Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Solution of capacitated vehicle routing problem with invasive weed and hybrid algorithms

Ümit Yıldırım[a] and Yusuf Kuvvetli[b*]

[a]*Department of Industrial Engineering, Cukurova University, 01330, Adana, Turkey*

C H R O N I C L E

Article history:
Received February 18 2021
Received in Revised Format March 31 2021
Accepted April 1 2021
Available online April, 3 2021

Keywords:
Vehicle routing problem with capacity constraints
Invasive weed optimization algorithm
Genetic algorithm
Savings algorithm
Hybrid metaheuristics

A B S T R A C T

The vehicle routing problem is widespread in terms of optimization, which is known as being NP-Hard. In this study, the vehicle routing problem with capacity constraints is solved using cost- and time-efficient metaheuristic methods: an invasive weed optimization algorithm, genetic algorithm, savings algorithm, and hybridized variants. These algorithms are tested using known problem sets in the literature. Twenty-four instances evaluate the performance of algorithms from P and five instances from the CMT data set group. The invasive weed algorithm and its hybrid variant with savings and genetic algorithms are used to determine the best methodology regarding time and cost values. The proposed hybrid approach has found optimal P group problem instances with a 2% difference from the best-known solution on average. Similarly, the CMT group problem is solved with about a 10% difference from the best-known solution on average. That the proposed hybrid solutions have a standard deviation of less than 2% on average from BKS indicates that these approaches are consistent.

1. Introduction

One of the issues frequently evaluated in logistics problems is the minimization of transportation costs. The vehicle routing problem (VRP) aims to determine the routes to be followed by the vehicle or vehicles to provide the minimum transportation cost and maximum customer service level. By determining the most suitable routes, the total distances and total vehicle numbers are minimized (Pichpibul & Kawtummachai, 2012). The VRP is one of the most studied problems in terms of optimization. In the classical VRP, the problem's parameters are known in advance and do not change during the planning horizon. In the literature, studies on vehicle routing problems reveal that vehicle routing problems are of the NP-Hard class. In the classical vehicle routing problem, customer demands are met from a central depot with homogeneous vehicles; a tour begins and ends at the depot. Customer demands for the static version of the vehicle routing problem are known in advance and do not change until the distribution has been completed. Each customer should be visited only once, and customer demands on a route cannot exceed vehicle capacity. Under these conditions, the goal is to create minimum-cost routes. The primary objective of the vehicle routing problem is to decrease the distribution time and cost by reducing the total distance. An example of a vehicle routing problem is presented in Fig. 1. The vehicle routing problem examined in this study is a multi-vehicle, heterogeneous, and capacitated vehicle routing problem. This problem aims to distribute with a minimum number of vehicles and a minimum distance. The constraints encountered in this problem are the vehicle capacity and tour restrictions. The vehicle's capacity cannot exceed the capacity of that vehicle. Meanwhile, routing restrictions are the restrictions that enable the route of the vehicle to be determined. The main contribution of this study is to propose different approaches to solving this problem. In this context, a savings algorithm, genetic algorithm, invasive weed algorithm, and hybrid versions of these approaches are proposed to solve the problem. The results obtained have been evaluated with the test problems in previous studies, and a quicker and more consistent solution has been proposed with the hybrid approach.

* Corresponding author
E-mail: ykuvvetli@cu.edu.tr (Y. Kuvvetli)

2021 Growing Science Ltd.
doi: 10.5267/j.ijiec.2021.4.002

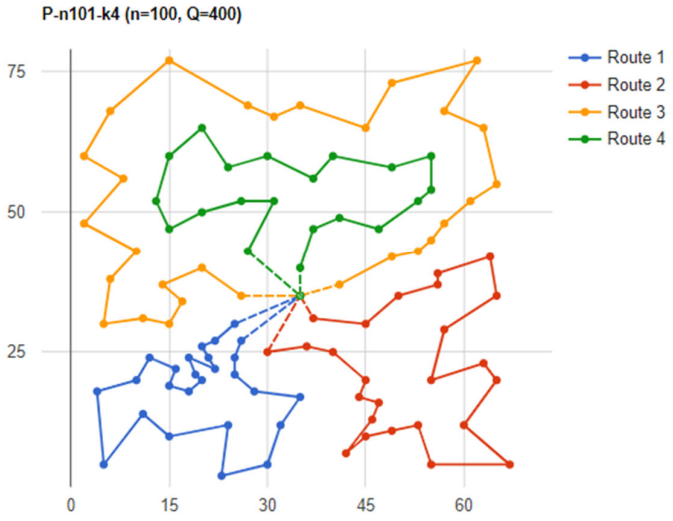


Fig. 1. An example VRP

2. Background

The vehicle routing problem entered the literature in 1959 with a study by Dantzig and Ramser (Dantzig & Ramser, 1959). In the study, they addressed the vehicle routing problem for the distribution of gasoline to gas stations, and for the first time, they developed a linear mathematical model and introduced an approach to the literature for the solution of this problem (Dantzig & Ramser, 1959). Meanwhile, with their work in 1964, Clarke and Wright (Clarke & Wright, 1964) proposed a heuristic algorithm for the first time for the vehicle routing problem. In their work, they explained the steps and operations of the algorithm and named their algorithm a savings algorithm. The savings algorithm is likewise known as the Clarke and Wright algorithm. When recent studies are evaluated, it is observed that heuristic approaches are often suggested due to the difficulty of finding an exact solution. In their study, Pisinger and Ropke proposed a combined heuristic algorithm for five types of vehicle routing problems (i.e., the time-limited vehicle routing problem, capacity vehicle routing problem, multi-depot vehicle routing problem, location-dependent vehicle routing problem, and open vehicle routing problem) (Pisinger & Ropke, 2007). A study conducted by Tarantilis et al. involving metaheuristic solution methodologies for the capacitated vehicle routing problem, a standard and widely studied version of the vehicle routing problem, is examined (Tarantilis, Ioannou, & Prastacos, 2005). Meanwhile, a hybrid approach consisting of ant colony optimization and the genetic algorithm has been proposed to solve the multi-depot vehicle routing problem (Yücenur & Demirel, 2011). Furthermore, Bozyer et al. proposed a heuristic method based on the grouping first and then route to solve the capacitated vehicle routing problem (CVRP) (Boyzer, Alkan, & Fığlalı, 2014). Finally, Wedyan and Narayanan proposed the intelligent water drop algorithm inspired by the water flow to solve capacitated vehicle routing (Wedyan & Narayanan, 2014). A Bilayer Local Search-based Particle Swarm Optimization for the solution of the CVRP is proposed (Ahmed & Sun, 2018). The researchers applied the proposed approach to the problem adopted from the literature. They demonstrated that they achieve the best-known solutions for most problems with short CPU times. Moreover, they noted that, according to the results, the performance achieved with the proposed algorithm outperformed some other particle swarm optimization-based approaches. Meanwhile, Cuevas et al. examined the CVRP for transportation companies and have verified the model with adapted test problems and actual data obtained from the transport company (Rojas-Cuevas, Caballero-Morales, Martinez-Flores, & Mendoza-Vazquez, 2018). Additionally, Arnold et al. explained how a local search heuristics method was designed that produced better solutions within a reasonable calculation period forvery large-scale examples of the CVRP (Arnold, Gendreau, & Sörensen, 2019). Through pruning and sequential search, they explored different means of reducing space complexity by limiting the time complexity and information stored. They revealed that the algorithm performs better than previously introduced heuristics for 10,000 or more client instances. The green capacitated vehicle routing problem through which alternative fuel-powered vehicles were used to distribute the products is addressed (Zhang, Gajpal, & Appadoo, 2018). Alternative fuel vehicles are assumed to have a low fuel tank capacity. Therefore, during the distribution processes, vehicles must visit alternative fuel stations for refueling. The two-phase heuristic algorithm and the ant colony algorithm-based algorithm are proposed as two methods for solving this problem. A simulated annealing-based heuristic approach is proposed for solving the green CVRP (Normasari, Yu, & Bachtiyar, 2019). The memetic algorithm is proposed for the green CVRP (Wang & Lu, 2019). Benrahou and Tairi addressed the CVRP for waste oil collection (Benrahou & Tairi, 2019). As such, they adapted the heuristic method called the Nearest Insertion Algorithm to solve the problem. Mulloorakam ve Nidhiry proposed a genetic algorithm method with swap mutation and an alternative crossover approach for a case of CVRP (Mulloorakam & Nidhiry, 2019). Furthermore, Toffolo et al. proposed a heuristic approach with some speed-up modifications, such as neighbor reductions, dynamic motion filters,

memory structures, and concatenation techniques (Toffolo, Vidal, & Wauters, 2019). Finally, a tunnel strategy was designed to reshape the search space as the algorithm progresses.

2.1. Studies on Invasive Weed Optimization Algorithm

The invasive weed optimization algorithm (IWO) is inspired by the invasive and robust growth of weeds, mimicking their robustness and adaptability. In a study by Mehrabian and Lucas, IWO was compared with evolutionary algorithms, such as genetic algorithms, memetic algorithms, particle swarm optimization, and the frog leap algorithm (Mehrabian & Lucas, 2006). Here, they employed the weed algorithm to solve an engineering problem. The experimental results demonstrated that the performance of the IWO is acceptable for all test functions. Meanwhile, Pahlavani et al. addressed the personalized urban multi-criteria path optimization problem (Pahlavani, Delavar, & Frank, 2012). This problem is a variant of the multi-criteria shortest path problem. To solve the problem, the modified IWO was applied and compared with the genetic algorithm. It was recorded that better results were obtained compared with the genetic algorithm. Additionally, Mohammadi et al. addressed stochastic green main distribution center location selection-routing, similar to the location selection-routing problem (Mohammadi, Razmi, & Tavakkoli-Moghaddam, 2013). Multi-objective mixed-integer linear programming formulations for the solution of the problem were proposed. They have likewise implemented multi-objective IWO, which is a hybrid multi-purpose heuristic. Here, they demonstrated that the proposed algorithm outperforms basic multi-purpose algorithms in the literature, such as NSGA-II (non-dominated sorting genetic algorithm II), PAES (Pareto archived evolution strategy), and SPEA (Strength Pareto Evolutionary Algorithm). Sur and Shukla discussed the Graph-Based Combinatorial Road Network Management Problem with the discrete IWO (Sur & Shukla, 2013). In their study, the discrete version of the IWO was introduced. They compared the discrete IWO with the ant colony algorithm and the intelligent water drop algorithm and presented them. The results noted that the discrete IWO exhibited a better convergence rate than the other two algorithms. Zhao et al. (2016) addressed the vehicle routing problem and employed the discrete hybrid IWO algorithm to solve it. To improve the algorithm results, they included adaptive mutation and adaptive crossover, a two-stage hybrid neighbor search algorithm. To evaluate the algorithm's performance, they selected A group, B group, and P group problem instances from the study by Augerat et al. (Augerat et al., 1995). As a result, when the capacity constraint is not considered, better results were obtained than the best-known problem. Nath et al. (2017) proposed the IWO algorithm to rotate the entire very large-scale integrated circuit (VLSI) routing. In VLSI routing, meta-heuristic algorithms, such as ant colony optimization, particle swarm optimization, and the firefly algorithm, have been applied in previous studies. Experiments have demonstrated that the IWO algorithm is a more effective algorithm for this problem than are the meta-heuristics used in the literature, such as particle swarm optimization and the firefly algorithm. The large-scale inventory routing problem is solved by two metaheuristics: the discrete IWO algorithm and genetic algorithm (Jahangir, Mohammadi, Pasandideh, & Nobari, 2019). The researchers used the solution results obtained to compare these algorithms. The results revealed that the discrete IWO algorithm has better convergence than the genetic algorithm. Additionally, they reported that the discrete IWO algorithm is better in terms of CPU time in small problem instances, while the genetic algorithm is better for large problems.

3. Methodology

3.1. Savings Algorithm

The Savings Algorithm, proposed by Clarke and Wright (Clarke & Wright, 1964) and also known as the Clarke and Wright algorithm, is a heuristic algorithm. The savings algorithm produces the optimum or near-optimum result in a single iteration. The necessary steps are presented in **Algorithm 1**. Accordingly, distances and savings values are calculated, and the savings values are sorted decreasingly. The rules of the savings algorithm are operated. As output, the problem is solved, and routes are created sequentially or simultaneously (in parallel). If demand points are added to the route until the vehicle capacity is complete, the routes are created sequentially. Otherwise, if routes are created simultaneously for each vehicle, the routes are created in parallel.

3.2 Genetic Algorithm

Genetic algorithms (GAs) originated from the theory of evolution. This algorithm is a search method obtained by applying the best natural selection to computers (Nabiyev, 2005). The GA is a search and optimization algorithm that bases on the evolution of living things. A GA encodes solution candidates in the solution space with a structure called a chromosome. It creates a new population by applying genetic operators, such as crossover and mutation, to solution candidates. A few generations later, the population includes members with better eligibility values (Jang, Sun, & Mizutani, 1997). A genetic algorithm is a search method based on evolution in nature (Davis, 1991) and has been found in many applications. The most significant reason for this is that the algorithm produces solutions for complex problems and has a solid and wide application area (Cheng, Gen, & Tsujimura, 1999).

Algorithm 1 Savings algorithm

```
procedure CW(d_ij.N)
  Routes← {}
  for i=1 to N do
    for j=1 to N do
      if i ≠ j then
        S_ij ←d_0i+d_j0-d_ij
      end if
    end for
  end for
  descend order S_ij
  for i=1 to N do
    for j=1 to N do
      if (i ∈ Routes) and (j ∈ Routes) then
      else if (i ∈ Routes) and (j ∉ Routes) then
        if i in the center
        else if i in the start of Routes(i)
          insert j as the first element to Routes(i)
        else if i at the end of Routes(i)
          insert j as the last element to Routes(i)
      else if (i ∉ Routes) and (j ∈ Routes) then
        if j in the center
        else if j in the start of Routes(j)
          insert i as the first element to Routes(j)
        else if j at the end of Routes(j)
          insert i as the last element to Routes(j)
      else
        create new route Routes(i.j)
      end if
    end for
  end for
  for i=1 to N do
    if (i ∉ Routes) then
      create new route Routes(i)
    end if
  end for
end procedure
```

To apply the genetic algorithm to a problem, the information and chromosome coding method that the gene will carry are determined. Afterward, the fitness function of the problem is determined. The genetic algorithm begins with possible solutions that are randomly generated, called populations. Then, the algorithm aims to produce better solutions by crossing individuals in the population. Diversity is provided by applying mutation to individuals at the determined mutation rate. Moreover, the population size parameter is determined to control the population size. Some individuals must be offspring, as the population will increase too much in each generation. At this phase, it is essential to protect better individuals and transfer them to other generations, a process called the concept of elitism. Individuals with high suitability are protected, while individuals with low elimination are eliminated during the elimination phase. According to the number of generations determined, the crossing process, mutation process, and elimination process are applied; in this way, the initial population is gradually improved. As a result of the algorithm, the best individual surviving in the population is the solution. A flow chart of the genetic algorithm is provided in Fig. 2.

In the CVRP with a genetic algorithm, customer visit orders are expressed as genes. A chromosome contains routes, and routes are a list of customer visit orders. Here, the index of the list determines the order of the customer. In the explanations, the individual and the chromosome are used synonymously. In Figure 3, sample gene and chromosome representation for 10 customer problem instances are presented. The fitness value of chromosomes in the population is determined as the total cost of the routes. To calculate the fitness value, genes on the chromosome are taken in order. Genes in the chromosome continue to insert routes until the vehicle's capacity is completed; if the amount of demand exceeds the vehicle capacity, a new route is created. This process repeats until all the genes in the chromosome are complete. Finally, depots are connected to the start and end of routes, and the cost of these routes is calculated. As a result of this process, the chromosomes' fitness values (individuals) in the population are calculated.

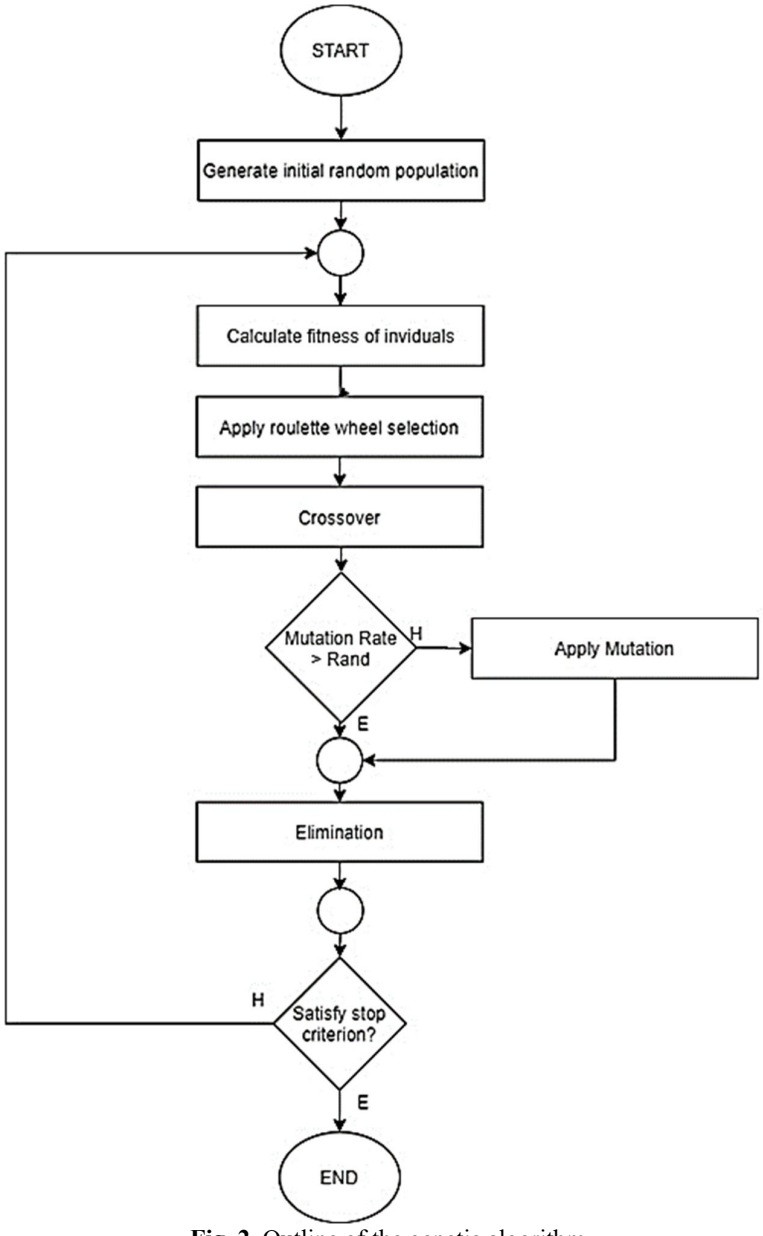


Fig. 2. Outline of the genetic algorithm

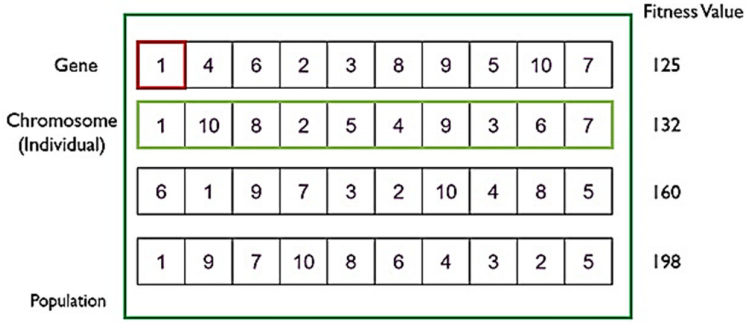


Fig. 3. Genetic Algorithm Components (Gene, Chromosome, Population, and Fitness Value)

The crossover methods are depicted in Figure 4. In this study, three crossover methods are used: swap, inversion, and insertion. Two new individuals are formed as a result of crossover, and repetitive genes can be found in these two new individuals. Since repetitive genes indicate coming to the same customer more than once, they cannot be accepted as a solution and must

be corrected. In the repairing process, repetitive genes are determined on the chromosome, one of the repeating genes remains, and the repairing process is applied by selecting randomly from the genes that have not been used on the chromosome. After repairing, new individuals are entirely created. The crossover method selection is performed by using Eq. (1):

$$crossover = \begin{cases} swap & 0 \le u < p_1 \\ inversion & p_1 \le u < p_2 \\ insertion & p_2 \le u < 1 \end{cases} \quad (1)$$

$where\ u = random(0.1).\ p_1$ and p_2 $are\ user\ defined\ probabilities\ for$ which $p_1 < p_2$.

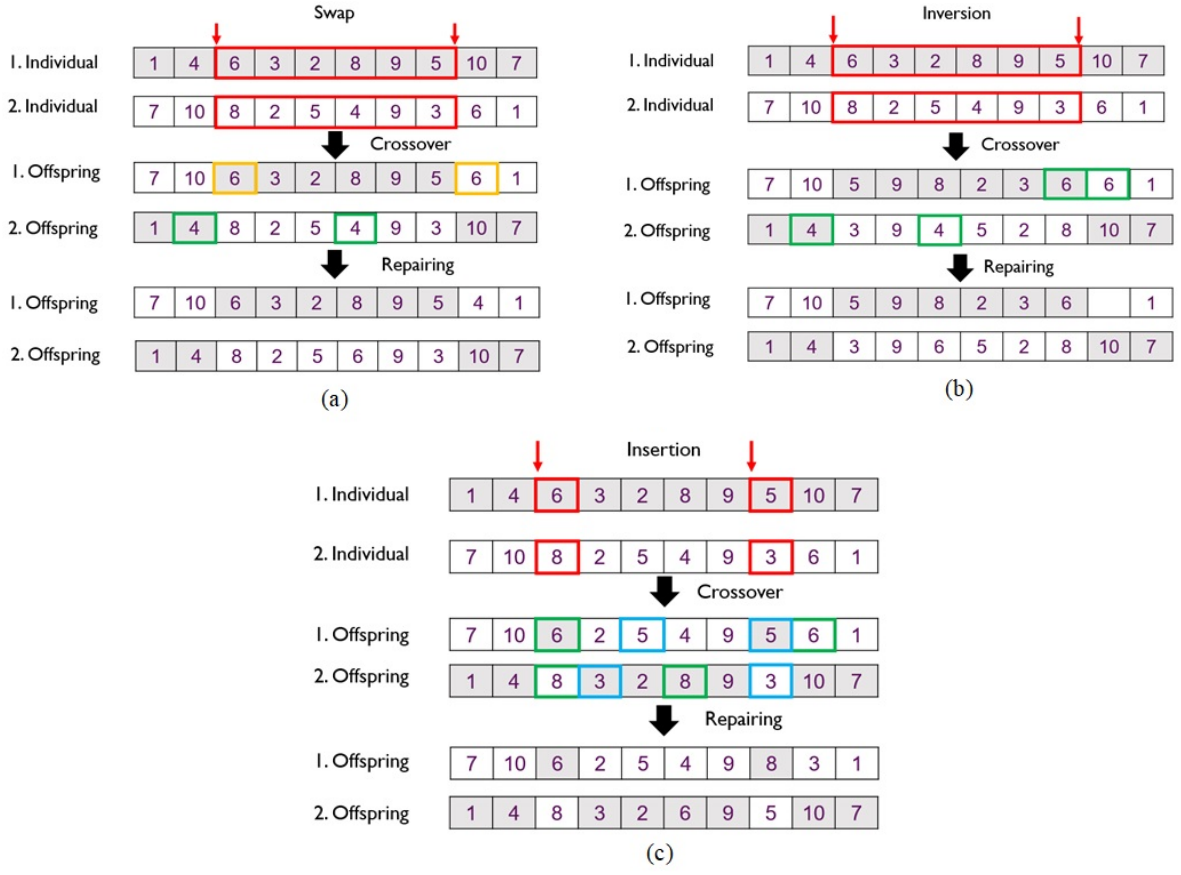


Fig. 4. Crossover operations (a) swap, (b) reversion, and (c) insertion

A random number is generated during the mutation phase. If this number is less than the specified mutation rate, the mutation is performed; otherwise, it is passed. While performing the mutation process, randomly selected chromosomes are mutated by changing two randomly selected genes with each other; it is ensured that these genes are not the same. After the selection process, the genes are replaced, and the mutation process is completed. Fig. 5 illustrates the mutation process. Finally, the population is sorted by fitness value in the offspring. If the number of individuals exceeds the maximum population number, individuals with low fitness values are deleted.

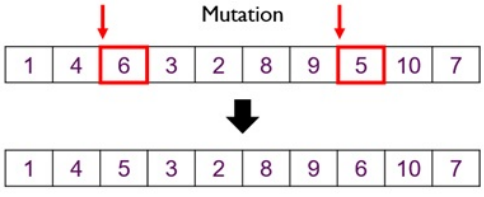


Fig. 5. Mutation Operation

3.3 Invasive Weed Optimization Algorithm

The Invasive Weed Optimization algorithm (IWO) is a metaheuristic algorithm created by Mehrabian and Lucas (Mehrabian & Lucas, 2006). The IWO is a simple yet effective optimization algorithm that simulates weeds, such as strength, adaptability,

and randomness. This algorithm effectively approaches the global optimum by applying weed characteristics (such as seed production, growth, and special competition) in a colony (Mehrabian & Lucas, 2006). The Invasive Weed Optimization algorithm is developed based on inspiration of invasive weeds in nature surviving despite challenging conditions. The algorithm is based on good weeds producing more seeds, while bad weeds produce fewer seeds (offspring). Thus, there is a mechanism that makes good weeds gradually get better and dominate the colony. From this perspective, this is similar to the genetic algorithm. Reasonable solutions allow us to produce more offspring, causing us to think that a better solution will be achieved quickly. That the IWO has fewer steps than the genetic algorithm has made us believe that the algorithm can be implemented more quickly and can work faster. The IWO algorithm has four necessary steps: initialization, reproduction, spatial dispersal, and competitive exclusion. Each step is described below.

- **Initialization**

The initial colony is created by randomly distributing a certain number of weeds in solution spaces.

- **Reproduction**

At this stage, the number of seeds that will be formed from weeds in the colony is determined. Better weeds produce more seeds, while worse weeds produce fewer seeds. The graph of the relationship between the fitness value and the seed number is presented in Figure 6. The number of seeds can be calculated using Eq. (2):

$$S_i = \left\lfloor \frac{f_i - f_{min}}{f_{max} - f_{min}} (S_{max} - S_{min}) \right\rfloor + S_{min} \tag{2}$$

where $\lfloor\ \rfloor$ denotes round down, and f_i is the fitness value of the i[th] weed. f_{max} and f_{min} represent the maximum and minimum fitness values of the colony, respectively. Moreover, S_{max} and S_{min} refer to the maximum and the minimum number of seeds a weed can produce, respectively. Finally, S_i denotes the number of seeds produced by the i[th] weed.

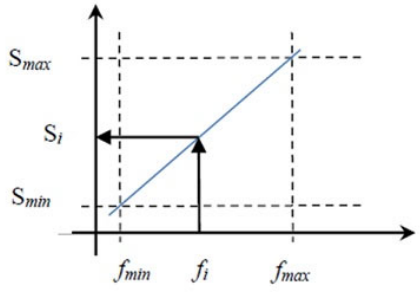


Fig. 6. Relationship Between Eligibility Value and Number of Seeds

- **Spatial Dispersal**

The specified number of seeds is randomly distributed in the problem space, and based on the variance relative to the average, these seeds are located near the primary weed. Here, the standard deviation (σ) of the function will start from a predetermined initial value ($\sigma_{initial}$) and be reduced throughout the iteration to a final value (σ_{final}). The σ_{iter} calculation method is provided in Eq. (3) (Koç, Nureddin, & Kahramanlı, 2018):

$$\sigma_{iter} = \frac{(iter_{max} - iter)^n}{(iter_{max})^n} \left(\sigma_{initial} - \sigma_{final}\right) + \sigma_{final} \tag{3}$$

where $iter_{max}$ denotes the maximum number of iteration (generation), σ_{iter} represents the standard deviation at the current iteration, and n denotes the variance reduction component.

- **Competitive Exclusion**

There is competition among plants to survive. Weeds in the colony overgrow, and all weeds are considered colonies. However, the total number of weeds in the colony should not exceed the colony's maximum weed value. For this reason, while more suitable weeds are included in the colony, less suitable weeds are removed from the colony (Koç et al., 2018). In this way, the mechanism by which the weak are eliminated and the strong survive is obtained. In solving CVRP with IWO, a weed is expressed as a customer list of routes. From this aspect, it can be said that this is the same as the genetic algorithm's chromosome structure. This process is performed as many times as the initial weed count. The weeds formed as a result of

this process are added to the colony, and thus, the starting colony is formed. The calculation of the weeds' fitness value in the colony and the creation of the routes are determined in the same way as in the genetic algorithm, as the total cost of the routes. The number of seeds that each weed in the colony can produce regarding the suitability degree is determined during the reproduction stage. In other words, in forming more seeds than useful herbs, fewer seeds than harmful herbs are created. The best weed of the colony produces as many seeds as the maximum number of seeds, while the worst weed produces as many seeds as the minimum number of seeds. It is ensured that the weeds are taken in order and that the seeds to be produced are created. While forming the seeds, it is desired to form them during the reproduction stage. In spatial dispersal, it is used to reach solutions around weeds, which is done by selecting two points of the weed and reversing it. The two selected points are found with the help of a random distribution formula. In this way, the solutions around the weed are reached. Figure 7 presents the production of a new seed from a weed at the stage of reproduction. Finally, in the competitive exclusion stage, the colony is ranked according to the fitness value. The number of weeds exceeding the maximum number of weeds must be offspring; therefore, weeds with low fitness values are destroyed.

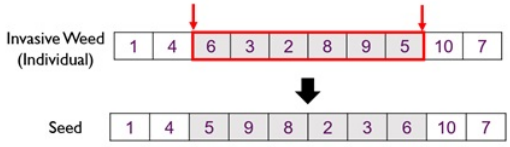


Fig. 7. Spatial dispersal operation

3.4 Hybrid Methods

The metaheuristic algorithms discussed in this study begin with random starting solutions, and it is attempted to gradually improve these solutions. When the stopping criterion is provided, an almost-optimal solution is produced. From this perspective, as the algorithm's initial solutions improve, the solution quality of the algorithm improves, as well. Considering this, various attempts have been made to feed the savings algorithm, genetic algorithm, and IWO in this study. The result produced by one algorithm has been added to the starting population of the other algorithm; for example, the savings algorithm's solution is run in a genetic algorithm and weed algorithm by adding it to the starting population. Three proposed hybrid heuristics are formed, as follows:

- H1: Savings results included genetic algorithm,
- H2: Savings results included IWO,
- H3: Savings first, a genetic algorithm with 50% of generation second, and IWO with 50% of generation third.

4. Results and Discussion

In this chapter, the parameters of methodologies are tuned using the Taguchi method. Thereby, the optimal user-defined parameters of the GA and IWO methods are found, as provided in Section 4.1. This study's material is selected from 24 different P group problem instances, 27 different A group problem instances, and 23 different B group problem instances (Augerat et al., 1995), widely used in the literature. In addition, five problem instances named CMT are selected from the study (Christofides, Mingozzi, & Toth, 1979). These data sets are capacitated vehicle routing problem instances. Each problem has a single depot and multiple vehicles, and the vehicles are homogeneous. The savings algorithm, genetic algorithm, weed algorithm, and hybrid algorithms were run with these data sets and compared the results from different studies in Section 4.2.

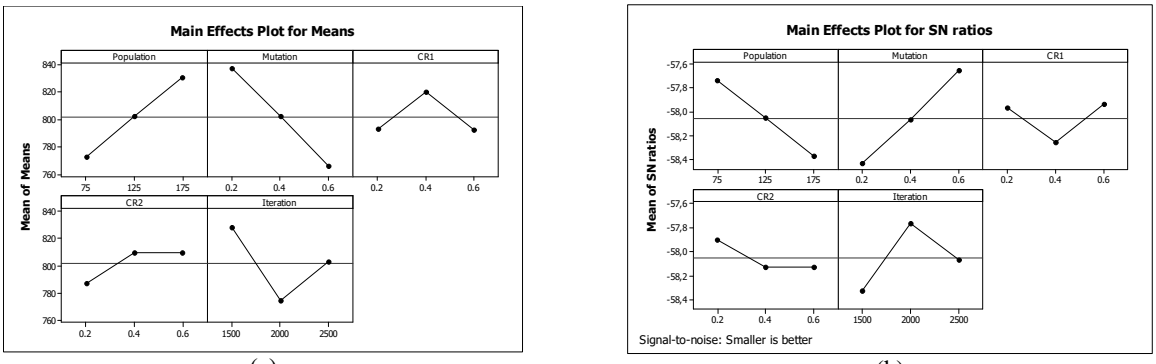
4.1. Parameter Tuning

In this study, to determine the best user-defined parameter values, the Taguchi experimental design approach is used. The Taguchi method, named by the pioneering engineer Genichi Taguchi, is an analysis tool based on experimental design and statistical calculations and is used to optimize factor level values (Shrestha and Manogharan, 2017). The factor level can be minimized or maximized considering the signal-to-noise (SN) values. In this study, Taguchi calculation of the S/N values in the "smaller is better" condition is given in Eq. (4) due to the cost minimization, where n denotes the number of observations, and y denotes the observed data.

$$S/N = -10\,log\,\frac{1}{n}\left(\sum_{i=1}^{n} y_i^2\right) \tag{4}$$

For the determination of experiments, a single-depot, 100-customer, and four-vehicle CVRP instance named P-n101-k4 is used. Each value to be tested with this medium-scale problem is run 100 times in the algorithm. As a result, the average cost values are calculated for 100 runs. According to the results, the most appropriate value is selected in terms of lower cost values. In the Taguchi experiment setting, the L27 design with three different levels of population size, mutation rate, P1, and P2 probabilities, and the number of generation parameters were applied to determine the best parameter values. The SN ratios for the main effects with the smaller-is-better condition are presented in Figure 8. The mutation rate and population size have the most significant parameters on the signal-to-noise ratio, while P_1 and P_2 appear to be insignificant. As a result of the S/N diagram, the parameter values of the genetic algorithm were as follows:

- Population size: 75
- Mutation rate: 0.6
- P_1: 0.6
- P_2: 0.2
- Number of generation: 2,000

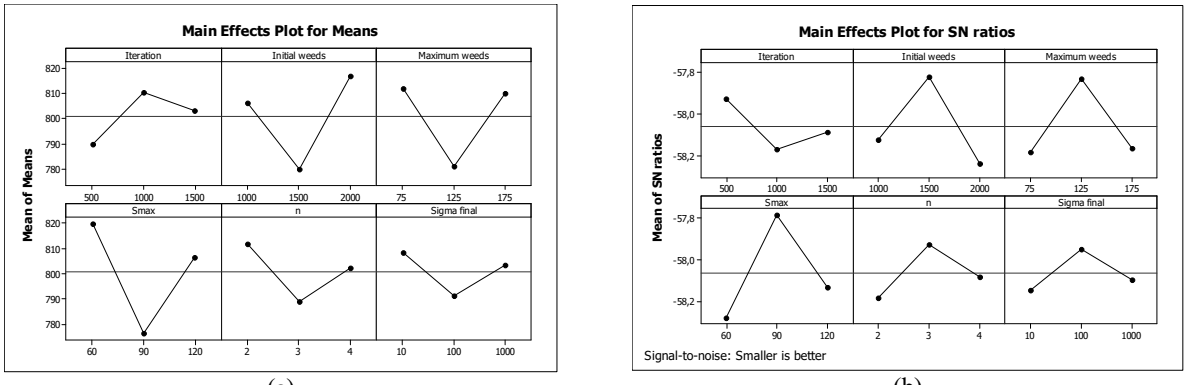


(a) (b)

Fig. 8. Taguchi results for parameter tuning of GA (a) main effects and (b) SN ratios

The Taguchi design is proposed for the IWO with the same problem as GA. The number of generations, initial number of weeds, maximum number of weeds, and S_{max}, n, and σ_{final} parameters with three levels are considered in the L27 design. The minimum number of seeds is considered to be 1 to give the worst individual of the colony an opportunity to produce seeds. The standard deviation initial value is accepted as 1 because it is the parameter that becomes meaningful compared with the standard deviation final value. The Taguchi results, presented in Figure 9, reveal that the number of initial weeds and maximum weeds and Smax parameters significantly affect the routing costs. The best parameter values of the IWO algorithm obtained from the SN ratio for the main effects diagram with the smaller-is-better condition were found, as follows:

- Number of generation: 500
- Initial number of weeds: 1,500
- Maximum number of weeds: 125
- S_{min}: 1
- S_{max}: 90
- n: 3
- $\sigma_{initial}$: 1
- σ_{final}: 100



(a) (b)

Fig. 9. Taguchi results for parameter tuning of IWO (a) main effects and (b) SN ratios

4.2. Results

Heuristic algorithms search the solution spaces randomly; therefore, the results might change from run to run. All the algorithms, save for the savings algorithm (a single-iteration method), are run 100 times for each problem instance. As a result of this process, performance criteria such as the minimum cost, average cost, maximum cost, standard deviation of cost, minimum speed, average speed, and maximum speed are selected. While evaluating the performance and quality of the algorithms, the performance criteria are calculated for each algorithm, since the cost, standard deviation, and speed are

essential. All the calculations are handled by the PC with Intel Core i7-7600U CPU 2.80 GHz, 16 GB RAM having 64 bit Windows 10 OS, and the Java Coding Platform.

4.2.1. Results for P problem instances

The results of P group problem instances are provided in Table 1. It can be said that among the savings algorithm, GA, and IWO, the IWO mostly produces the best results. The IWO adapted to the CVRP yields exceptionally competitive and efficient results. Moreover, it is concluded that the savings algorithm generally has the worst success; thus, the performance of both the GA and IWO is better than that of the savings algorithm. When all results are analyzed in general, it is seen that hybrid approaches yield better results than other algorithms. Hybrid approaches can produce results close to known reasonable solutions. The H1 approach has generally produced worse results than other hybrids, while the H2 approach and H3 approach are generally close to each other. However, the H3 approach is better than the H2 approach in 13 of the 24 problems. The addition of the savings algorithm to the GA's and the IWO's initial populations improved the algorithms' solutions and provided better results. From this perspective, it can be concluded that adding the solution of another algorithm to the initial populations of meta-heuristic algorithms improves the performance of the algorithm. When the standard deviation results in Table 1 are examined, the hybrid approaches' standard deviations are relatively lower. Therefore, the hybrid approaches are considered to be more stable than other algorithms. Additionally, the means of the standard deviations of hybrid approaches are close to one another. Among the hybrid approaches, the lowest standard deviation result was obtained from H2. As a result, the H2 approach is more stable than the other algorithms discussed. The standard deviation of the savings algorithm is not calculated, since it is a single iteration and provides the same solution in all runs. Table 2 displays that the average CPU times calculated as a result of running the algorithms 100 times for each problem are evaluated. Among the algorithms, the lowest running time—thereby the fastest algorithm—is the IWO. The results of the H2 approach are close to the IWO, as well. According to the results, it is seen that the CPU time of the algorithms depends on the problem size. As the problem size grew, the CPU times increased in direct proportion. Note that the savings algorithm is not included in Table 2 due to the CPU times being relatively equal to zero seconds for all instances.

Table 1
Comparison of the methods for P group test instances

Problem Instances	Total Costs							Standard Deviation				
	BKS*	Savings	GA	IWO	H1	H2	H3	GA	IWO	H1	H2	H3
P-n16-k8	450	482	450.22	450.42	450.55	450.85	450.05	1.21	1.63	2.40	2.60	0.50
P-n19-k2	212	226	224.56	223.13	218.75	212.03	215.14	9.59	9.05	1.13	0.30	3.15
P-n20-k2	216	222	226.12	224.70	217.98	219.62	217.98	8.61	8.41	0.20	2.05	0.20
P-n21-k2	211	224	221.32	218.51	213.75	214.42	213.42	9.41	8.19	4.76	5.63	4.56
P-n22-k2	216	232	226.56	224.07	217.93	219.12	219.15	9.26	8.92	4.14	5.46	5.05
P-n22-k8	590	590	595.71	598.83	590.00	590.00	590.00	8.94	11.17	0.00	0.00	0.00
P-n23-k8	529	537	536.74	538.64	530.95	535.16	531.19	7.09	8.90	2.87	3.18	3.06
P-n40-k5	458	561	521.24	503.19	499.60	497.02	496.15	22.75	19.58	10.48	11.18	9.48
P-n45-k5	510	676	583.07	551.67	533.46	532.73	530.17	28.73	20.90	13.03	4.47	12.90
P-n50-k7	554	693	626.53	610.10	590.43	590.61	588.73	25.92	21.39	6.42	5.07	6.59
P-n50-k8	629	716	703.17	683.58	670.57	670.47	668.61	24.72	16.39	3.20	2.08	3.74
P-n50-k10	696	775	774.72	758.27	726.44	715.94	718.14	27.08	27.26	8.36	7.21	8.45
P-n51-k10	741	825	833.27	824.22	771.52	767.70	769.32	28.02	33.35	9.76	7.60	9.44
P-n55-k7	568	702	647.10	629.66	596.20	591.27	594.17	29.94	21.37	5.16	4.31	4.93
P-n55-k8	588	722	651.41	630.02	617.70	612.11	615.24	27.78	22.05	9.04	7.54	9.68
P-n55-k10	694	792	763.49	747.44	725.48	722.77	724.30	20.51	19.16	3.89	4.36	4.14
P-n55-k15	945	1000	1020.65	1018.94	983.98	972.07	979.13	24.79	24.57	4.11	10.00	8.09
P-n60-k10	744	859	833.91	811.29	786.05	790.12	785.74	27.37	23.09	7.81	2.59	7.58
P-n60-k15	968	1020	1060.69	1058.29	1000.29	1008.13	1000.75	24.69	25.32	2.16	6.99	1.09
P-n65-k10	792	982	884.73	865.20	846.47	861.20	844.77	27.20	18.77	10.51	8.01	9.41
P-n70-k10	827	1047	959.49	939.20	905.23	906.92	903.69	36.48	38.02	13.71	10.86	13.63
P-n76-k4	593	930	706.76	673.43	662.95	650.35	655.26	32.45	26.56	16.76	13.47	15.61
P-n76-k5	627	989	739.00	709.47	699.72	693.00	696.79	37.94	30.36	10.35	13.13	10.44
P-n101-k4	681	1266	846.84	811.81	794.06	762.99	787.54	45.37	37.48	25.63	22.99	26.82
	*** Best Known Solution**					**% Difference with**		**3.96**	**3.49**	**1.29**	**1.21**	**1.33**

The proposed H2 and H3 approaches are compared with the previous works in the literature and the best-known solution (BKS). The algorithms find that the best result for the problem instance is presented in bold. Decreasing the difference value causes the approaches to obtain the best-known solution. Therefore, the concept of change demonstrates the quality of the algorithm's solution. A comparison of the best hybrids proposed in Table 3 with the algorithm results in the previous study is presented. According to the results, the proposed H3 approach quickly finds very similar results to BKS. Compared with other solution methods, the BKS was reached in 16 of the 23 test samples with GA (Ahmed and Sun, 2018), 6 of the 8 test samples with DHIWO (Zhao et al., 2016), and 17 of the 24 test samples with the proposed H3. However, since the standard deviations of the study are unknown, these values could not be compared. According to CPU times, the proposed H3 has a slight increase over others when the problem instances are complicated.

Table 2
CPU time (msec) comparisons of the algorithms for P group test instances

Problem Instances	GA	IWO	H1	H2	H3
P - n16 - k8	704	62	775	65	406
P - n19 - k2	710	73	712	74	407
P - n20 - k2	736	80	741	79	420
P - n21 - k2	752	83	755	83	420
P - n22 - k2	784	87	787	87	440
P - n22 - k8	799	83	816	81	461
P - n23 - k8	835	87	849	86	474
P - n40 - k5	1264	174	1274	177	714
P - n45 - k5	1377	206	1395	217	771
P - n50 - k7	1616	241	1631	246	940
P - n50 - k8	1588	246	1597	252	893
P - n50 - k10	1584	240	1595	243	885
P - n51 - k10	1598	245	1613	271	902
P - n55 - k7	1713	283	1722	295	954
P - n55 - k8	1701	284	1719	306	976
P - n55 - k10	1704	287	1721	295	950
P - n55 - k15	1747	259	1784	265	968
P - n60 - k10	1835	332	1839	341	1004
P - n60 - k15	1824	310	1840	313	1002
P - n65 - k10	1971	385	2062	391	1063
P - n70 - k10	2075	437	2248	446	1134
P - n76 - k4	2198	480	2265	481	1210
P - n76 - k5	2235	503	2421	509	1225
P - n101 - k4	2904	828	2989	835	1590

Table 3
Comparison of the results with previous works

Problem Instances	BKS	GA			DHIWO			H2			H3			
		Costs	% Diff.	CPU Time (msec)	Minimum Costs	% Diff.	Mean Costs	Minimum Costs	% Diff.	CPU Time (msec)	Mean Costs	Minimum Costs	% Diff.	CPU Time (msec)
P-n16-k8	450	**450**	0.00	110	**422**	-6.22	450.85	**450**	0.00	65	450.05	**450**	0.00	406
P-n19-k2	212	**212**	0.00	100	**	-	212.03	**212**	0.00	74	215.14	**212**	0.00	407
P-n20-k2	216	**216**	0.00	350	**216**	0.00	219.62	**216**	0.00	79	217.98	**216**	0.00	420
P-n21-k2	211	**211**	0.00	320	**	-	214.42	**211**	0.00	83	213.42	**211**	0.00	420
P-n22-k2	216	**216**	0.00	710	**216**	0.00	219.12	**216**	0.00	87	219.15	**216**	0.00	440
P-n22-k8	590	603	2.20	830	**589**	-2.35	590.00	**590**	0.00	81	590.00	**590**	0.00	461
P-n23-k8	529	**529**	0.00	1020	**	-	535.16	**529**	0.00	86	531.19	**529**	0.00	474
P-n40-k5	458	**458**	0.00	1330	**	-	497.02	470	2.62	177	496.15	468	2.18	714
P-n45-k5	510	**510**	0.00	1450	**	-	532.73	526	3.14	217	530.17	**510**	0.00	771
P-n50-k7	554	**554**	0.00	1480	**	-	590.61	573	3.43	246	588.73	567	2.35	940
P-n50-k8	629	631	0.32	1050	**	-	670.47	662	5.25	252	668.61	658	4.61	893
P-n50-k10	696	**696**	0.00	2230	**696**	0.00	715.94	711	2.16	243	718.14	**696**	0.00	885
P-n51-k10	741	**741**	0.00	3380	**	-	767.70	754	1.75	271	769.32	**741**	0.00	902
P-n55-k7	568	**568**	0.00	4320	**	-	591.27	583	2.64	295	594.17	**568**	0.00	954
P-n55-k8	588	**	-	-	**	-	612.11	**588**	0.00	306	615.24	**588**	0.00	976
P-n55-k10	694	**694**	0.00	4940	**	-	722.77	709	2.16	295	724.30	**694**	0.00	950
P-n55-k15	945	989	4.66	4290	955	1.06	972.07	958	1.38	265	979.13	958	1.38	968
P-n60-k10	744	**744**	0.00	5830	**	-	790.12	782	5.11	341	785.74	769	3.36	1004
P-n60-k15	968	**968**	0.00	5370	**	-	1008.13	993	2.58	313	1000.75	**968**	0.00	1002
P-n65-k10	792	**792**	0.00	6440	**	-	861.20	822	3.79	391	844.77	823	3.91	1063
P-n70-k10	827	833	0.73	9240	**	-	906.92	880	6.41	446	903.69	873	5.56	1134
P-n76-k4	593	598	0.84	16110	595	0.34	650.35	611	3.04	481	655.26	**593**	0.00	1210
P-n76-k5	627	636	1.44	15850	**	-	693.00	664	5.90	509	696.79	**627**	0.00	1225
P-n101-k4	681	692	1.62	20170	**681**	0.00	762.99	719	5.58	835	787.54	**681**	0.00	1590

$$\%\, Difference = \frac{(Costs - BKS)}{BKS} * 100$$

** are not reported

4.2.2. Results for CMT problem instances

In Table 4, the mean costs calculated as a result of running the algorithms 100 times for each problem of the CMT group are evaluated. When the results are analyzed, the savings algorithm exhibits the least amount of success in most problem instances.

For all problem instances, hybrid approaches produced better results. Although the performance of hybrid approaches is close to one another, it can be said that the H2 approach produces the best results. The H1 and H2 algorithms increase the performance of the GA and IWO, respectively. Moreover, the IWO obtains better results compared with the GA and savings algorithms. It can be said that the invasive weed optimization algorithm, which is a meta-heuristic algorithm adapted to the capacitated vehicle routing problem, produces highly competitive and efficient results for CMT group data, as well. In general, all the hybrid approaches have similar results and outperformed the savings, GA, and IWO singularly.

Table 4
Comparison of the methods for CMT group test instances

Problem Instances	Total Costs							Standard Deviation				
	BKS*	Savings	GA	IWO	H1	H2	H3	GA	IWO	H1	H2	H3
CMT1(50)	524.61	731	612.84	582.72	577.64	567.56	575.33	35.8	27.4	12.2	5.50	12.8
CMT2(75)	835.26	1053	971.31	962.99	906.84	894.69	904.83	35.3	40.4	14.5	10.1	13.9
CMT3(100)	826.14	1321	1012.0	980.74	935.34	925.96	928.23	56.9	57.7	16.6	15.8	20.3
CMT4(150)	1028.4	1382	1334.3	1301.4	1181.2	1149.2	1191.4	62.3	86.8	27.2	20.2	27.5
CMT5(199)	1291.3	1722	1745.7	1709.5	1498.2	1480.5	1502.7	70.8	94.2	30.6	29.5	26.5
% Difference with BKS		**38.61**	**24.11**	**20.80**	**12.56**	**10.76**	**12.52**	**5.90**	**6.56**	**2.22**	**1.68**	**2.26**

*** Best Known Solution**

Table 4 also presents the standard deviation values calculated as a result of running algorithms 100 times for each problem instance of the CMT group. The standard deviation values of the savings algorithm are 0 for each problem. A standard deviation values comparison is made among the other algorithms. Here, the standard deviation of the GA and IWO is higher than that of the hybrid approaches. When the results are examined, it could be said that the standard deviations of the hybrid approaches are better than the others. In other words, it can be interpreted that when the initial solutions are fed with another algorithm, the standard deviation decreases, and the algorithm's stability increases. The H2 approach has the lowest standard deviation in 4 of the 5 CMT problems. The lowest value belongs to the hybrid approach consisting of the savings and weed algorithms. As a result, it can be said that the H2 algorithm is more stable than the other algorithms discussed. Table 5 presents the evaluation of the calculated CPU times from running the algorithms 100 times for each problem in the CMT group. Among the algorithms, the lowest running time—thereby the fastest algorithm—is the savings algorithm. It does not appear possible to select the fastest among the algorithms other than the savings algorithm. The CPU time of the IWO is low for medium-sized problems, but it was unable to perform as well in large-scale problems. For 199 customer problems within the CMT group, the best CPU time belongs to the H3 approach. The hybridization of the genetic algorithm and the weed algorithm with the savings algorithm separately affected the CPU times. Since the savings algorithm is low in terms of operating time, it can be interpreted that hybridization does not overextend the CPU times.

Table 5
CPU time (msec) comparisons of the algorithms for CMT group test instances

Problem Instances	Savings	GA	IWO	H1	H2	H3
CMT1(50)	0	1576	252	1622	292	845
CMT2(75)	1	2092	505	2173	525	1109
CMT3(100)	1	2783	857	2883	965	1494
CMT4(150)	3	3945	2220	4231	2439	2197
CMT5(199)	0	5403	3945	5757	3988	3083

4.2.3. Results for A problem instances

In the A problem instances, nodes vary from 32 to 80, and the number of vehicles varies from 5 to 10. When the A problem instances are examined, it is seen that the H3 algorithm reaches the BKS in a short CPU time or obtains results very close to the BKS. It is seen that the H3 algorithm reaches the BKS in 13 of 27 test samples. Additionally, it is seen that the DHIWO algorithm (Zhao et al., 2016) reaches the BKS in 7 of the 8 test samples. When the two algorithms are compared, it is seen that H3 reaches the BKS in all instances where DHIWO is applied (See Table 6).

4.2.4 Results for B problem instances

In the B problem instances, nodes vary from 31 to 78, and the number of vehicles varies from 5 to 10. When the B problem instances are examined, it is seen that the H3 algorithm reaches the BKS in a short CPU time or obtains results very close to the BKS, similar to other problem instances. It is seen that the H3 algorithm reaches the BKS in 15 of the 23 test samples. Additionally, it is seen that the DHIWO algorithm (Zhao et al., 2016) reaches the BKS in 5 of the 8 test samples. When the two algorithms are compared, it is seen that H3 reaches the BKS in all instances where DHIWO is applied (See Table 7).

Table 6
Comparison of the results with previous works for problem instances A

Problem Instances	Total Costs						CPU Time (msec)		
	BKS*	DHIWO Min (Zhao et al.,2016)	H3 Min	H3 Mean	H3 Max	Standard Deviation of H3	Min	Mean	Max
A-n32-k5	784	**784**	**784**	848.66	892	10.77	500	523	833
A-n33-k5	661	**661**	**661**	690.06	699	9.73	497	513	532
A-n33-k6	742	**	**742**	773.66	781	5.45	491	506	530
A-n34-k5	778	**778**	**778**	797.49	824	12.22	506	519	543
A-n36-k5	799	**	**799**	824.26	878	11.75	534	550	592
A-n37-k5	669	**	693	737.24	814	17.83	538	555	575
A-n37-k6	949	**	967	982.14	984	4.28	559	572	600
A-n38-k5	730	**	**730**	772.62	823	22.29	559	587	643
A-n39-k5	822	**	853	855.2	865	2.71	549	564	594
A-n39-k6	831	**831**	**831**	856.94	903	16.03	558	577	638
A-n44-k6	937	**937**	**937**	1006.9	1029	11.18	596	609	652
A-n45-k6	944	**	977	986.17	1015	8.83	611	627	651
A-n45-k7	1146	**	1155	1189.37	1215	8.40	636	649	680
A-n46-k7	914	**914**	**914**	925.29	935	4.57	630	646	683
A-n48-k7	1073	**	1115	1134.33	1180	14.65	660	671	720
A-n53-k7	1010	**	**1010**	1076.06	1183	23.64	761	782	827
A-n54-k7	1167	**	**1167**	1188.54	1227	11.45	753	772	829
A-n55-k9	1073	**	1085	1121.61	1154	16.99	777	797	855
A-n60-k9	1354	**1354**	**1354**	1389.42	1438	12.18	841	856	904
A-n61-k9	1034	**	1048	1073.54	1109	13.46	850	867	919
A-n62-k8	1288	**	1349	1382.91	1422	15.84	850	872	944
A-n63-k9	1616	**	1645	1664	1694	8.56	865	882	927
A-n63-k10	1314	**	1351	1405.31	1509	28.87	874	896	1004
A-n64-k9	1401	**	1472	1490.9	1514	10.03	885	902	969
A-n65-k9	1174	**	1202	1272.95	1402	38.96	885	909	968
A-n69-k9	1159	**	1206	1262.73	1342	25.35	934	964	1057
A-n80-k10	1763	1764	**1763**	1885.67	1924	10.62	1067	1089	1148

% Difference with BKS
*** Best Known Solution**

Table 7
Comparison of the results with previous works for problem instances B

Problem Instances	Total Costs						CPU Time (msec)		
	BKS*	DHIWO Min (Zhao et al.,2016)	H3 Min	H3 Mean	H3 Max	Standard Deviation of H3	Min	Mean	Max
B-n31-k5	672	**672**	**672**	703.72	729	6.77	483	503	811
B-n34-k5	788	**788**	**788**	821.64	878	16.18	510	529	579
B-n35-k5	955	**	**955**	971.4	1054	19.75	517	533	585
B-n38-k6	805	**	810	821.78	837	5.04	551	564	589
B-n39-k5	549	**	**549**	616.19	635	20.92	566	589	630
B-n41-k6	829	**829**	**829**	866.01	928	22.16	584	603	643
B-n43-k6	742	**	**742**	764.53	852	20.09	618	637	689
B-n44-k7	909	**	926	931.83	936	3.69	616	628	657
B-n45-k5	751	**751**	**751**	769.89	785	7.58	631	645	679
B-n45-k6	678	**	690	730.98	795	22.65	619	636	669
B-n50-k7	741	**741**	**741**	757.21	801	10.58	707	725	774
B-n50-k8	1312	**	1340	1371.06	1378	10.03	705	724	777
B-n51-k7	1032	**	**1032**	1048.85	1088	14.05	735	749	813
B-n52-k7	747	**	**747**	758.36	774	5.17	734	754	805
B-n56-k7	707	**	**707**	757.26	787	17.76	787	809	881
B-n57-k7	1153	**	**1153**	1192.33	1323	36.52	795	815	907
B-n57-k9	1598	**	1636	1662.4	1724	18.94	773	795	854
B-n63-k10	1496	1497	**1496**	1598.3	1659	19.31	850	865	905
B-n64-k9	861	**	890	933.65	994	24.66	853	875	966
B-n66-k9	1316	**	1344	1401.48	1476	26.77	885	923	1042
B-n67-k10	1032	1035	**1032**	1092.4	1129	11.74	890	906	955
B-n68-k9	1272	**	1292	1304.88	1316	8.29	935	958	1014
B-n78-k10	1221	1223	**1221**	1301.48	1352	19.35	1002	1030	1116

% Difference with BKS
*** Best Known Solution**

5. Conclusion

Capacitated vehicle routing problems are among the NP-Hard class of problems, which are difficult to solve exactly. For this reason, heuristic approaches have often been suggested for this problem in previous studies. In this study, the problem is solved using different heuristic approaches in this context. The first approach is a savings algorithm, which has higher cost routes compared with other approaches. Additionally, when the solution time is examined, it is seen that it is a reasonable method, since there is a single-iteration solution. The second approach is a genetic algorithm, which is frequently used in vehicle routing problems. Here, it can be seen that the results obtained are relatively better than in the savings algorithm. However, when the speed of the algorithm is examined, it is seen that the genetic algorithm finds a solution over a long time compared with other approaches. Therefore, the invasive weed optimization algorithm is developed for the problem. It has been seen in previous studies that the IWO algorithm obtains good results for problems in the continuous solution space, and it is adapted in the traveling salesman problem, as well. The IWO provides better costs in a shorter time than the other algorithms; thus, the IWO produces extraordinarily competitive and efficient results very quickly. The hybrid versions of these algorithms are compared, as well. For this purpose, three different hybrid heuristics that include the savings, genetic and invasive weed optimization algorithms, named H1, H2, and H3, are proposed. While H1 improves the solutions' costs, the solution times are still quite long due to the genetic algorithm; meanwhile, H2 yields an excellent cost and solution time. With this approach, reasonable costs have been achieved with very low standard deviations and acceptable solution times. The final approach is the H3 approach, which contains all algorithms and achieves reasonable solution costs with very low standard deviations and acceptable solution times. It is concluded that hybridizing the algorithms provides better costs and solution times for test problem instances. Parameter selection is essential for the IWO algorithm's solution quality adapted to the vehicle routing problem. The use of different approaches when producing seeds during the reproduction stage is another factor that might affect the quality of the solution. The algorithm's solution quality can be increased if detailed studies are conducted to select appropriate parameters in future studies and different approaches during the reproduction stage. It can investigate the performance of the IWO in large-scale problems.

Conflicts of interest

The authors declare that they have no conflicts of interest.

References

Ahmed, A., & Sun, J. U. (2018). Bilayer local search enhanced particle swarm optimization for the capacitated vehicle routing problem. *Algorithms, 11*, 31.

Arnold, F., Gendreau, M., & Sörensen, K. (2019). Efficiently solving very large-scale routing problems. *Computers & operations research, 107*, 32-42.

Augerat, P., Belenguer, J. M., Benavent, E., Corberán, A., Naddef, D., & Rinaldi, G. (1995). *Computational results with a branch and cut code for the capacitated vehicle routing problem* (Vol. 34): IMAG.

Benrahou, F., & Tairi, A. (2019). Capacitated Vehicle Routing Problem for Collection Waste Lube Oil in Algiers. *Fresenius Environ. Bull, 28*, 4500-4505.

Boyzer, Z., Alkan, A., & Fığlalı, A. (2014). Cluster-first, then-route based heuristic algorithm for the solution of capacitated vehicle routing problem. *International Journal of Informatics Technologies, 7*, 29-37.

Cheng, R., Gen, M., & Tsujimura, Y. (1999). A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies. *Computers & Industrial Engineering, 36*, 343-364.

Christofides, N., Mingozzi, A., & Toth, P. (1979). The vehicle routing problem. In A. M. N. Christofides, P. Toth, C. Sandi (Eds.) (Ed.), *Combinatorial Optimization* (pp. 315-338). Chichester, UK: Wiley.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research, 12*, 568-581.

Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science, 6*, 80-91.

Davis, L. (1991). *Handbook of genetic algorithms,* (Vol. 115). New York: Van Nostrand Reinhold.

Jahangir, H., Mohammadi, M., Pasandideh, S. H. R., & Nobari, N. Z. (2019). Comparing performance of genetic and discrete invasive weed optimization algorithms for solving the inventory routing problem with an incremental delivery. *Journal of Intelligent Manufacturing, 30*, 2327-2353.

Jang, J.-S. R., Sun, C.-T., & Mizutani, E. (1997). Neuro-fuzzy and soft computing-a computational approach to learning and machine intelligence [Book Review]. *IEEE Transactions on automatic control, 42*, 1482-1484.

Koç, İ., Nureddin, R., & Kahramanlı, H. (2018). Implementation of GSA (Gravitation Search Algorithm) and IWO (Invasive Weed Optimization) for the prediction of the energy demand in Turkey using linear form. *Selcuk University Journal of Engineering Science and Techology, 6*, 529-543.

Mehrabian, A. R., & Lucas, C. (2006). A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics, 1*, 355-366.

Mohammadi, M., Razmi, J., & Tavakkoli-Moghaddam, R. (2013). Multi-Objective Invasive Weed Optimization For Stochastic Green Hub Location Routing Problem With Simultaneous Pick-Ups And Deliveries. *Economic Computation & Economic Cybernetics Studies & Research, 47*.

Mulloorakam, A. T., & Nidhiry, N. M. (2019). Combined objective optimization for vehicle routing using genetic algorithm. *Materials Today: Proceedings, 11*, 891-902.

Nabiyev, V. V. (2005). Artificial Intelligence: Problems, Methods, Algorithms. *Seckin Pub. Co., Ankara*.

Nath, S., Chakravarty, A. K., Ghosh, S., & Sarkar, S. K. (2017). Invasive weed optimization approach to VLSI routing. In *2017 Devices for Integrated Circuit (DevIC)* (pp. 615-619): IEEE.

Normasari, N. M. E., Yu, V. F., & Bachtiyar, C. (2019). A simulated annealing heuristic for the capacitated green vehicle routing problem. *Mathematical Problems in Engineering, 2019*.

Pahlavani, P., Delavar, M. R., & Frank, A. U. (2012). Using a modified invasive weed optimization algorithm for a personalized urban multi-criteria path optimization problem. *International Journal of Applied Earth Observation and Geoinformation, 18*, 313-328.

Pichpibul, T., & Kawtummachai, R. (2012). An improved Clarke and Wright savings algorithm for the capacitated vehicle routing problem. *ScienceAsia, 38*, 307-318.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & operations research, 34*, 2403-2435.

Rojas-Cuevas, I.-D., Caballero-Morales, S.-O., Martinez-Flores, J.-L., & Mendoza-Vazquez, J.-R. (2018). Capacitated vehicle routing problem model for carriers. *Journal of Transport and Supply Chain Management, 12*, 1-9.

Shrestha, S., & Manogharan, G. (2017). Optimization of binder jetting using Taguchi method. Jom, 69(3), 491-497.

Sur, C., & Shukla, A. (2013). Discrete invasive weed optimization algorithm for graph based combinatorial road network management problem. In *2013 International Symposium on Computational and Business Intelligence* (pp. 254-257): IEEE.

Tarantilis, C. D., Ioannou, G., & Prastacos, G. (2005). Advanced vehicle routing algorithms for complex operations management problems. *Journal of Food Engineering, 70*, 455-471.

Toffolo, T. A., Vidal, T., & Wauters, T. (2019). Heuristics for vehicle routing problems: Sequence or set optimization? *Computers & operations research, 105*, 118-131.

Wang, L., & Lu, J. (2019). A memetic algorithm with competition for the capacitated green vehicle routing problem. *IEEE/CAA Journal of Automatica Sinica, 6*, 516-526.

Wedyan, A. F., & Narayanan, A. (2014). Solving capacitated vehicle routing problem using intelligent water drops algorithm. In *2014 10th International Conference on Natural Computation (ICNC)* (pp. 469-474): IEEE.

Yücenur, G. N., & Demirel, N. Ç. (2011). A hybrid algorithm with genetic algorithm and ant colony optimization for solving multi-depot vehicle routing problems. *Sigma Journal of Engineering and Natural Sciences, 29*, 340-350.

Zhang, S., Gajpal, Y., & Appadoo, S. (2018). A meta-heuristic for capacitated green vehicle routing problem. *Annals of Operations Research, 269*, 753-771.

Zhao, Y., Leng, L., Qian, Z., & Wang, W. (2016). A discrete hybrid invasive weed optimization algorithm for the capacitated vehicle routing problem. *Procedia Computer Science, 91*, 978-987.