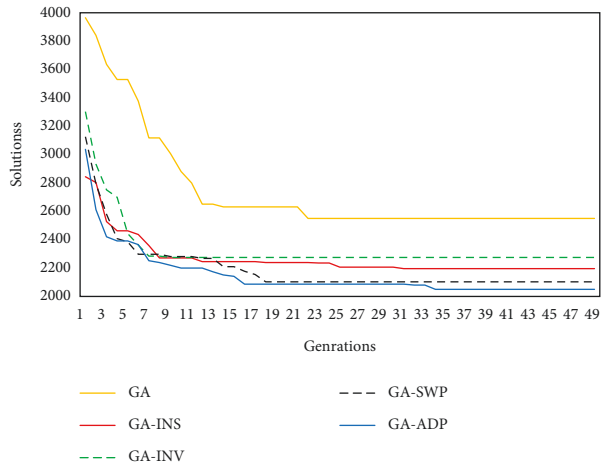


Figure 7: Solutions for ftv70 with 2 vehicles and infinite maximum distance constraint within only first 50 generations by GA and HGAs.

Table 6: Description of different notations used in the tables that contain results.

| Notation | Description |
|---|---|
| INST | Name of a TSPLIB instance |
| Opt | Optimal solution |
| AS | Average solution in 20 runs |
| SD | Standard deviation of obtained solutions |
| AT | Average computational time in seconds in 20 runs |
| AI | Average percentage of improvement of average solution obtained by a HGA over average solution obtained by simple GA |
| $\text{Max}_i$ | The maximum distance traveled by a vehicle among $m$ vehicles in a route |

standard deviation of second sample, $n_1$ is first sample size, and $n_2$ is second sample size.

Here, $\overline{X}_2$ and $\text{SD}_2$ are found by GA-ADP, and $\overline{X}_1$ and $\text{SD}_1$ are found by remaining HGAs. Table 8 reports t-statistic values, which can be positive or negative. As the problem is a minimization problem, positive value implies that GA-ADP found better solution than its rival HGA found, and negative value implies that the rival HGA found better solution than GA-ADP found. We applied 95% confidence level ($t_{0.05} = 1.73$), so if t-value is higher than 1.73, they have significant difference. So, if t-value is positive, then GA-ADP is significantly better; otherwise, its competitive HGA is better. If t-value is smaller than 1.73, then they have no statistical and significant differences. We further report the name of better algorithm.

Table 7: Results of unrestricted ADVRP on TSPLIB asymmetric instances.

| INST | $n$ | $M$ | $D_{\max}$ | Max1 | GA | | | GA-INS | | | | GA-INV | | | | GA-SWP | | | | GA-ADP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | AS | SD | AT | AS | SD | AT | AI | AS | SD | AT | AI | AS | SD | AT | AI | AS | SD | AT | AI |
| br17 | 17 | 2 | Inf | 39 | 39.00 | 0.00 | 0.02 | 39.00 | 0.00 | 0.00 | 0.00 | 39.00 | 0.00 | 0.00 | 0.00 | 39.00 | 0.00 | 0.00 | 0.00 | 39.00 | 0.00 | 0.00 | 0.00 |
| | | 3 | Inf | 31 | 42.00 | 0.00 | 0.00 | 42.00 | 0.00 | 0.00 | 0.00 | 42.00 | 0.00 | 0.00 | 0.00 | 42.00 | 0.00 | 0.00 | 0.00 | 42.00 | 0.00 | 0.00 | 0.00 |
| ftv33 | 34 | 2 | Inf | 1216 | 1381.78 | 13.07 | 0.17 | 1352.07 | 26.52 | 0.22 | 2.20 | 1356.70 | 23.92 | 0.25 | 1.85 | 1353.37 | 23.13 | 0.18 | 2.10 | 1346.72 | 24.07 | 0.11 | 2.60 |
| | | 3 | Inf' | 1195 | 1400.20 | 15.48 | 0.18 | 1371.03 | 16.70 | 0.16 | 2.13 | 1365.07 | 20.14 | 0.22 | 2.57 | 1365.10 | 24.61 | 0.18 | 2.57 | 1356.47 | 21.52 | 0.12 | 3.22 |
| ftv35 | 36 | 2 | Inf | 1463 | 1563.25 | 23.68 | 0.26 | 1491.50 | 9.59 | 0.34 | 4.81 | 1499.03 | 16.53 | 0.34 | 4.28 | 1494.30 | 17.92 | 0.22 | 4.61 | 1492.60 | 6.25 | 0.16 | 4.73 |
| | | 3 | Inf | 1393 | 1558.00 | 14.31 | 0.23 | 1517.83 | 13.84 | 0.31 | 2.65 | 1524.93 | 12.88 | 0.31 | 2.17 | 1519.27 | 13.93 | 0.23 | 2.55 | 1517.10 | 11.21 | 0.18 | 2.70 |
| ftv38 | 39 | 2 | Inf | 1533 | 1637.23 | 27.51 | 0.66 | 1546.33 | 11.33 | 0.79 | 5.88 | 1574.97 | 19.42 | 0.42 | 3.95 | 1560.90 | 19.07 | 0.33 | 4.89 | 1547.32 | 4.71 | 0.21 | 5.81 |
| | | 3 | Inf | 886 | 1619.12 | 24.45 | 0.63 | 1576.57 | 8.64 | 0.61 | 2.70 | 1596.03 | 19.65 | 0.55 | 1.45 | 1592.10 | 20.60 | 0.39 | 1.70 | 1576.57 | 8.02 | 0.22 | 2.70 |
| $p43$ | 43 | 2 | Inf | 5617 | 5645.56 | 2.28 | 1.07 | 5634.83 | 3.04 | 1.21 | 0.19 | 5633.00 | 0.00 | 0.82 | 0.22 | 5633.07 | 2.61 | 0.56 | 0.22 | 5633.00 | 0.00 | 0.21 | 0.22 |
| | | 3 | Inf | 5613 | 5720.42 | 3.76 | 1.08 | 5688.63 | 4.23 | 1.19 | 0.56 | 5685.00 | 0.00 | 0.91 | 0.62 | 5685.17 | 3.55 | 0.61 | 0.62 | 5685.00 | 0.00 | 0.18 | 0.62 |
| ftv44 | 45 | 2 | Inf | 1609 | 1765.41 | 28.14 | 1.13 | 1650.67 | 20.87 | 1.13 | 6.95 | 1670.07 | 40.69 | 0.95 | 5.71 | 1646.10 | 32.45 | 0.85 | 7.25 | 1654.60 | 28.23 | 0.53 | 6.70 |
| | | 3 | Inf | 1547 | 1786.88 | 16.69 | 1.02 | 1680.88 | 21.09 | 1.19 | 6.31 | 1704.77 | 47.25 | 1.06 | 4.82 | 1685.40 | 31.95 | 0.91 | 6.02 | 1678.17 | 20.15 | 0.56 | 6.48 |
| ftv47 | 48 | 2 | Inf | 1311 | 2062.06 | 104.55 | 1.13 | 1876.53 | 22.21 | 1.49 | 9.89 | 1890.23 | 47.82 | 1.18 | 9.09 | 1872.00 | 34.85 | 1.05 | 10.15 | 1858.17 | 31.16 | 0.74 | 10.97 |
| | | 3 | Inf | 1672 | 2169.50 | 68.78 | 1.12 | 1942.43 | 27.10 | 1.26 | 11.69 | 1967.57 | 56.26 | 1.22 | 10.26 | 1943.37 | 26.29 | 1.15 | 11.64 | 1933.23 | 27.42 | 0.84 | 12.22 |
| ry48p | 48 | 2 | Inf | 14255 | 16263.01 | 188.67 | 1.36 | 14923.97 | 125.70 | 1.96 | 8.97 | 14828.96 | 66.17 | 1.28 | 9.67 | 14972.56 | 136.60 | 1.31 | 8.62 | 14835.00 | 51.33 | 1.41 | 9.63 |
| | | 3 | Inf | 14119 | 16326.01 | 202.59 | 1.27 | 15181.33 | 99.35 | 2.03 | 7.54 | 15087.63 | 56.77 | 2.01 | 8.21 | 15226.03 | 118.73 | 1.42 | 7.22 | 15083.16 | 47.57 | 1.74 | 8.24 |
| ft53 | 53 | 2 | Inf | 4461 | 8380.06 | 242.42 | 2.07 | 7198.50 | 141.50 | 2.80 | 16.41 | 7374.43 | 192.59 | 2.03 | 13.64 | 7203.68 | 122.46 | 1.93 | 16.33 | 7128.80 | 160.23 | 1.77 | 17.55 |
| | | 3 | Inf | 3452 | 8307.16 | 180.16 | 2.13 | 7333.50 | 103.49 | 2.98 | 13.28 | 7661.60 | 217.87 | 2.12 | 8.43 | 7390.27 | 98.26 | 2.06 | 12.41 | 7305.03 | 135.92 | 1.82 | 13.72 |
| ftv55 | 56 | 2 | Inf | 1053 | 1861.16 | 43.79 | 2.28 | 1719.70 | 23.68 | 2.36 | 8.23 | 1748.90 | 31.06 | 2.23 | 6.42 | 1716.57 | 29.70 | 2.09 | 8.42 | 1697.23 | 22.66 | 1.86 | 9.66 |
| | | 3 | Inf | 947 | 1919.06 | 40.38 | 2.33 | 1789.63 | 30.09 | 2.71 | 7.23 | 1827.70 | 28.15 | 2.17 | 5.00 | 1781.30 | 33.32 | 2.16 | 7.73 | 1774.03 | 21.02 | 1.98 | 8.18 |
| ftv64 | 65 | 2 | Inf | 1786 | 2131.36 | 32.85 | 3.47 | 1903.07 | 28.84 | 3.92 | 12.00 | 1957.00 | 48.76 | 3.01 | 8.91 | 1883.33 | 41.35 | 2.73 | 13.17 | 1884.13 | 29.65 | 2.72 | 13.12 |
| | | 3 | Inf | 1679 | 2090.06 | 44.66 | 3.33 | 1917.40 | 26.45 | 3.85 | 9.00 | 1971.60 | 53.26 | 3.14 | 6.01 | 1916.17 | 20.67 | 2.94 | 9.07 | 1909.27 | 29.17 | 2.91 | 9.47 |
| ft70 | 70 | 2 | Inf | 38585 | 41879.88 | 241.22 | 4.03 | 39942.93 | 187.13 | 4.17 | 4.85 | 40898.37 | 229.28 | 3.96 | 2.40 | 39751.03 | 305.59 | 3.01 | 5.36 | 39762.73 | 250.41 | 3.01 | 5.32 |
| | | 3 | Inf | 22295 | 42709.01 | 270.83 | 4.16 | 40758.00 | 264.21 | 4.11 | 4.79 | 41774.53 | 189.39 | 3.80 | 2.24 | 40696.80 | 294.00 | 3.35 | 4.94 | 40701.40 | 235.31 | 3.02 | 4.93 |
| ftv70 | 71 | 2 | Inf | 1938 | 2187.36 | 39.83 | 4.53 | 2002.83 | 38.13 | 4.59 | 9.21 | 2131.10 | 45.18 | 4.03 | 2.64 | 2007.77 | 43.45 | 3.75 | 8.94 | 2022.33 | 24.79 | 3.04 | 8.16 |
| | | 3 | Inf | 1888 | 2240.89 | 38.73 | 4.78 | 2073.70 | 37.76 | 4.96 | 8.06 | 2165.27 | 34.41 | 4.16 | 3.49 | 2039.90 | 43.05 | 3.90 | 9.85 | 2047.93 | 25.10 | 3.13 | 9.42 |
| kro124p | 100 | 2 | Inf | 27336 | 43753.76 | 746.76 | 5.43 | 39387.77 | 603.29 | 7.27 | 11.08 | 40520.10 | 470.44 | 6.58 | 7.98 | 38541.40 | 321.72 | 5.28 | 13.52 | 38427.57 | 310.33 | 5.02 | 13.86 |
| | | 3 | Inf | 22203 | 44607.76 | 528.36 | 5.44 | 39329.70 | 442.91 | 8.16 | 13.42 | 41207.70 | 359.10 | 6.84 | 8.25 | 39038.83 | 508.58 | 6.53 | 14.27 | 39073.87 | 425.54 | 5.21 | 14.16 |
| ftv170 | 171 | 2 | Inf | 2900 | 3611.78 | 66.87 | 14.81 | 3211.37 | 90.66 | 32.88 | 12.47 | 3379.13 | 82.69 | 31.15 | 6.88 | 3103.03 | 57.46 | 25.12 | 16.40 | 3157.23 | 76.97 | 29.66 | 14.40 |
| | | 3 | Inf | 1877 | 3569.32 | 44.16 | 14.37 | 3206.30 | 66.34 | 39.37 | 11.32 | 3391.97 | 88.52 | 39.08 | 5.23 | 3115.27 | 77.41 | 31.05 | 14.57 | 3140.53 | 80.42 | 31.63 | 13.65 |
| Average | | | | 9007.60 | 109.83 | 3.30 | 8309.67 | 83.16 | 9.26 | 7.13 | 8515.81 | 83.27 | 8.46 | 5.08 | 8260.50 | 83.44 | 9.14 | 7.51 | 8243.67 | 70.31 | 8.78 | 7.75 | |

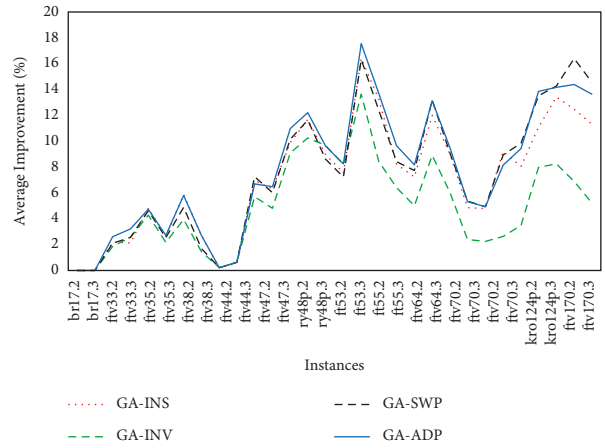


Figure 8: Average improvement (%) of solution by HGAs over solution by GA for TSPLIB asymmetric instances.

Table 8: The $t$-values against GA-ADP and the result about HGAs that found significantly better solutions for the unrestricted ADVRP.

| INST | GA-INS | GA-INV | GA-SWP | Instance | GA-INS | GA-INV | GA-SWP |
|---|---|---|---|---|---|---|---|
| br17.2 | — | — | — | ry48p.3 better | 6.24 | 0.42 | 7.82 |
| Better | — | — | — | | GA-ADP | — | — |
| br17.3 | — | — | — | ft53.2 | 2.28 | 6.86 | 2.60 |
| Better | — | — | — | Better | GA-ADP | GA-ADP | GA-ADP |
| ftv33.2 | 1.05 | 2.06 | 1.39 | ft53.3 | 1.17 | 9.72 | 3.56 |
| Better | — | GA-ADP | — | Better | — | GA-ADP | GA-ADP |
| ftv33.3 | 3.74 | 2.04 | 1.85 | ftv55.2 | 4.80 | 9.41 | 3.62 |
| Better | GA-ADP | GA-ADP | GA-ADP | Better | GA-ADP | GA-ADP | GA-ADP |
| ftv35.2 | −0.67 | 2.55 | 0.63 | ftv55.3 | 2.98 | 10.69 | 1.29 |
| Better | — | GA-ADP | — | Better | GA-ADP | GA-ADP | — |
| ftv35.3 | 0.29 | 3.21 | 0.85 | ftv64.2 | 3.21 | 8.94 | −0.11 |
| Better | — | GA-ADP | — | Better | GA-ADP | GA-ADP | — |
| ftv38.2 | −0.56 | 9.69 | 4.84 | ftv64.3 | 1.45 | 7.19 | 1.35 |
| Better | — | GA-ADP | GA-ADP | Better | — | GA-ADP | — |
| ftv38.3 | 0.00 | 6.42 | 4.92 | ft70.2 | 4.04 | 23.41 | −0.21 |
| Better | — | GA-ADP | GA-ADP | Better | GA-ADP | GA-ADP | — |
| $p43.2$ | 4.21 | — | 0.19 | ft70.3 | 1.12 | 24.87 | −0.09 |
| Better | GA-ADP | — | — | Better | — | GA-ADP | — |
| $p43.3$ | 6.01 | — | 0.34 | ftv70.2 | −3.00 | 14.77 | −2.04 |
| Better | GA-ADP | — | — | Better | GA-INS | GA-ADP | GA-SWP |
| ftv44.2 | −0.78 | 2.19 | −1.38 | ftv70.3 | 3.98 | 19.28 | −1.13 |
| Better | — | GA-ADP | — | Better | GA-ADP | GA-ADP | — |
| ftv44.3 | 0.65 | 3.62 | 1.34 | kro124p.2 | 9.91 | 25.99 | 1.78 |
| Better | — | GA-ADP | — | Better | GA-ADP | GA-ADP | GA-ADP |
| ftv47.2 | 3.36 | 3.93 | 2.07 | kro124p.3 | 2.92 | 26.83 | −0.37 |
| Better | GA-ADP | GA-ADP | GA-ADP | Better | GA-ADP | GA-ADP | — |
| ftv47.3 | 1.67 | 3.84 | 1.87 | ftv170.2 | 3.19 | 13.75 | −3.95 |
| Better | — | GA-ADP | GA-ADP | Better | GA-ADP | GA-ADP | GA-SWP |
| ry48p.2 | 4.59 | −0.50 | 6.60 | ftv170.3 | 4.42 | 14.72 | −1.58 |
| Better | GA-ADP | — | GA-ADP | Better | GA-ADP | GA-ADP | — |

The algorithms GA-ADP and GA-INS have no statistical and significant differences on thirteen instances. On the sixteen instances, GA-ADP is better than GA-INS, and only on ftv70.2, GA-INS is better than GA-ADP. On six instances, GA-ADP and GA-INV have no statistical and significant differences. On the remaining twenty-four instances, GA-ADP is better than GA-INV. On eighteen instances, GA-ADP and GA-SWP have no statistical and significant differences. On two instances—ftv70.2 and ftv170.2, GA-SWP is better than GA-ADP, and on the other ten instances, GA-ADP is better than GA-SWP. From these experimental results, we can tell that GA-ADP is statistically significant and is the best among the HGAs for unrestricted ADVRP instances.

Further, we conducted $t$-test to check whether average solutions obtained by GA-SWP are statistically and

Table 9: Results of restricted ADVRP where $D_{max} = 0.9 * Max_1$ is used.

| INST | $n$ | $m$ | $D_{max}$ | Max1 | GA | | | GA-INS | | | | GA-INV | | | | GA-SWP | | | | GA-ADP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | AS | SD | AT | AS | SD | AT | AI | AS | SD | AT | AI | AS | SD | AT | AI | AS | SD | AT | AI |
| *br17* | 17 | 2 | Inf | 39 | 39.00 | 0.00 | 0.02 | 39.00 | 0.00 | 0.00 | 0.00 | 39.00 | 0.00 | 0.00 | 0.00 | 39.00 | 0.00 | 0.00 | 0.00 | 39.00 | 0.00 | 0.00 | 0.00 |
| | | 3 | Inf | 31 | 42.00 | 0.00 | 0.00 | 42.00 | 0.00 | 0.00 | 0.00 | 42.00 | 0.00 | 0.00 | 0.00 | 42.00 | 0.00 | 0.00 | 0.00 | 42.00 | 0.00 | 0.00 | 0.00 |
| *ftv33* | 34 | 2 | Inf | 1216 | 1381.78 | 13.07 | 0.17 | 1352.07 | 26.52 | 0.22 | 2.20 | 1356.70 | 23.92 | 0.25 | 1.85 | 1353.37 | 23.13 | 0.18 | 2.10 | 1346.72 | 24.07 | 0.11 | 2.60 |
| | | 3 | Inf | 1195 | 1400.20 | 15.48 | 0.18 | 1371.03 | 16.70 | 0.16 | 2.13 | 1365.07 | 20.14 | 0.22 | 2.57 | 1365.10 | 24.61 | 0.18 | 2.57 | 1356.47 | 21.52 | 0.12 | 3.22 |
| *ftv35* | 36 | 2 | Inf | 1463 | 1563.25 | 23.68 | 0.26 | 1491.50 | 9.59 | 0.34 | 4.81 | 1499.03 | 16.53 | 0.34 | 4.28 | 1494.30 | 17.92 | 0.22 | 4.61 | 1492.60 | 6.25 | 0.16 | 4.73 |
| | | 3 | Inf | 1393 | 1558.00 | 14.31 | 0.23 | 1517.83 | 13.84 | 0.31 | 2.65 | 1524.93 | 12.88 | 0.31 | 2.17 | 1519.27 | 13.93 | 0.23 | 2.55 | 1517.10 | 11.21 | 0.18 | 2.70 |
| *ftv38* | 39 | 2 | Inf | 1533 | 1637.23 | 27.51 | 0.66 | 1546.33 | 11.33 | 0.79 | 5.88 | 1574.97 | 19.42 | 0.42 | 3.95 | 1560.90 | 19.07 | 0.33 | 4.89 | 1547.32 | 4.71 | 0.21 | 5.81 |
| | | 3 | Inf | 886 | 1619.12 | 24.45 | 0.63 | 1576.57 | 8.64 | 0.61 | 2.70 | 1596.03 | 19.65 | 0.55 | 1.45 | 1592.10 | 20.60 | 0.39 | 1.70 | 1576.57 | 8.02 | 0.22 | 2.70 |
| *p43* | 43 | 2 | Inf | 5617 | 5645.56 | 2.28 | 1.07 | 5634.83 | 3.04 | 1.21 | 0.19 | 5633.00 | 0.00 | 0.82 | 0.22 | 5633.07 | 2.61 | 0.56 | 0.22 | 5633.00 | 0.00 | 0.21 | 0.22 |
| | | 3 | Inf | 5613 | 5720.42 | 3.76 | 1.08 | 5688.63 | 4.23 | 1.19 | 0.56 | 5685.00 | 0.00 | 0.91 | 0.62 | 5685.17 | 3.55 | 0.61 | 0.62 | 5685.00 | 0.00 | 0.18 | 0.62 |
| *ftv44* | 45 | 2 | Inf | 1609 | 1765.41 | 28.14 | 1.13 | 1650.67 | 20.87 | 1.13 | 6.95 | 1670.07 | 40.69 | 0.95 | 5.71 | 1646.10 | 32.45 | 0.85 | 7.25 | 1654.60 | 28.23 | 0.53 | 6.70 |
| | | 3 | Inf | 1547 | 1786.88 | 16.69 | 1.02 | 1680.88 | 21.09 | 1.19 | 6.31 | 1704.77 | 47.25 | 1.06 | 4.82 | 1685.40 | 31.95 | 0.91 | 6.02 | 1678.17 | 20.15 | 0.56 | 6.48 |
| *ftv47* | 48 | 2 | Inf | 1311 | 2062.06 | 104.55 | 1.13 | 1876.53 | 22.21 | 1.49 | 9.89 | 1890.23 | 47.82 | 1.18 | 9.09 | 1872.00 | 34.85 | 1.05 | 10.15 | 1858.17 | 31.16 | 0.74 | 10.97 |
| | | 3 | Inf | 1672 | 2169.50 | 68.78 | 1.12 | 1942.43 | 27.10 | 1.26 | 11.69 | 1967.57 | 56.26 | 1.22 | 10.26 | 1943.37 | 26.29 | 1.15 | 11.64 | 1933.23 | 27.42 | 0.84 | 12.22 |
| *ry48p* | 48 | 2 | Inf | 14255 | 16263.01 | 188.67 | 1.36 | 14923.97 | 125.70 | 1.96 | 8.97 | 14828.96 | 66.17 | 1.28 | 9.67 | 14972.56 | 136.60 | 1.31 | 8.62 | 14835.00 | 51.33 | 1.41 | 9.63 |
| | | 3 | Inf | 14119 | 16326.01 | 202.59 | 1.27 | 15181.33 | 99.35 | 2.03 | 7.54 | 15087.63 | 56.77 | 2.01 | 8.21 | 15226.03 | 118.73 | 1.42 | 7.22 | 15083.16 | 47.57 | 1.74 | 8.24 |
| *ft53* | 53 | 2 | Inf | 4461 | 8380.06 | 242.42 | 2.07 | 7198.50 | 141.50 | 2.80 | 16.41 | 7374.43 | 192.59 | 2.03 | 13.64 | 7203.68 | 122.46 | 1.93 | 16.33 | 7128.80 | 160.23 | 1.77 | 17.55 |
| | | 3 | Inf | 3452 | 8307.16 | 180.16 | 2.13 | 7333.50 | 103.49 | 2.98 | 13.28 | 7661.60 | 217.87 | 2.12 | 8.43 | 7390.27 | 98.26 | 2.06 | 12.41 | 7305.03 | 135.92 | 1.82 | 13.72 |
| *ftv55* | 56 | 2 | Inf | 1053 | 1861.16 | 43.79 | 2.28 | 1719.70 | 23.68 | 2.36 | 8.23 | 1748.90 | 31.06 | 2.23 | 6.42 | 1716.57 | 29.70 | 2.09 | 8.42 | 1697.23 | 22.66 | 1.86 | 9.66 |
| | | 3 | Inf | 947 | 1919.06 | 40.38 | 2.33 | 1789.63 | 30.09 | 2.71 | 7.23 | 1827.70 | 28.15 | 2.17 | 5.00 | 1781.30 | 33.32 | 2.16 | 7.73 | 1774.03 | 21.02 | 1.98 | 8.18 |
| *ftv64* | 65 | 2 | Inf | 1786 | 2131.36 | 32.85 | 3.47 | 1903.07 | 28.84 | 3.92 | 12.00 | 1957.00 | 48.76 | 3.01 | 8.91 | 1883.33 | 41.35 | 2.73 | 13.17 | 1884.13 | 29.65 | 2.72 | 13.12 |
| | | 3 | Inf | 1679 | 2090.06 | 44.66 | 3.33 | 1917.40 | 26.45 | 3.85 | 9.00 | 1971.60 | 53.26 | 3.14 | 6.01 | 1916.17 | 20.67 | 2.94 | 9.07 | 1909.27 | 29.17 | 2.91 | 9.47 |
| *ft70* | 70 | 2 | Inf | 38585 | 41879.88 | 241.22 | 4.03 | 39942.93 | 187.13 | 4.17 | 4.85 | 40898.37 | 229.28 | 3.96 | 2.40 | 39751.03 | 305.59 | 3.01 | 5.36 | 39762.73 | 250.41 | 3.01 | 5.32 |
| | | 3 | Inf | 22295 | 42709.01 | 270.83 | 4.16 | 40758.00 | 264.21 | 4.11 | 4.79 | 41774.53 | 189.39 | 3.80 | 2.24 | 40696.80 | 294.00 | 3.35 | 4.94 | 40701.40 | 235.31 | 3.02 | 4.93 |
| *ftv70* | 71 | 2 | Inf | 1938 | 2187.36 | 39.83 | 4.53 | 2002.83 | 38.13 | 4.59 | 9.21 | 2131.10 | 45.18 | 4.03 | 2.64 | 2007.77 | 43.45 | 3.75 | 8.94 | 2022.33 | 24.79 | 3.04 | 8.16 |
| | | 3 | Inf | 1888 | 2240.89 | 38.73 | 4.78 | 2073.70 | 37.76 | 4.96 | 8.06 | 2165.27 | 34.41 | 4.16 | 3.49 | 2039.90 | 43.05 | 3.90 | 9.85 | 2047.93 | 25.10 | 3.13 | 9.42 |
| *kro124p* | 100 | 2 | Inf | 27336 | 43753.76 | 746.76 | 5.43 | 39387.77 | 603.29 | 7.27 | 11.08 | 40520.10 | 470.44 | 6.58 | 7.98 | 38541.40 | 321.72 | 5.28 | 13.52 | 38427.57 | 310.33 | 5.02 | 13.86 |
| | | 3 | Inf | 22203 | 44607.76 | 528.36 | 5.44 | 39329.70 | 442.91 | 8.16 | 13.42 | 41207.70 | 359.10 | 6.84 | 8.25 | 39038.83 | 508.58 | 6.53 | 14.27 | 39073.87 | 425.54 | 5.21 | 14.16 |
| *ftv170* | 171 | 2 | Inf | 2900 | 3611.78 | 66.87 | 14.81 | 3211.37 | 90.66 | 32.88 | 12.47 | 3379.13 | 82.69 | 31.15 | 6.88 | 3103.03 | 57.46 | 25.12 | 16.40 | 3157.23 | 76.97 | 29.66 | 14.40 |
| | | 3 | Inf | 1877 | 3569.32 | 44.16 | 14.37 | 3206.30 | 66.34 | 39.37 | 11.32 | 3391.97 | 88.52 | 39.08 | 5.23 | 3115.27 | 77.41 | 31.05 | 14.57 | 3140.53 | 80.42 | 31.63 | 13.65 |
| *Average* | | | | 9007.60 | 109.83 | 3.30 | 8309.67 | 83.16 | 9.26 | 7.13 | 8515.81 | 83.27 | 8.46 | 5.08 | 8260.50 | 83.44 | 9.14 | 7.51 | 8243.67 | 70.31 | 8.78 | 7.75 | |

significantly distinct from the average solutions found by GA-INS. We saw (not reported here) that for 25 instances there is no statistical difference between them, and for 5 instances, GA-SWP is better than GA-INS. So, GA-SWP is the second best.

Table 9 reports the results for restricted ADVRP instances where $D_{\max} = 0.9 * \text{Max}_1$ is used to find $\text{Max}_2$. From this table, it is seen that the GA could find best average solutions for the instance br17 with both 2 and 3 vehicles. The algorithms GA-INS, GA-INV, GA-SWP, and GA-ADP could find best average solutions for 4, 2, 6, and 22 instances, respectively. On average, GA-ADP, GA-SWP, GA-INS, and GA-INV have average improvement (%) as 10.97, 9.97, 9.72, and 6.59, respectively. It shows that the average improvement of GA-ADP is the largest, GA-SWP is the second largest, GA-INS is the third largest, and GA-INV shows the smallest average improvement. From these results, we can tell that GA-ADP is the best one, GA-SWP is the second best, GA-INS is the third best, and GA-INV is positioned in fourth position. Further, by looking at SD, we can say that results by GA-ADP are stable because its obtained solutions have lower SD. It is to be noted that no algorithm could solve the instance *p*43 with both 2 and 3 vehicles, so their results are not stated in Table 9. Figure 9 shows the average improvements (%) that also signifies the appropriateness of the HGAs, especially GA-ADP and GA-SWP. So, for these restricted ADVRP instances GA-ADP is the best algorithm and GA-SWP is the second best algorithm. Regarding the computational time, almost all HGAs are taking same time. However, simple GA takes less time. We further can see in this table that a number of vehicles have significant effect on the solution; i.e., almost for all instances, as the number of vehicles increases, solution also increases.

We see from the experiment that HGAs have fantastic improvements in the solution over GA for the restricted ADVRP instances. Among the algorithms, GA-ADP is the best and GA is the worst. To confirm whether average solutions obtained by GA-ADP are statistically and significantly distinct from the average solutions found by other HGAs, Student's *t*-test is performed, and the results are shown in Table 10. There is no statistical and significant difference between GA-INS and GA-ADP on twelve instances. On the remaining sixteen instances, GA-ADP is better than GA-INS. There is no statistical and significant difference between GA-INV and GA-ADP on five instances. On the remaining twenty-three instances, GA-ADP is better than GA-INV. There is no statistical and significant difference between GA-SWP and GA-ADP on ten instances. On the seventeen instances, GA-ADP is better than GA-SWP. On only the instance ftv38 with 3 vehicles, GA-SWP is better than GA-ADP. From this experiment, we can say that GA-ADP is statistically significant and is the best among the HGAs for the restricted ADVRP instances also.

Further, we conducted *t*-test to check whether average solutions obtained by GA-SWP are statistically and significantly distinct from the average solutions found by GA-INS. We saw (not reported here) that for 20 instances there is no statistical difference between them, for 3 instances GA-INS is better than GA-SWP, and for 5 instances GA-SWP is better than GA-INS. So, GA-SWP is the second best and GA-INS is the third best one.

We further report the results in Table 11 for restricted ADVRP instances where $D_{\max} = 0.9 * \text{Max}_2$ is used to find $\text{Max}_3$. It is seen that no algorithm could solve the instances *p*43, ft53, and ftv170 with both 2 and 3 vehicles; ftv35, ftv38, ftv44, ftv47, ftv55, ft70, and kro124p with 2 vehicles; and br17 with 3 vehicles. It seems that these problem instances are more complex. So, we did not report them, and we reported the results on 17 instances only.

Among the reported instances, GA could not solve kro124p with 2 vehicles; GA-INV could not solve ftv38 with 3 vehicles, and ftv64 and kro124p with 2 vehicles; however, the algorithms GA-INS, GA-SWP, and GA-ADP could solve these instances. It is noticed that the GA could find best average solutions for the instance br17 with 2 vehicles only. The algorithms GA-INS, GA-INV, GA-SWP, and GA-ADP could find best average solutions for 2, 1, 5, and 12 instances, respectively.

On average, GA-ADP, GA-INS, GA-SWP, and GA-INV have average improvement (%) as 10.77, 9.90, 8.98, and 6.17, respectively. It shows that the average improvement of GA-ADP is the largest, GA-INS is the second largest, GA-SWP is the third largest, and GA-INV shows the smallest average improvement. From these results, we can tell that for these restricted ADVRP instances GA-ADP is the best one, GA-INS is the second best, GA-SWP is the third best, and GA-INV is positioned in fourth position. Further, by looking at SD, we can say that results by GA-ADP are stable because its obtained solutions have lowest SD.

We further can see in this table that a number of vehicles have significant effect on the solution; i.e., almost for all instances, as the number of vehicles increases, solution also increases. It is also observed that as the distance-constrained becomes tight finding feasible solution becomes difficult. Regarding the computational time, almost all HGAs are taking same time. However, simple GA takes less time.

To prove whether average solutions found by GA-ADP are statistically and significantly different from the average solutions found by remaining HGAs, we conducted Student's *t*-test and reported the results in Table 12. There is no statistical and significant difference between GA-INS and GA-ADP on five instances. On one instance, GA-INS is better, and on the other ten instances GA-ADP is better. There is no statistical and significant difference between GA-INV and GA-ADP on four instances. On the remaining twelve instances, GA-ADP is better. There is no statistical and significant difference between GA-SWP and GA-ADP on three instances, on one instance GA-SWP is better, and on the other twelve instances GA-ADP is better. From this experiment, we can say that GA-ADP is the best for the restricted ADVRP instances also. However, GA-INS and GA-SWP are still competing for 2nd rank. We further perform Student's *t*-test between GA-SWP and GA-INS but found them equivalent. From all above experiments, we can assume that GA-ADP is the best, GA-SWP and GA-INS are the second best, and GA is the worst.

We further report a performance comparison of GA-ADP against HVT algorithm [9] on some asymmetric CVRP

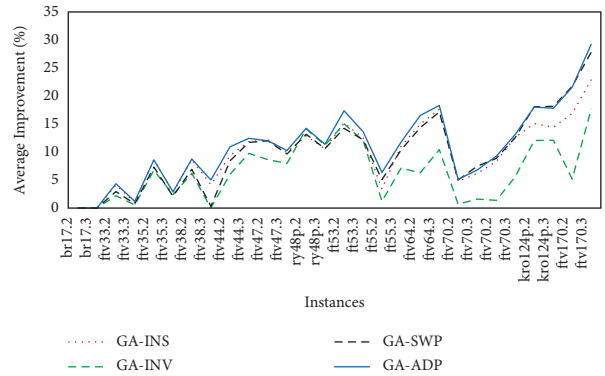


Figure 9: Average improvement (%) of solution by HGAs over solution by GA for TSPLIB asymmetric instances.

Table 10: The $t$-values against GA-ADP and the result about HGAs that found significantly better solutions for the restricted ADVRP.

| INST | GA-INS | GA-INV | GA-SWP | Instance | GA-INS | GA-INV | GA-SWP |
|---|---|---|---|---|---|---|---|
| br17.2 | — | — | — | ft53.2 better | 2.86 GA-ADP | 2.99 GA-ADP | 4.01 GA-ADP |
| Better | — | — | — | | | | |
| br17.3 | — | — | — | ft53.3 better | 1.58 | 2.07 GA-ADP | 2.06 GA-ADP |
| Better | — | — | — | | — | | |
| ftv33.2 | 1.13 | 7.64 | 4.80 | ftv55.2 | 10.56 | 12.43 | 3.33 |
| Better | — | GA-ADP | GA-ADP | Better | GA-ADP | GA-ADP | GA-ADP |
| ftv33.3 | 1.20 | 5.58 | 5.49 | ftv55.3 | 1.68 | 12.97 | 6.16 |
| Better | — | GA-ADP | GA-ADP | Better | — | GA-ADP | GA-ADP |
| ftv35.2 | −0.26 | 5.88 | 2.80 | ftv64.2 | 4.82 | 13.55 | 5.44 |
| Better | — | GA-ADP | GA-ADP | Better | GA-ADP | GA-ADP | GA-ADP |
| ftv35.3 | 0.98 | 1.65 | 1.85 | ftv64.3 | 3.48 | 21.04 | 5.44 |
| Better | — | — | GA-ADP | Better | GA-ADP | GA-ADP | GA-ADP |
| ftv38.2 | 0.84 | 11.39 | 7.68 | ft70.2 | 1.75 | 27.89 | 0.58 |
| Better | — | GA-ADP | GA-ADP | Better | GA-ADP | GA-ADP | — |
| ftv38.3 | 2.00 | 5.01 | 11.78 | ft70.3 | 5.06 | 29.94 | −2.30 |
| Better | GA-ADP | GA-ADP | GA-ADP | Better | GA-ADP | GA-ADP | GA-SWP |
| ftv44.2 | 4.50 | 7.19 | 4.95 | ftv70.2 | 1.93 | 18.94 | 0.75 |
| Better | GA-ADP | GA-ADP | GA-ADP | Better | GA-ADP | GA-ADP | — |
| ftv44.3 | 2.90 | 5.38 | 4.40 | ftv70.3 | 1.07 | 19.95 | 1.53 |
| Better | GA-ADP | GA-ADP | GA-ADP | Better | — | GA-ADP | — |
| ftv47.2 | −0.81 | 3.75 | −0.11 | kro124p.2 | 6.02 | 15.31 | −0.05 |
| Better | — | GA-ADP | — | Better | GA-ADP | GA-ADP | — |
| ftv47.3 | 1.65 | 6.00 | 1.58 | kro124p.3 | 7.01 | 17.21 | −0.98 |
| Better | — | GA-ADP | — | Better | GA-ADP | GA-ADP | — |
| ry48p.2 | 6.13 | 0.99 | 5.72 | ftv170.2 | 9.88 | 16.42 | −0.31 |
| Better | GA-ADP | — | GA-ADP | Better | GA-ADP | GA-ADP | — |
| ry48p.3 | 4.11 | 1.34 | 5.19 | ftv170.3 | 5.90 | 9.02 | 1.75 |
| Better | GA-ADP | — | GA-ADP | Better | GA-ADP | GA-ADP | GA-ADP |

(ACVRP) instances [28] of sizes from 34 to 71. As in [9], we run each instance 10 times. Further, we increase in the maximum generations to 250 generations for each run. The results are reported in Table 13. The percentage of gap (Gap) is calculated by

$$\text{Gap} = \frac{100\,(\text{BS} - \text{Opt})}{\text{Opt}}. \tag{15}$$

Looking at the best solutions, HVT could not find optimal solution for the A065-03f, whereas our proposed GA-ADP could find optimal solutions of all instances at least once in ten runs. So, in terms of best solution, our proposed algorithm GA-ADP is better than HVT. Looking at the average solutions, for the first four instances, both algorithms could obtain same average solutions, for the remaining three instances—A048-03f, A065-03f, and A071-03f, our GA-ADP is better than HVT, whereas only for the instance A056-03f, HVT is better than GA-ADP. Overall, our proposed algorithm GA-ADP is better than HVT. Regarding the computational time, HVT was executed on Intel Pentium core i7 duo 2.10 GHz CPU with 8 GB RAM, whereas our algorithm is executed on Intel Pentium core i7 1.30 GHz CPU with 8 GB RAM. It shows that their machine is faster than our machine. Looking at the computer specifications of both machines and computational times,

Table 11: Results of restricted ADVRP where $D_{max} = 0.9 * Max_2$ is used.

| INST | $n$ | M | $D_{max}$ | Max3 | GA | | | GA-INS | | | | GA-INV | | | | GA-SWP | | | | GA-ADP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | AS | SD | AT | AS | SD | AT | AI | AS | SD | AT | AI | AS | SD | AT | AI | AS | SD | AT | AI |
| br17 | 17 | 2 | 28 | 28 | 46.00 | 0.00 | 0.00 | 46.00 | 0.00 | 0.00 | 0.00 | 46.00 | 0.00 | 0.00 | 0.00 | 46.00 | 0.00 | 0.00 | 0.00 | 46.00 | 0.00 | 0.00 | 0.00 |
| ftv33 | 34 | 2 | 790 | 786 | 1478.23 | 23.02 | 0.24 | 1408.43 | 19.41 | 0.12 | 4.96 | 1426.78 | 17.18 | 0.37 | 3.61 | 1436.07 | 22.15 | 0.10 | 2.94 | 1405.73 | 19.21 | 0.06 | 5.16 |
| | | 3 | 790 | 748 | 1484.30 | 26.33 | 0.34 | 1393.30 | 14.25 | 0.15 | 6.53 | 1409.30 | 43.93 | 0.26 | 5.32 | 1403.23 | 22.79 | 0.21 | 5.78 | 1388.00 | 0.00 | 0.10 | 6.94 |
| ftv35 | 36 | 3 | 798 | 780 | 1769.78 | 61.73 | 0.24 | 1625.03 | 29.24 | 0.35 | 8.91 | 1703.43 | 149.72 | 0.32 | 3.90 | 1681.20 | 76.93 | 0.24 | 5.27 | 1617.07 | 70.57 | 0.21 | 9.44 |
| ftv38 | 39 | 3 | 716 | 671 | 1853.23 | 73.12 | 0.35 | 1693.50 | 26.58 | 0.26 | 9.43 | — | — | — | — | 1778.37 | 42.81 | 0.21 | 4.21 | 1687.27 | 77.94 | 0.34 | 9.84 |
| ftv44 | 45 | 3 | 823 | 657 | 2049.86 | 78.32 | 1.03 | 1828.80 | 72.03 | 0.34 | 12.09 | 1987.73 | 95.21 | 0.34 | 3.13 | 1890.27 | 65.21 | 1.05 | 8.44 | 1860.50 | 59.94 | 0.65 | 10.18 |
| ftv47 | 48 | 3 | 1143 | 957 | 2208.46 | 45.49 | 0.61 | 1944.73 | 9.21 | 1.07 | 13.56 | 1988.13 | 37.25 | 1.08 | 11.08 | 1969.10 | 43.19 | 1.06 | 12.16 | 1936.47 | 26.83 | 1.06 | 14.05 |
| ry48p | 48 | 2 | 11226 | 10872 | 17090.50 | 288.57 | 1.03 | 15048.43 | 115.14 | 1.39 | 13.57 | 14946.30 | 119.80 | 1.27 | 14.35 | 15124.60 | 193.42 | 1.05 | 13.00 | 14945.57 | 112.57 | 1.54 | 14.35 |
| | | 3 | 10986 | 10904 | 17131.78 | 296.36 | 0.61 | 15455.63 | 162.93 | 1.29 | 10.84 | 15390.07 | 96.14 | 1.13 | 11.32 | 15433.70 | 47.05 | 1.32 | 11.00 | 15340.63 | 82.47 | 1.27 | 11.68 |
| ftv55 | 56 | 3 | 766 | 736 | 2011.18 | 81.42 | 1.39 | 1820.17 | 20.66 | 1.80 | 10.49 | 1880.20 | 43.50 | 1.81 | 6.97 | 1822.70 | 44.53 | 2.21 | 10.34 | 1798.93 | 32.57 | 2.13 | 11.80 |
| ftv64 | 65 | 2 | 1202 | 1059 | 2242.23 | 78.34 | 1.32 | 1964.23 | 28.50 | 1.44 | 14.15 | — | — | — | — | 2010.07 | 64.82 | 2.16 | 11.55 | 1939.73 | 36.53 | 2.53 | 15.59 |
| | | 3 | 955 | 705 | 2318.26 | 92.56 | 2.05 | 2073.07 | 45.03 | 1.62 | 11.83 | 2157.27 | 75.26 | 2.31 | 7.46 | 2139.07 | 78.26 | 2.89 | 8.38 | 2058.43 | 73.28 | 3.06 | 12.62 |
| ft70 | 70 | 3 | 18059 | 16232 | 44730.41 | 304.83 | 3.63 | 41931.23 | 162.88 | 2.56 | 6.68 | 43881.17 | 410.78 | 4.54 | 1.94 | 42129.47 | 405.29 | 4.37 | 6.17 | 41776.10 | 421.49 | 5.17 | 7.07 |
| ftv70 | 71 | 2 | 1467 | 1067 | 2320.18 | 25.16 | 3.57 | 2166.03 | 62.12 | 2.98 | 7.12 | 2297.43 | 44.00 | 4.44 | 0.99 | 2076.40 | 39.24 | 4.09 | 11.74 | 2082.90 | 31.18 | 4.33 | 11.39 |
| | | 3 | 1521 | 1215 | 2322.59 | 34.95 | 3.11 | 2073.20 | 55.13 | 3.24 | 12.03 | 2255.23 | 51.37 | 3.32 | 2.99 | 2061.37 | 34.12 | 5.58 | 12.67 | 2063.67 | 37.98 | 4.59 | 12.55 |
| kro124p | 100 | 2 | 20751 | 19719 | — | — | — | 40602.53 | 894.25 | 5.82 | — | — | — | — | — | 40109.33 | 684.12 | 5.96 | — | 40351.37 | 423.12 | 8.02 | — |
| | | 3 | 17638 | 14999 | 48560.23 | 345.21 | 5.65 | 41801.93 | 1228.00 | 5.77 | 16.17 | 42842.53 | 465.71 | 6.11 | 13.35 | 40445.80 | 743.17 | 7.60 | 20.06 | 40559.37 | 858.64 | 8.02 | 19.73 |
| Average | | | | 9351.08 | 115.96 | 1.57 | 10286.84 | 173.26 | 1.52 | 9.90 | 9586.54 | 117.85 | 1.95 | 6.17 | 10209.22 | 153.36 | 2.13 | 8.98 | 10168.10 | 139.08 | 2.53 | 10.77 | |

Table 12: The $t$-values against GA-ADP and the result about HGAs that found significantly better solutions for the restricted ADVRP.

| INST | GA-INS | GA-INV | GA-SWP | Instance | GA-INS | GA-INV | GA-SWP |
|---|---|---|---|---|---|---|---|
| ftv33.2 | 0.69 | 5.72 | 7.24 | ftv55.3 | 3.85 | 10.47 | 3.02 |
| Better | — | **GA-ADP** | **GA-ADP** | Better | **GA-ADP** | **GA-ADP** | **GA-ADP** |
| ftv33.3 | 2.60 | 3.39 | 4.68 | ftv64.2 | 3.70 | — | 6.62 |
| Better | **GA-ADP** | **GA-ADP** | **GA-ADP** | Better | **GA-ADP** | — | **GA-ADP** |
| ftv35.3 | 0.73 | 3.65 | 4.30 | ftv64.3 | 1.19 | 6.59 | 5.27 |
| Better | — | **GA-ADP** | **GA-ADP** | Better | — | **GA-ADP** | **GA-ADP** |
| ftv38.3 | 0.53 | — | 7.17 | ft70.3 | 2.40 | 25.04 | 4.23 |
| Better | — | — | **GA-ADP** | Better | **GA-ADP** | **GA-ADP** | **GA-ADP** |
| ftv44.3 | −2.37 | 7.92 | 2.35 | ftv70.2 | 8.37 | 27.85 | −0.91 |
| Better | **GA-INS** | **GA-ADP** | **GA-ADP** | Better | **GA-ADP** | **GA-ADP** | — |
| ftv47.3 | 2.04 | 7.88 | 4.49 | ftv70.3 | 1.00 | 20.99 | −0.32 |
| Better | **GA-ADP** | **GA-ADP** | **GA-ADP** | Better | — | **GA-ADP** | — |
| ry48p.2 | 4.47 | 0.03 | 5.60 | kro124p.2 | 4.16 | — | −4.00 |
| Better | **GA-ADP** | — | **GA-ADP** | Better | **GA-ADP** | — | **GA-SWP** |
| ry48p.3 | 4.41 | 2.73 | 6.86 | kro124p.3 | 5.80 | 16.36 | −0.70 |
| Better | **GA-ADP** | **GA-ADP** | **GA-ADP** | Better | **GA-ADP** | **GA-ADP** | — |

Table 13: A comparative study between HVT [9] and GA-ADP on the ACVRP.

| INST | $n$ | $m$ | Opt | HVT | | | | GA-ADP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | BS | AS | Gap | Time | BS | AS | Gap | Time |
| A034-02f | 34 | 2 | 1406 | 1406 | 1406.00 | 0.00 | 0.27 | 1406 | 1406.00 | 0.00 | 0.16 |
| A036-03f | 36 | 3 | 1644 | 1644 | 1644.00 | 0.00 | 0.31 | 1644 | 1644.00 | 0.00 | 0.18 |
| A039-03f | 39 | 3 | 1654 | 1654 | 1654.00 | 0.00 | 0.44 | 1654 | 1654.00 | 0.00 | 0.26 |
| A045-03f | 45 | 3 | 1740 | 1740 | 1740.00 | 0.00 | 0.52 | 1740 | 1740.00 | 0.00 | 0.53 |
| A048-03f | 48 | 3 | 1891 | 1891 | 1891.51 | 0.00 | 0.59 | 1891 | 1891.23 | 0.00 | 0.74 |
| A056-03f | 56 | 3 | 1739 | 1739 | 1739.00 | 0.00 | 0.63 | 1739 | 1739.01 | 0.00 | 1.96 |
| A065-03f | 65 | 3 | 1974 | 1976 | 1976.21 | 0.10 | 2.61 | 1974 | 1975.06 | 0.00 | 2.63 |
| A071-03f | 71 | 3 | 2054 | 2054 | 2054.51 | 0.00 | 2.80 | 2054 | 2054.35 | 0.00 | 2.95 |

one can say that our computational time is comparable with that of HVT. Overall, looking at the solution quality and computational time, our suggested GA-ADP is found to be better than HVT.

A real-life application of the ADVRP may be the sales representative who visits customers without pick up or delivery constraints but with distance constraints. This study uses three local search methods to develop three separate HGAs and adaptive search that randomly selects one of three local search methods to develop fourth HGA to solve the ADVRP. The fourth HGA, i.e., GA-ADP, provides cost-effective optimal solution to the problem. The proposed GA-ADP provides a cost-effective optimal routing plan to the sales representative. It is observed that as the number of vehicles increases solution value also increases, so removal of a vehicle from the fleet can reduce the workers. Hence, this gives managerial interpretation for the optimal fleet sizing and route designing.

## 6. Conclusion and Future Works

This paper developed a simple GA and four hybrid HGAs for solving the asymmetric distance-constrained vehicle routing problem (ADVRP). The proposed GA used random initial population followed by sequential constructive crossover and swap mutation. The HGAs improved the initial solution using 2-opt search method and incorporated local search techniques along with an immigration procedure to find better solution to this problem. Experimental study has been carried out among the proposed GA and HGAs, by solving some TSPLIB asymmetric instances of various sizes.

Three sets of experiments were performed on asymmetric TSPLIB instances. The first experiment was unrestricted ADVRP that used a very big predefined maximum distance for every vehicle, in the 2nd experiment, the predefined maximum distance was restricted by multiplying 0.9 to the maximum distance obtained in the 1st experiment, and the third experiment used the maximum distance as 0.9 multiple of maximum distance obtained in 2nd experiment. Our computational experience reveals that the suggested HGAs are very good. From the experiments, we found that HGA using adaptive search is the best, and HGA using swap search is the second best for the restricted and unrestricted ADVRP instances. We further performed Student's $t$-test and confirmed our claim. However, since no research reported the exact solutions for the instances, hence, we could not claim how good our obtained solutions are. So, one can verify the optimality of our best solutions, which is also

under our next investigation. However, it is observed that as the distance-constrained becomes tight finding feasible solution becomes difficult. Finally, we reported a comparative study between our GA-ADP and a state-of-the-art algorithm on asymmetric capacitated VRP and found that our algorithm is better than the state-of-the-art algorithm for the instances.

Though the proposed HGAs found very effective solutions with small differences among average solutions, we acknowledge that still there is possibility to enhance the solutions by merging better local search approaches and/or heuristic procedures and perturbation technique to the algorithms which will be our investigation. Also, proposing a new metaheuristic procedure for solving many other instances effectively could be very interesting for the researchers.

## Data Availability

The data set used to support the findings of this study is available from the corresponding author upon request.

## Conflicts of Interest

The authors declare that they have no potential conflicts of interest.

## Acknowledgments

The authors extend their appreciation to the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University for funding this work through research group no. RG-21-09-17.

## References

[1] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[2] F. Daneshzand, "The vehicle-routing problem," *Logistics Operations and Management*, vol. 8, pp. 127–153, 2011.

[3] P. Toth and D. Vigo, Eds., *Vehicle Routing: Problems, Methods, and Applications*, Society for Industrial and Applied Mathematics, 3600 University City Science Center Philadelphia, PA, USA, 2014.

[4] Z. H. Ahmed, "A lexisearch algorithm for the distance-constrained vehicle routing problem," *International Journal of Mathematical and Computational Methods*, vol. 1, pp. 165–174, 2016.

[5] S. Akpinar, "Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem," *Expert Systems with Applications*, vol. 61, pp. 28–38, 2016.

[6] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York, USA, 1989.

[7] K. Alabdulkareem and Z. H. Ahmed, "Comparison of four genetic crossover operators for solving distance-constrained vehicle routing problem," *IJCSNS International Journal of Computer Science and Network Security*, vol. 20, no. 7, pp. 114–123, 2020.

[8] Z. H. Ahmed, "Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator," *International Journal of Biometric and Bioinformatics*, vol. 3, no. 6, pp. 96–105, 2010.

[9] H.-B. Ban and P. K. Nguyen, "A hybrid metaheuristic for solving asymmetric distance-constrained vehicle routing problem," *Computational Social Networks*, vol. 8, no. 1, p. 3, 2021.

[10] G. Laporte, Y. Nobert, and S. Taillefer, "A branch-and-bound algorithm for the asymmetrical distance-constrained vehicle routing problem," *Journal of Mathematical Modelling*, vol. 9, no. 12, pp. 857–868, 1987.

[11] S. Almoustafa, S. Hanafi, and N. Mladenović, "New exact method for large asymmetric distance-constrained vehicle routing problem," *European Journal of Operational Research*, vol. 226, no. 3, pp. 386–394, 2013.

[12] M. H. Rachid, W. R. Cherif-Khettaf, P. Chatonnay, and C. Bloch, "A study of performance on crossover and mutation operators for vehicle routing problem," in *Proceedings of the 3rd International Conference on Information Systems, Logistics and Supply Chain - ILS'2010*, Casablanca, Morocco, April 2010.

[13] P. Krunoslav and M. Robert, "Comparison of eight evolutionary crossover operators for the vehicle routing problem," *Mathematical Communications*, vol. 18, pp. 359–375, 2013.

[14] Z. H. Ahmed, "A hybrid algorithm combining lexisearch and genetic algorithms for the quadratic assignment problem," *Cogent Engineering*, vol. 5, no. 1, Article ID 1423743, 2018.

[15] T. Tlili, S. Faiz, and S. Krichen, "A hybrid metaheuristic for the distance-constrained capacitated vehicle routing problem," *Procedia - Social and Behavioral Sciences*, vol. 109, pp. 779–783, 2014.

[16] Y. Xiao, Q. Zhao, I. Kaku, and N. Mladenovic, "Variable neighbourhood simulated annealing algorithm for capacitated vehicle routing problems," *Engineering Optimization*, vol. 46, no. 4, pp. 562–579, 2014.

[17] E. Teymourian, V. Kayvanfar, G. M. Komaki, and M. Zandieh, "Enhanced intelligent water drops and cuckoo search algorithms for solving the capacitated vehicle routing problem," *Information Sciences*, vol. 334-335, pp. 354–378, 2016.

[18] A. Faiz, S. Subiyanto, and U. M. Arief, "An efficient metaheuristic algorithm for solving capacitated vehicle routing problem," *International Journal of Advances in Intelligent Informatics*, vol. 4, no. 3, pp. 212–225, 2018.

[19] A. M. Altabeeb, A. M. Mohsen, and A. Ghallab, "An improved hybrid firefly algorithm for capacitated vehicle routing problem," *Applied Soft Computing*, vol. 84, Article ID 105728, 2019.

[20] İ. İlhan, "An improved simulated annealing algorithm with crossover operator for capacitated vehicle routing problem," *Swarm and Evolutionary Computation*, vol. 64, Article ID 100911, 2021.

[21] J. Euchi and A. Sadok, "Hybrid genetic-sweep algorithm to solve the vehicle routing problem with drones," *Physical Communication*, vol. 44, Article ID 101236, 2021.

[22] A. Baniamerian, M. Bashiri, and F. Zabihi, "A modified variable neighborhood search hybridized with genetic algorithm for vehicle routing problems with cross-docking," *Electronic Notes in Discrete Mathematics*, vol. 66, pp. 143–150, 2018.

[23] J. Long, Z. Sun, P. M. Pardalos, Y. Hong, S. Zhang, and C. Li, "A hybrid multi-objective genetic local search algorithm for the prize-collecting vehicle routing problem," *Information Sciences*, vol. 478, pp. 40–61, 2019.

[24] J. K. Lenstra and A. H. G. R. Kan, "Some simple applications of the travelling salesman problem," *Operational Research Quarterly (1970-1977)*, vol. 26, no. 4, pp. 717–733, 1975.

[25] Z. H. Ahmed, A. S. Hameed, M. L. Mutar, M. F. Alrifaie, and M. M. Taresh, "Experimental study of hybrid genetic algorithms for the maximum scatter travelling salesman problem," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 8, pp. 471–482, 2021.

[26] G. Reinelt, "TSPLIB," http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

[27] Z. H. Ahmed, "Adaptive sequential constructive crossover operator in a genetic algorithm for solving the traveling salesman problem," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 2, pp. 593–605, 2020.

[28] M. Fischetti, P. Toth, and D. Vigo, "A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs," *Operations Research*, vol. 42, no. 5, pp. 846–859, 1994, http://www.vrp-rep.org/datasets/item/2017-0002.html.