IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON COMMUNICATION AND FOG/EDGE COMPUTING TOWARDS INTELLIGENT CONNECTED VEHICLES (ICVS)

Received October 26, 2019, accepted November 29, 2019, date of publication December 5, 2019,
date of current version December 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2957722

An Ameliorative Hybrid Algorithm for Solving the Capacitated Vehicle Routing Problem

ALI ASGHAR RAHMANI HOSSEINABADI[1], ADAM SLOWIK[2], (Senior Member, IEEE), MEHDI SADEGHILALIMI[3], MOHAMMAD FAROKHZAD[4], MORTEZA BABAZADEH SHAREH[5], AND ARUN KUMAR SANGAIAH[6]

[1]Young Researchers and Elite Club, Ayatollah Amoli Branch, Islamic Azad University, Amol 4865116915, Iran
[2]Department of Electronics and Computer Science, Koszalin University of Technology, 75-453 Koszalin, Poland
[3]Department of IT and Computer Engineering, Qazvin Branch, Qazvin Islamic Azad University, Qazvin 15195-34199, Iran
[4]Department of Electronics, Iran University of Science and Technology, Tehran 16846-13114, Iran
[5]Department of Computer Engineering, Babol Branch, Islamic Azad University, Babol 4865116915, Iran
[6]School of Computing Science and Engineering, Vellore Institute of Technology (VIT), Vellore 632014, India

Corresponding author: Adam Slowik (aslowik@ie.tu.koszalin.pl)

ABSTRACT One of the most applicable versions of the Vehicle Routing Problem (VRP) which has been widely studied in logistic services is Capacitated Vehicle Routing Problem (CVRP). There are many algorithms to solve the CVRP to minimize total travelled distance. Some of the most recent and efficient metaheuristic algorithms are capable of generating solutions within 0.5% to 1% gap from the optimum for instance problems adopted from the literature considering hundreds or thousands of demand points. In this contribution, a novel hybrid algorithm is proposed based on Gravitational Emulation Local Search (GELS) and Genetic Algorithm (GA). This algorithm alleviates the weaknesses of the GELS algorithm. The performance of the proposed algorithm, which is called GELSGA, is compared with other meta-heuristics. The obtained results show that the proposed algorithm can compete vigorously with them. In addition, the proposed algorithm could obtain solutions close to the Best Known Solutions (BKS) for many instance problems.

INDEX TERMS Capacitated vehicle routing problem, hybrid algorithm, genetic algorithm, gravitational emulation local search, optimization.

I. INTRODUCTION

Transportation enjoys a special status in the production and services sectors of economic systems and is a major contributor to the gross domestic product of all countries. Therefore, the questions of improving routes, eliminating unnecessary travel, and constructing optimal routes have attracted interest. Vehicle Routing Problem (VRP) has an important role in reducing total transportation cost of distribution logistics and is deemed to be as one of the most significant combinatorial optimization problems. Transportation and distribution are frequently modeled as VRP. VRP involves finding a set of optimal routes for each vehicle for delivering services to customers from a central depot with the minimum cost [1], [2]. VRP is one of the most widely reviewed issues in logistic systems [3], which has been first presented by Dantzig and Ramser [10] as a generalization of the Traveling Salesman Problem (TSP) introduced by Flood [5]. On the other hand, there is a requirement for all vehicles that they can traverse nodes just once and then return to the depot (destination node). Many clustering approaches have been studied for TSP to solve complicated problem instances and reduce computational time [6]. In fact, clustering approaches have many different applications besides routing problems [7], such as medical diagnosis [8], and social network analysis [9]. All of these methods apply different learning algorithms for grouping similar data. In a comprehensive review of different versions for VRP, the Capacitated Vehicle Routing Problem (CVRP) occupies an important position. This problem is defined in the following [10]. Input data includes $n + 1$ nodes, a unique depot, and n customers; a $(n + 1) \times (n + 1)$ matrix $d = [d_{ij}]$ denotes the distances between nodes i and j; an n-dimensional demand vector $q = [q_i]$ that explains the volume of products needs to be delivered to the customers

The associate editor coordinating the review of this manuscript and approving it for publication was Lei Shu.

by observing the limitation of the capacity of vehicles up to Q. A feasible solution consists of a set of tours that begins from a depot and ends at the depot. In this solution, each vehicle constructs a single tour and all customers are served just once such that the total demand of the served customers is not greater than the capacity of the respective vehicle. The objective of the problem is to find a feasible solution with a minimum total distance traveled. Some authors consider the number of routes as a constant value K (almost always defined as a minimum number of possible routes K_{min}). CVRP has a significant application in developing different exact and heuristic solution methods for VRP. It provides an empirical test substrate for presenting novel ideas as the most basic version. Its relative simplicity allows for clear descriptions and implementation without considering unnecessary contexts. As an example, the heuristic method introduced for CVRP by Clarke and Wright [4] has been developed for VRP with Time Windows (VRPTW) [11] and the other different versions [12]. Optimization methods and algorithms are divided into two categories: exact algorithms and approximate algorithms [13]. Optimization methods are intelligent algorithms used for finding the optimal solution. These algorithms provide strategies to escape from local optimum and can be applied to a wide range of problems. There are various optimization algorithms which have been used to solve different problems in recent years. Some of these algorithms are as follows: Genetic Algorithm (GA) [14], Ant Colony Optimization (ACO) [15], Bee Colony (BC) [16], Particle Swarm Optimization (PSO) [17], Tabu Search (TS) [18], Simulated Annealing (SA) [19] and so on. These algorithms can be applied to different routing problems [20]. Recently a new optimization algorithm named Gravitational Emulation Local Search Algorithm (GELS) was developed by Barry Webster in 2004 which was inspired by Newton's law for optimization [21]. GELS is defined based on the randomization concepts considering two different parameters of *velocity* and *force* which use random numbers of local search algorithms to avoid local optimums. The main idea of GELS is adopted from the principle of gravitational force that causes objects to attract each other so that a heavier object has more gravitational force for applying on other objects and attracting lightweight objects [21]. So far GELS algorithm has been used in various problems such as Grids Computing [22]–[26], Job Shop [27], Open Shop [28], [29], Flexible Manufacturing System [30], Cloud Computing [31], VRP and TSP [32]–[34], and could achieve relatively good solutions for these problems. Since GA is strong at global search in problem space but is weak in terms of stability and local search, we decided to propose a new algorithm by combining the global search capability of GA and local search capability of GELS algorithm. The proposed algorithm could obtain better solutions than those of compared algorithms. The remainder of the paper is categorized as follows. Section II presents the literature review. Section III introduces the proposed algorithms i.e. GELS and GA. Section IV explains the numerical results. Finally, Section V provides the conclusion.

II. LITERATURE REVIEW

Researchers have introduced various heuristic- and meta-heuristic-based techniques to solve the CVRP in recent years, a few of which are reviewed below. In 2004, Mazzeo and Loiseau expanded an ACO introduced by Maniezzo, Dorigo, and Colorni to solve the CVRP based on meta-heuristic techniques, and showed their proposed algorithm could compete with other meta-heuristic methods [35]. Mester and Braysy proposed an active guided evolution strategy in 2005 to solve the large-scale CVRP. They conducted computational tests on a set of 76 test problems and found their proposed method was highly competitive and gave the best solutions for 70 of the tested problems [36]. Lin *et al.* [37] presented a hybrid meta-heuristic algorithm to solve the CVRP. They employed a Hybrid Simulated Annealing (HSA) and Local Search Algorithm (LSA) to solve the CVRP that combined the advantages of Simulated Annealing or SA and local search and reported results they found for 14 classic instances and 20 large sized instances. According to simulation results, the proposed algorithm found 8 better solutions for the 14 classic samples and proved it could compete with other available algorithms in solving the CVRP. In 2010, Chen *et al.* [38] introduced an iterative reduced Variable Neighborhood Search (VNS) algorithm based on multi operators optimization for solving the CVRP. They designed a perturbation strategy with a cross-exchange operator in order to provide the possibility of escape from local minima and tested the performance of the algorithm for 34 standard CVRP benchmarks. Results indicated the proposed algorithm worked well and could compete with other heuristic methods. In [39], an HSA and ACO were proposed, which combined the advantages of both algorithms in order to solve the CVRP. Results showed that the proposed algorithm could find the solution for the CVRP. A GA was employed for solving the CVRP [40] and computational results indicated that the GA could find the optimal routes for each vehicle by considering the capacity and total travel time constraints. Stanojević *et al.* [41] proposed enhanced savings calculation and its application for solving the CVRP and presented a novel methodology for combining routes and solutions to develop a formula for calculating these savings. They used an advanced combination for developing a new expanded savings algorithm as a heuristic one that recalculated savings in each iteration. Computational results showed that, the expanded savings algorithm yielded better solutions in comparison with the original savings algorithm. Duhamel *et al.* [42] introduced a multiple starts evolutionary LSA for solving the two-dimensional loading CVRP. They suggested an extension of the CVRP in which customer demands consisted of two-dimensional weighted items. Their goal was to design a set of routes to minimize transportation costs using a fleet of homogeneous vehicles based on a depot node. The two-dimensional orthogonal packing limitations of the items had to be considered for each vehicle. Numerical tests indicated the proposed approach was superior to previous methods. Tlili *et al.* [43] introduced a hybrid meta-heuristic algorithm

for the CVRP with distance restrictions by extracting suitable vehicle routes with the purpose of minimizing the distance traveled by each vehicle. They formulated the proposed problem as a mixed-integer linear programming (MILP) model and proposed a swarm-based hybrid meta-heuristic method that combined a variable neighborhood search algorithm with a PSO algorithm. Results on standard samples indicated the proposed algorithm could compete with the state of the art methods. Cardoso *et al.* [44] introduced a real-time random solution for the CVRP with time windows. They suggested an advanced optimization system combined with the present Enterprise Resource Planning (ERP) without bringing forth any significant disruptions in the current distribution process of a company. The proposed system contains a route optimization module, a module for communicating within and outside the system, a non-communication database for providing local information warehouse regarding the optimization method, and a map dependent subsystem. Jin *et al.* [45] a cooperative parallel meta-heuristic algorithm is proposed for the CVRP which consists of several Parallel Tabu Search (PTS) algorithms cooperating by non-simultaneously exchanging best solutions through an existed solutions pool. Then, these solutions are clustered based on their resemblances. Finally, the information of the solution clusters search history is used to help the diversification of the PTS algorithms. Vidal *et al.* [46] developed a component-based heuristic algorithm for the multi-attribute VRP which depends on different approaches. Their extensive computational experiments demonstrate the remarkable performance of the proposed method. Recent CVRP showed that many of them use evolution algorithms to solve the CVRP problems without considering their inherent characteristics. Some of them have used the combination of an evolutionary algorithm and a heuristic one. However, none of these studies have paid attention to the inherent characteristics of evolutionary algorithms. In this paper, we tried to utilize the strengths of the evolution algorithms and combination of two algorithms in order to provide a method that best suited for the search in solution space of CVRP problem. Due to the high capability of GA in exploration phase and high capability of GELS algorithm in exploitation phase we use the combination of these two algorithms for tuning a strong algorithm.

III. THE PROPOSED ALGORITHM

In this section, a new efficient algorithm called GELSGA (GELS + GA) is developed to solve the CVRP. The main motive for providing this method is to present a new hybrid approach to solve the CVRP problem. Two main goals are followed for combining GA and GELS algorithm. The first goal is to provide a strong solution for CVRP problem. Comparing the results of the proposed method with previous ones based on the standard benchmarks shows how close we are to this end. The second goal is to provide a method as a complement of creative and heuristic functions. For example, this method can inspire clustering based methods for solving the problem. Because of the mutation operator, GA has a high ability for searching in the problem space. That is, by increasing mutation rate, exploration phase can be done properly. On the other hand, by shifting the GA toward the mutation, we may be away from the exploitation phase. For this reason, we have used GELS algorithm which has great power in local optimization, to complete the GA. In fact, the combination of these two algorithms will generate a variety of solutions from all search space and these solutions will be optimized locally by GELS algorithm. The combination of these two algorithms is described below. GELSGA is a hybrid of two algorithms GA and GELS and tries to find the best solution for the CVRP. Figure 1 depicts the general trend of the proposed algorithm. How to hybridize these two algorithms is shown in this figure.

As shown in Figure 1, the process of problem solving begins with GA. In each generation, mutation and crossover operators are first performed and then a number of chromosomes are selected for the next generation. At this moment, the GELS algorithm starts. Selected chromosomes enter into the GELS algorithm one by one as *CU* and local optimization is performed on them. Optimized chromosomes are reinstated into GA and this trend continues. The pseudo-code of the proposed GELSGA is presented in Algorithm 1. The abbreviation *CU* indicates "Current Solution" and *CA* indicates candidate solution. Current Solution is one of the important parameters on GELS.

Algorithm 1 Pseudo-Code of the Proposed Algorithm GELSGA

```
Initialize GA and GELS parameters
Create initial population and calculate fitness
do
    for each chromosome S_i in population do
        crossover(S_i)
        mutation(S_i)
    end
    select new population
    for each chromosome S_i in new population do
        set S_i as CU
        do
            calculate CA by CU
            if fitness (CA) < fitness (CU) then
                CU = CA
                update(velocity matrix)
                update(time matrix)
            end
        while termination criteria GELS
    end
while termination criteria of GA
return best chromosome
```

In the following subsection, the details of the GELSGA algorithm and the precise function of GA and GELS in the proposed algorithm are described.

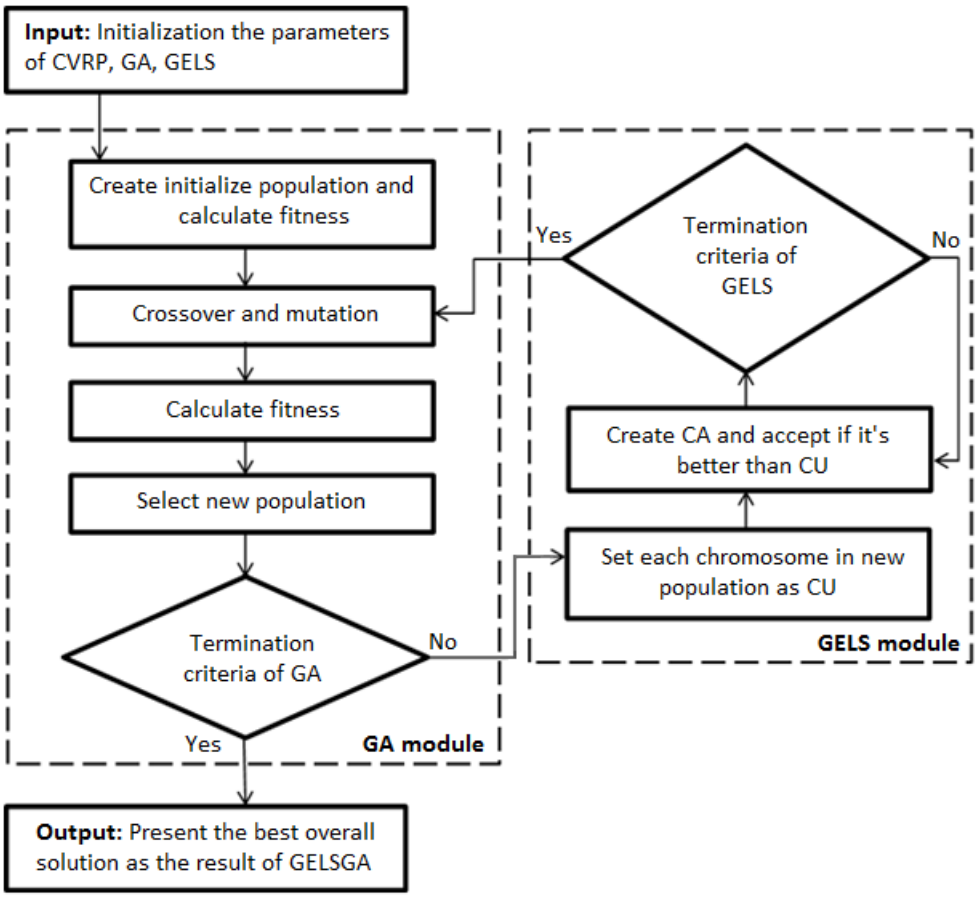

FIGURE 1. Flowchart of the proposed algorithm.

Vehicle	1					2				3		
Customer	3	5	7	1		2	10	6		4	9	8

FIGURE 2. The structure of a sample chromosome in the proposed algorithm.

A. CHROMOSOME REPRESENTATION

In the proposed algorithm, each chromosome represents a solution to the CVRP. A one-dimensional array is applied to represent the chromosomes. If the number of customers is equal to n and if the number of vehicles is equal to k, then $n + k - 1$ is considered to be the length of the array. To generate a solution, first, we put zero in some random cells of the array and then put a permutation of $[1 \ldots n]$ in the remaining cells of the array. Each number indicates one of n customers. Figure 2 shows a chromosome sample in the proposed algorithm.

In the chromosome shown in Figure 2, serving the customers $\{3, 5, 7, 1\}$ is done by vehicle No. 1, serving the customers $\{2, 10, 6\}$ is done by vehicle No. 2 and serving the customers $\{4, 9, 8\}$ is done by vehicle No. 3. In fact, zero numbers in the chromosome separate the constructed routes of the different vehicles.

B. FITNESS FUNCTION

The fitness value of each chromosome is determined by calculating the total traveled distance by vehicles and considering the vehicle's capacity. Depending on the type of chromosome representation, the customers of each vehicle are scrolled from left to right, and sum of their distances are calculated. A penalty function is used for capacity violation of a vehicle. If customers' total demand assigned to a vehicle exceeds this capacity, then a penalty is added to the total cost of that vehicle. Hence, the total cost of all vehicles is considered as the fitness of the chromosome. Equation 1 expresses the fitness function of a chromosome. Index t is used to denote the vehicles. The maximum number of the used vehicles to serve all the customers is equal to V.

$$\begin{aligned} F(x) &= \sum_{t=1}^{V} F(x(t)) \\ F(x(t)) &= \sum_{j=1}^{n} \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2} \\ &\quad + penalty(x(t)) \end{aligned} \tag{1}$$

In Equation 1, the term $\sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2}$ is used to calculate the distance between customers j and $j + 1$ and $penalty(x(t))$ is the penalty function. Customer j is in the location (x_j, y_j). Finally, after calculating the fitness of each customer, the sum of finesses of all customers is considered as the overall fitness of the chromosome. To calculate the value

Chromosome												
Parent 1	3	4	10	0	9	1	5	0	8	6	2	7
Parent 2	9	3	2	8	5	0	10	4	0	6	7	1
Child 1	9	3	2	0	8	5	10	0	4	6	7	1
Child 2	3	4	10	9	1	0	5	8	0	6	2	7

FIGURE 3. Crossover type 1.

Chromosome												
Parent 1	3	7	9	0	1	10	8	6	0	2	4	5
Parent 2	7	10	6	8	0	9	4	5	3	0	1	2
Step 1	3	7	9	0	0	9	4	5	3	0	1	2
Step 2	3	7	9	0	*del*	9	4	5	3	0	1	2
Step 3	3	7	9	0	*10*	*6*	4	5	*8*	0	1	2
Child	3	7	9	0	10	6	4	5	8	0	1	2

FIGURE 4. Crossover type 2.

of $penalty(x(t))$, Equation 2 is used. In this Equation, $devoted_t$ indicates the amount of load allocated to the vehicle t^{th} and is $maxCapacity_t$ is the maximum capacity of vehicle t^{th}. Therefore, $devoted_t - maxCapacity_t$ indicates the overload of vehicle. To calculate the penalty, $devoted_t - maxCapacity_t$ is calculated for all vehicles and then summed up.

$$penalty\,(x\,(t)) = \sum_{t=1}^{V} (devoted_t - maxCapacity_t) \qquad (2)$$

C. CROSSOVER OPERATION

Two types of crossover are designed for the proposed algorithm. In the first one, each child inherits the order of serving to customers from a chromosome and the number of customers for each vehicle from another chromosome. Figure 3 shows an example of this operator. The genes of a chromosome are shown in black and the genes of another chromosome are shown in underlined numbers.

In crossover type 2, each child inherits part of the scheduling from parent one and another part from parent two. In this case, there are two problems. Firstly, the number of zeroes may be less or more than $k-1$ (k: number of vehicles). Then, some numbers appear twice in the child's chromosomes. To fix the first problem, we set the number of zeros by deleting or adding to exactly $k-1$ zero in chromosome. To fix the second problem, we remove the repeated number and replace it with the deleted number. Algorithm 2 shows the pseudo-code of this type of crossover.

Figure 4 shows the crossover type 2. As seen, only one child is generated. The genes of a chromosome are shown in underlined numbers and genes of other chromosome are shown in black. It is assumed that $randposition = 4$. There are three steps to generate a child. In the first step, four genes from the first chromosome and eight genes from the second chromosome are concatenated together. In the second step, one of zeroes shown in italic is removed. In the third step, the deleted numbers of 10, 6, and 8 are returned to the chromosome, and replaced with the numbers 3, 9 and the deleted zero.

Algorithm 2 Crossover Chromosome C_i and C_j

```
randposition = random integer number from 1 to k
    // where k is number of vehicles
for indice = 1 to (n + k − 1) do
    // where n is number of Customers
    if indice < randposition then
        copy C_i (indice) to child (indice) else
            copy C_j (indice) to child (indice)
        end
    end
    adjust number of zeros in child chromosome
    joint = {C_i (1 ... randposition) and
    C_j (1 ... randposition)}
    repeated = C_i (1 ... randposition) − joint
    replace repeated by C_j (1 ... randposition) − joint
    in child (randposition + 1 ... (n + k − 1))
end
return child
```

Chromosome												
Before mutation	3	4	9	0	5	1	6	7	0	8	10	2
After mutation	3	4	9	0	5	1	0	7	6	8	10	2

FIGURE 5. Mutation type 1.

Chromosome												
Before mutation	3	4	9	0	5	1	6	7	0	8	10	2
After mutation	3	4	9	0	*7*	6	*5*	*1*	0	8	10	2

FIGURE 6. Mutation type 2.

D. MUTATION OPERATION

The proposed algorithm uses two types of mutations. The first type of mutation affects the entire solution. In this mutation, two random points are selected from the chromosome and their contents are exchanged. This way, we can exchange the customers of two vehicles and also change the customers of a vehicle. Figure 5 shows an example of this type of mutation. Underlined genes are mutated.

In mutation type 1, only the order of serving of a vehicle is changed and the structure of scheduling is not changed. In this mutation, a non-zero gene is selected as a pivot and all customers of a vehicle are mutated based on this pivot. Figure 6 shows an example of this type of mutation. In this example, the gene contains six as a selected pivot shown in underlined numbers. Mutated genes are shown in italic.

E. MATRICES OF RADIUS, VELOCITY AND MASS IN GELS

The GELS algorithm is implemented using three matrices for *radius*, *velocity* and *mass*. *Radius* matrix is set once, and remains constant until the end of the algorithm. But the *velocity* and *mass* matrices are updated in each iteration of

	Distance (Radius)			Velocity			Mass		
	a	b	c	a	b	c	a	b	c
a	0	10	20	0	100	100	0	6	12
b	10	0	30	100	0	100	6	0	18
c	20	30	0	100	100	0	12	18	0

FIGURE 7. An example of matrices $distance_{(n\times n)}$, $velocity_{(n\times n)}$, and $mass_{(n\times n)}$.

	Distance (Radius)			Velocity			Mass		
	a	b	c	a	b	c	a	b	c
A	0	10	20	0	100	100	0	6	12
B	10	0	30	102.67	0	100	5.84	0	18
C	20	30	0	100	100	0	12	18	0

FIGURE 8. Updating matrices after accepting *CA*.

GELS. This section describes the details of each matrix and how they are updated.

The first matrix in GELS algorithm is the *radius* matrix. In CVRP, the radius is equal to the distance between customers. If n is the number of customers in CVRP, then *radius* matrix is defined as $distance_{(n\times n)}$, in which entry $distance(i, j)$ indicates the n distance between two customers i and j. Obviously, the main diameter of the matrix is zero. *Velocity* matrix indicates the speed of each customer towards another customer. This matrix is defined as $velocity_{(n\times n)}$, in which entry $velocity\,(i, j)$ indicates the current speed of customer i toward customer j. At the beginning, a default value is considered as the velocity. In each iteration of GELS algorithm, the *velocity* matrix is updated. The default velocity of all entries of the matrix is set to 100. *Mass* matrix is defined as $mass_{(n\times n)}$. Using $distance\,(i, j)$ and $velocity\,(i, j)$ we can calculate the $mass\,(i, j)$. In the proposed algorithm, the arrival time from customer i to customer j is considered as $mass\,(i, j)$. Equation 3 is used to calculate the entries of the matrix $mass_{(n\times n)}$.

$$mass(i, j) = \frac{distance\,(i, j)}{velocity\,(i, j)} \cdot 60 \tag{3}$$

Figure 7 shows three matrices $distance_{(n\times n)}$, $velocity_{(n\times n)}$ and $mass_{(n\times n)}$ for $n = 3$ and three customers a, b, c. Entries of distance matrix are filled randomly. Velocity matrix contains the default value 100 for all entries (except the main diameter) and the mass matrix is filled according to Equation 3.

F. PRODUCING CA FROM CU

In the proposed algorithm, each selected chromosome in GA enters into GELS for optimization. Therefore, each chromosome is considered as a *CU* (Current Solution). GELS algorithm creates a number of *CA* (Candidate Solution) to optimize the chromosome. If *CA* is better, it is accepted as *CU* for the next iteration. After accepting *CA*, the velocity matrix is updated using Equations 4 and 5.

$$F = G \cdot \frac{fitness\,(CU) - fitness\,(CA)}{distance\,(i, j)^2} \tag{4}$$

$$velocity_{m+1}\,(i, j) = velocity_m\,(i, j) + F \tag{5}$$

In the Equations 4 and 5, F is the amount of force, G is gravitational constant set to 6.672 [58]. $fitness\,(CU)$ is the fitness of CU, $fitness\,(CA)$ is the fitness of CA and i and j are customers on which gravitational force is applied. $velocity_m\,(i, j)$ is current value of velocity and $velocity_{m+1}\,(i, j)$ is new value of velocity where m is counter of GELS execution. After updating $velocity_{m+1}\,(i, j)$, $mass\,(i, j)$ is updated using Equation 3. For applying gravitational force on points i and j of chromosome CU, we arrange all genes (non-zero ones) after j according to mass and in ascending order. According to Figure 7, assume that $CU = [b, 0, c, a]$ and $(i, j) = (CU(1), CU(3))$. By arranging mass from gene c to the end of the chromosome, we have $CA = [b, 0, c, a]$. Now if assume that $fitness\,(CU) = 100$ and $fitness\,(CA) = 60$, then we must accept CA as CU. Entries $velocity\,(b, a)$ and $mass\,(b, a)$ are updated according to Equations 4, 5 and 3. The states of these three matrices after updating are shown in Figure 8. Updated entries are shown in underlined numbers.

Gravitational force is applied to all non-zero genes of the chromosome. So GELS algorithm is run n times for each chromosome (n is the number of customers). Algorithm 3 shows the pseudo-code of producing *CA* and updating matrices.

Algorithm 3 Pseudo-code of function GELS on chromosome *Ch*

```
1  CU = Ch
2  for each non-zero gene j in CU do
3      CA = sort (Ch (j ... (n + k − 1))) based on
       mass (j, :)
4      if fitness(CU)<fitness(CA) then
5          CU = CA
6          i = first non-zero gene before j
7          F = G · (fitness(CU)−fitness(CA)) / distance(i,j)^2
8          velocity (i, j) = velocity (i, j) + F
9          mass () = (distance (i, j) /velocity (i, j)) · 60
10     end
11     return CU
12 end
```

IV. COMPUTATIONAL RESULTS

The computational results are presented including well-known sets of test data and the algorithmic configurations in this section. Also, the obtained results using proposed GELSGA method are compared with the results of the state-of-the-art solution methods as well as the best known solutions (BKS) existed in the literature. The proposed algorithm is coded using Matlab programming language and is run on Intel(R) Core(TM) i7 CPU 950 @ 3.07GHz (24596 Mflops), and a conversion factor is applied to make the algorithm compatible with other existed algorithms in different result tables of this section according to Dongarra benchmarks [47].

TABLE 1. The obtained results for the benchmarks of Christofides *et al.* [48] – Part I.

Problem	SBA [51]	TS [51]	GA_BB [52]	SS-ACO [53]	PSO [54]	Hyb GENPSO [55]	POHA [56]	CVRP_GELS [57]		GELSGA		BKS
	Best val.	Best val.	Best val.	Best val.	Best val.	Best val.	Best val.	Avg. of 10	Best of 10	Avg. of 10	Best of 10	
Vrpnc1	528	**524**	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	528	**524.61**	527	**524.61**	524.61
Vrpnc2	838	844	**835.26**	**835.26**	844.42	**835.26**	**835.26**	838	**835.26**	837	**835.26**	835.26
Vrpnc3	829	835	827.39	830.14	829.40	**826.14**	**826.14**	830	**826.14**	829	**826.14**	826.14
Vrpnc4	1058	1052	1036.16	1038.20	1048.89	**1028.42**	**1028.42**	1035	**1028.42**	1034	**1028.42**	1028.42
Vrpnc5	1376	1354	1324.06	1307.18	1323.89	1294.21	1291.83	1302	1294.21	1299	**1291.29**	1291.29
Vrpnc6	**555**	**555**	**555.43**	559.12	**555.43**	**555.43**	**555.43**	559	**555.43**	558	**555.43**	555.43
Vrpnc7	**909**	913	**909.68**	912.68	917.68	**909.68**	**909.68**	920	914.13	917	**909.68**	909.68
Vrpnc8	866	866	868.32	869.34	867.01	**865.94**	**865.94**	876	869.34	872	**865.94**	865.94
Vrpnc9	1164	1188	1169.15	1179.4	1181.14	1163.41	1170.25	1174	**1162.55**	1169	**1162.55**	1162.55
Vrpnc10	1418	1422	1418.79	1410.26	1428.46	1397.51	1400.13	1405	**1395.85**	1403	**1395.85**	1395.85
Vrpnc11	1176	**1042**	1043.11	1044.12	1051.87	**1042.11**	**1042.11**	1058	**1042.11**	1051	**1042.11**	1042.11
Vrpnc12	826	**819**	**819.56**	824.31	**819.56**	**819.56**	**819.56**	829	**819.56**	822	**819.56**	819.56
Vrpnc13	1547	1547	1553.12	1556.52	1546.20	1544.57	**1383.98**	1564	1541.14	1549	1541.14	1541.14
Vrpnc14	890	**866**	**866.37**	870.26	**866.37**	**866.37**	**866.37**	873	**866.37**	870	**866.37**	866.37

TABLE 2. The obtained results for the benchmarks of Christofides *et al.* [48] – Part II.

Problem	OCGA [59]	AGES [36]	VNSA [60]	HGSADC [61]	HGSADC+ [46]	SEPAS [62]	GA_P [63]	GELSGA		BKS
	Best val.	Best val.	Best val.	Best val.	Best val.	Best val.	Best val.	Avg. of 10	Best of 10	
Vrpnc1	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	**524.61**	527	**524.61**	524.61
Vrpnc2	**835.26**	**835.26**	**835.26**	**835.26**	**835.26**	**835.26**	**835.26**	837	**835.26**	835.26
Vrpnc3	**826.14**	**826.14**	**826.14**	**826.14**	**826.14**	**826.14**	**826.14**	829	**826.14**	826.14
Vrpnc4	**1028.42**	**1028.42**	1031.44	**1028.42**	**1028.42**	**1028.42**	1030.46	1034	**1028.42**	1028.42
Vrpnc5	1299.64	**1291.29**	-	1294.06	1291.45	1311.48	1296.39	1299	**1291.29**	1291.29
Vrpnc6	**555.43**	**555.43**	-	**555.43**	**555.43**	**555.43**	**555.43**	558	**555.43**	555.43
Vrpnc7	**909.68**	**909.68**	-	**909.68**	**909.68**	**909.68**	**909.68**	917	**909.68**	909.68
Vrpnc8	**865.94**	**865.94**	-	**865.94**	**865.94**	**865.94**	**865.94**	872	**865.94**	865.94
Vrpnc9	1163.38	**1162.55**	-	**1162.55**	**1162.55**	**1162.55**	**1162.55**	1169	**1162.55**	1162.55
Vrpnc10	1406.23	1401.12	-	1400.23	**1395.85**	1407.21	1402.75	1403	**1395.85**	1395.85
Vrpnc11	**1042.11**	**1042.11**	**1042.11**	**1042.11**	**1042.11**	**1042.11**	**1042.11**	1051	**1042.11**	1042.11
Vrpnc12	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	**819.56**	822	**819.56**	819.56
Vrpnc13	1542.25	**1541.14**	-	1543.07	**1541.14**	1544.01	1542.86	1549	**1541.14**	1541.14
Vrpnc14	**866.37**	**866.37**	-	**866.37**	**866.37**	**866.37**	**866.37**	870	**866.37**	866.37

TABLE 3. CPU times of the algorithms for solving Christofides *et al.* [48] benchmarks.

Problem	SBA [51]	TS [51]	GA_BB [52]	SS-ACO [53]	Hyb GENPSO [55]	VNSA [60]	HGSADC+ [46]	POHA [56]	CVRP_GELS [57]	GELSGA
Vrpnc1	0.33	0.22	0.48	32.39	0.70	3.65	7.34	11.93	51	75
Vrpnc2	12.56	0.35	3.48	41.23	15.33	7.1	10.46	29.83	68	89
Vrpnc3	18.22	3.01	6.67	70.67	17.42	8.51	22.11	39.38	136	157
Vrpnc4	9.78	6.95	11.87	147.83	43.21	17.53	39.05	52.51	173	194
Vrpnc5	4.52	6.33	13.31	416.98	126.14	23.66	69.63	180.19	541	611
Vrpnc6	6.65	0.34	0.61	38.28	0.70	-	7.12	22.67	58	62
Vrpnc7	1.22	2.06	2.41	53.01	13.94	-	10.74	33.41	89	97
Vrpnc8	1.87	5.85	1.2	123.68	43.90	-	21.01	40.57	192	218
Vrpnc9	164.52	9.28	4.41	306.85	75.96	-	36.09	157.52	437	471
Vrpnc10	11.14	8.90	10.62	596.03	152.62	-	89.22	225.54	997	1046
Vrpnc11	0.62	2.82	5.43	136.64	10.45	11.05	32.56	83.53	225	278
Vrpnc12	1.23	1.74	0.25	91.88	18.12	7.1	16.38	36.99	149	183
Vrpnc13	14.88	5.53	8.2	275.04	21.60	-	32.03	77.57	451	495
Vrpnc14	0.60	2.30	0.39	217.33	16.73	-	18.36	35.8	357	384
Mean	**17.72**	**3.98**	**4.95**	**181.99**	**39.77**	**11.23**	**29.44**	**73.39**	**280.29**	**311.43**

Note. Computational times are presented in seconds using a machine with the function of 24596 Mflops.

A. THE TEST DATA SETS

Three sets of standard benchmarks including 14 test problems from Christofides *et al.* [48] and 12 test problems from Taillard [49] and 20 test problem from Golden *et al.* [50] have been considered in order to verify the optimality of the proposed algorithm. In Christofides *et al.* [48], instances 1, 2, 3, 4, 5, 11, and 12 have only capacity constraints. Instances 6, 7, 8, 9, 10, 13, and 14 have, besides the capacity constraint, maximum route length limits and non-zero service times (service time is added to their limitation when they serve the customer). In instances 1 to 10, customers have been distributed randomly, but they have been clustered in instances 11 to 14. Twenty instance problems of Golden *et al.* [50] have been defined with 200 to 483 customers. Route length limits are defined for the first 8 problems. Each problem has a geometrical structure, such as concentric circles including 8 instances, squares including 4 instances, rhombuses including 4 instances, and six-point stars including 4 instances. For each benchmark, two tables are provided. One paper shows the average and best solutions of different problems and the other one shows the processing time needed to generate these solutions. Most methods chosen for comparison with the proposed method have used evolutionary algorithms to solve this problem. One of the features of evolutionary algorithms is that it can produce better solutions by running longer. Therefore, the comparison of the two evolutionary based methods in terms on processing time is not reasonable. In order to have a correct comparison, we should investigate processing time table according to the solution table. For example, if two different methods have produced the same solution based on their own parameters, then the method with less time to produce this solution is best one. In the Tables 1, 2, 4, and 6, the bold cells represent the best results obtained using different methods. If the best results

TABLE 4. The obtained results for the benchmarks of Taillard [49].

Problem	OCGA [59]	AGNES [36]	JCell2oli [64]	CVRP_GELS [57]		GELSGA		BKS
	Best val.	Best val.	Best val.	Avg. of 10	Best of 10	Avg. of 10	Best of 10	
Tai75a	**1618.36**	**1618.36**	**1618.36**	1625	**1618.36**	1621	**1618.36**	1618.36
Tai75b	1344.63	1344.64	**1344.62**	1348	**1344.62**	1346	**1344.62**	1344.62
Tai75c	**1291.01**	**1291.01**	**1291.01**	1298	**1291.01**	1293	**1291.01**	1291.01
Tai75d	**1365.42**	**1365.42**	**1365.42**	1369	**1365.42**	1366	**1365.42**	1365.42
Tai100a	2050.64	**2041.34**	2047.90	2056	**2041.34**	2053	**2041.34**	2041.34
Tai100b	**1939.90**	**1939.90**	1940.36	1967	1947.07	1955	1940.61	1940.61
Tai100c	1408.40	**1406.20**	1411.66	1429	**1406.20**	1419	**1406.20**	1406.20
Tai100d	1581.22	1581.25	1584.20	1609	1581.25	1602	**1580.19**	1581.25
Tai150a	**3055.23**	**3055.23**	3056.41	3099	3069.14	3064	**3055.23**	3055.23
Tai150b	2755.09	2727.67	2732.75	2769	**2656.47**	2741	**2656.47**	2656.47
Tai150c	2352.86	2343.11	2364.08	2394	**2341.84**	2372	**2341.84**	2341.84
Tai150d	2660.33	**2645.40**	2654.69	2702	2659.02	2685	2646.19	2645.39

TABLE 5. CPU times of the algorithms for solving Taillard [49] benchmarks.

Problem	AGNES [36]	CVRP_GELS [57]	GELSGA
Tai75a	0.49	39	62
Tai75b	0.24	48	87
Tai75c	1.22	41	71
Tai75d	0.73	63	96
Tai100a	31.71	75	103
Tai100b	3.9	89	118
Tai100c	8.05	84	137
Tai100d	19.76	92	162
Tai150a	14.64	117	179
Tai150b	102.46	114	175
Tai150c	0.49	138	194
Tai150d	219.55	129	189
Mean	**33.6**	**85.75**	**131.08**

were obtained using two or more algorithms then as a ''best one algorithm'' we can select the algorithm with the lowest computational time.

B. RESULTS OF CHRISTOFIDES et al. [48] BENCHMARKS

Tables 1 and 2 show the computational results. Compared to other algorithms, the proposed algorithm could find 13 better solutions as illustrated in Table 1 and 14 better solutions as illustrated in Table 2. In the illustrative tables, two columns are presented for the proposed algorithm. The first column indicates the average quality for the solution (averaged over 10 experiments), and the second column indicates the best result obtained from the 10 experiments. CPU time of the proposed algorithm and the other compared algorithms is shown in Table 3 for solving Christofides benchmarks. All the CPU times are represented in seconds.

C. RESULTS OF TAILLARD [49] BENCHMARKS

The obtained results for 12 instances of Taillard [49] are shown in Table 4. Each test problem is solving 10-fold using given algorithm. The average values and the best results are computed. The ''BKS'' column in Table 4 present a current best solution known in literature, but of course we are not sure whether this solution is optimal one. Therefore the better solution than ''BKS'' can be found. The computational times of Taillard benchmark instances are also presented in Table 5 in seconds.

D. RESULTS OF GOLDEN et al. [50] BENCHMARKS

The obtained results for 20 instances of Golden *et al.* [50] are shown in Table 6. The first two columns denote the number

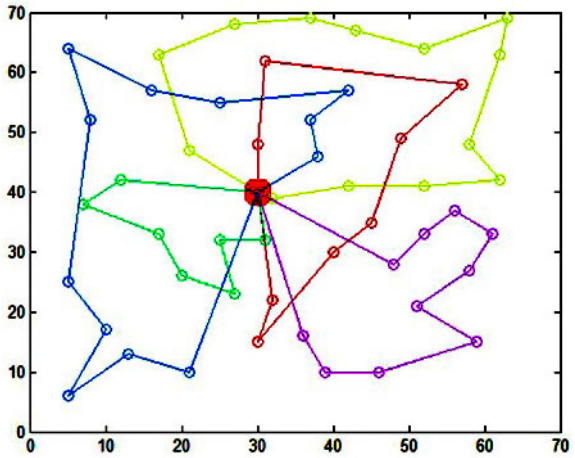

FIGURE 9. The obtained results of GELSGA for 50 customers and the capacity of 160 (Vrpnc1).

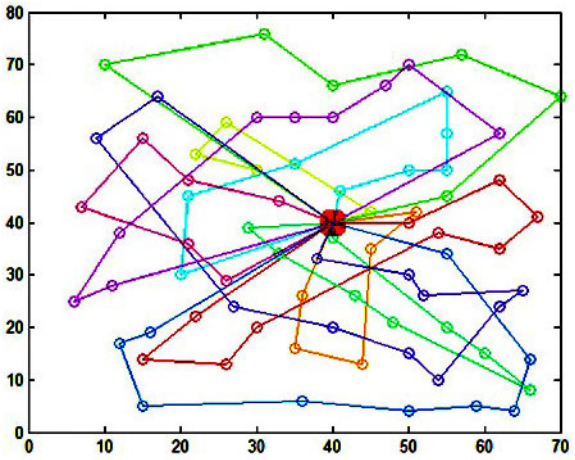

FIGURE 10. The obtained results of GELSGA for 75 customers and the capacity of 140 (Vrpnc2).

of customers n and the number of vehicles k, respectively. Q and L denote capacity of vehicles and maximum tour length, respectively. Each instance is run 10 times under the experimental scenario. Average values and best results with standard deviation are computed. As it was mentioned in previous sections, the stopping criterion has been determined based on different methods. The ''BKS'' column in Table 6 present a current best solution known in literature, but of

TABLE 6. The obtained results for the benchmarks of Golden *et al.* [50].

Problem	n	k	Q	L	CPM standard [45] Best val.	HGSADC+ [46] Best val.	GTS [65] Best val.	Hybrid GA [63] Best val.	PA [37] Best val.	POHA [56] Best val.	GELSGA Best val.	BKS
Kelly01	240	10	550	650	5623.65	5623.47	5736.15	5646.63	5795.61	5623.47	**5578.30**	5646.46
Kelly02	320	10	700	900	8434.78	8413.82	8553.03	8447.92	8501.67	8405.45	**8392.21**	8566.04
Kelly03	400	10	900	1200	11036.22	11036.22	11402.75	11036.22	11364.69	11036.22	**10997.42**	11649.06
Kelly04	480	12	1000	1600	**13620.30**	13624.53	14910.62	13624.52	14136.32	13624.53	13753.16	14639.32
Kelly05	200	5	900	1800	6460.98	6460.98	6697.53	6460.98	6512.27	6460.98	**6385.94**	6702.73
Kelly06	280	8	900	1500	8404.06	8412.90	8963.32	8412.80	8553.19	8412.90	**8403.68**	9016.93
Kelly07	360	9	900	1300	10134.93	10115.58	10547.44	10195.59	10422.65	10107.75	**10103.59**	11047.69
Kelly08	440	11	900	1200	11635.34	11635.34	12036.24	11828.78	11986.73	11635.34	**11632.14**	12250.06
Kelly09	255	14	1000	-	580.04	579.71	593.35	591.54	586.68	579.71	**577.10**	587.09
Kelly10	323	16	1000	-	**737.16**	737.43	751.66	751.41	748.89	737.43	737.31	746.56
Kelly11	399	18	1000	-	912.72	913.15	936.04	933.04	924.70	913.12	**912.48**	932.68
Kelly12	483	19	1000	-	**1103.20**	1104.96	1147.14	1133.79	1125.71	1109.09	1110.82	1137.18
Kelly13	252	27	1000	-	858.57	857.19	868.80	875.16	867.29	857.19	**853.65**	881.04
Kelly14	320	30	1000	-	**1080.55**	**1080.55**	1096.18	1086.24	1098.86	**1080.55**	**1080.55**	1103.69
Kelly15	396	34	1000	-	1340.13	1339.75	1369.44	1367.37	1356.65	1338.00	**1337.79**	1364.23
Kelly16	480	38	1000	-	1614.73	1616.09	1652.32	1650.94	1642.90	1612.53	**1611.99**	1650.94
Kelly17	240	22	200	-	707.80	707.79	711.07	710.42	712.26	707.76	**707.72**	666.84
Kelly18	300	28	200	-	998.90	**995.13**	1016.83	1014.8	1017.91	**995.13**	997.26	973.60
Kelly19	360	33	200	-	1366.12	1366.40	1400.96	1376.49	1384.93	1365.60	**1363.16**	1338.74
Kelly20	420	41	200	-	1819.76	1819.59	1915.83	1846.55	1855.91	1818.25	**1818.21**	1831.62

TABLE 7. CPU times of the algorithms for solving Golden *et al.* [50] benchmark instances.

Problem	CMP standard [45]	HGSADC+ [46]	GTS [65]	Hybrid GA [63]	PA [37]	POHA [56]	GELSGA
Kelly01	578.34	195.26	0.003	0.67	1.35	101.43	101
Kelly02	897.54	324.6	0.005	1.62	3.24	121.72	112
Kelly03	1170.85	475.61	0.008	2.52	5.03	266.11	156
Kelly04	1596.54	548.58	0.009	3.91	7.81	330.55	318
Kelly05	415.2	107.08	0.001	0.02	0.04	101.43	103
Kelly06	701.88	197.43	0.003	0.21	0.42	130.07	123
Kelly07	1023.18	411.83	0.007	0.81	1.63	156.33	147
Kelly08	1432.35	507.35	0.007	1.84	3.68	258.95	239
Kelly09	509.62	288.92	0.007	0.3	0.6	99.05	98
Kelly10	755.91	478.57	0.01	0.76	1.52	142.01	134
Kelly11	1084.03	726.51	0.02	1.63	3.27	261.34	239
Kelly12	1528.87	841.9	0.026	0.64	1.29	431.98	385
Kelly13	473.69	166.31	0.007	0.32	0.64	103.82	96
Kelly14	657.81	173.71	0.009	0.71	1.42	112.17	117
Kelly15	952.89	502.78	0.011	0.71	1.42	346.06	327
Kelly16	1262.65	713.28	0.014	2.73	5.45	461.82	402
Kelly17	429.89	110.35	0.009	0.12	0.24	114.56	122
Kelly18	655.98	209.3	0.013	0.82	1.64	126.49	119
Kelly19	855.05	316.76	0.018	1.55	3.09	136.04	138
Kelly20	1099.77	546.28	0.026	4.38	8.76	250.6	237
Mean	**904.1**	**392.12**	**0.01**	**1.31**	**2.63**	**202.63**	**185.65**

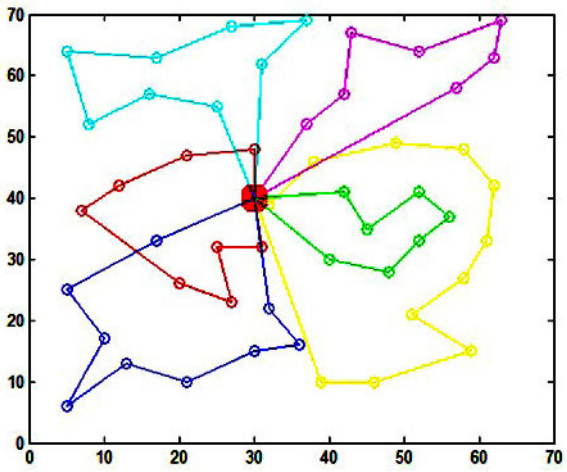

FIGURE 11. The obtained results of GELSGA for 50 customers and the capacity of 160 (Vrpnc6).

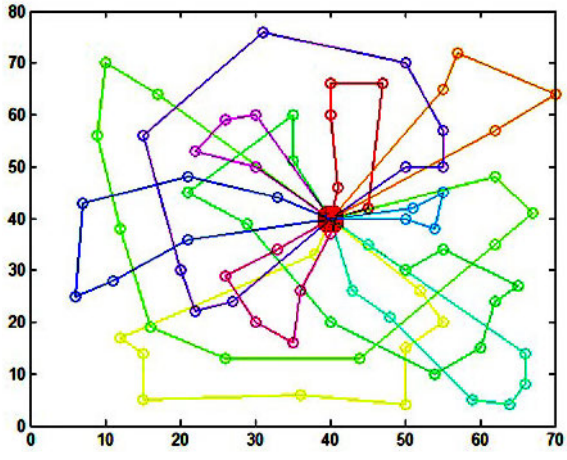

FIGURE 12. The obtained results of GELSGA for 75 customers and the capacity of 140 (Vrpnc7).

course we are not sure whether this solution is optimal one. Therefore the better solution than "BKS" can be found. According to the obtained results, the proposed algorithm is capable of finding remarkable solutions in comparison with other tested algorithms. The computational times of Golden et al. benchmarks are also shown in Table 7.

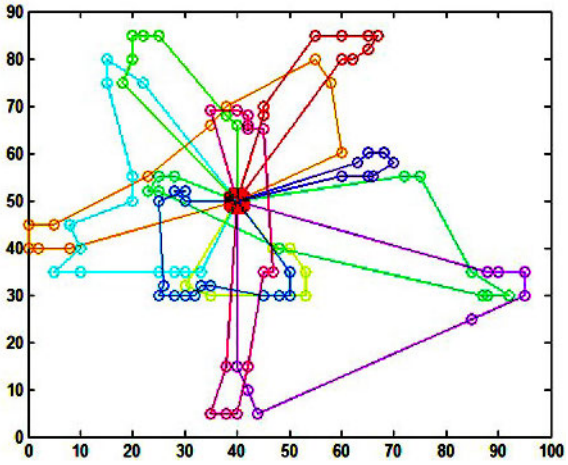

FIGURE 13. The obtained results of GELSGA for 100 customers and the capacity of 200 (Vrpnc12).

Figures 9- 13 illustrate the solutions obtained for some instances of CVRP. As it can be seen, the proposed algorithm (GELSGA) could solve the problems well and compete with best-known algorithms.

V. CONCLUSION

In this paper, a new hybrid algorithm named GELSGA is proposed to solve the CVRP. We modified some parameters of the basic GELS algorithm to improve the performance of the proposed algorithm. The modifications included building neighborhoods, various mutation operators, and population optimization on the best and worst solutions in each iteration. These modifications made the algorithm more efficient in escaping local optimum points and obtaining good solutions. Experimental results show that the proposed algorithm is very efficient for solving the CVRP and could achieve better solutions, however, it requires more running time because of its combination with LSA. It seems that that these combinations with other meta-heuristic methods such as TS or PSO and employing strong LSA such as the dual improving search algorithm may yield better solutions. Moreover, the proposed algorithm can be applied to other combinatorial optimization problems such as Vehicle Routing Problem with Time Windows (VRPTW) and Capacitated Clustering Problem. Work on these ideas and their applications will be dealt in future research.

REFERENCES

[1] S. Shamshirband, M. Shojafar, A. R. Hosseinabadi, and A. Abraham, "OVRP_ICA: An imperialist-based optimization algorithm for the open vehicle routing problem," in *Proc. Int. Conf. Hybrid Artif. Intell. Syst. (HAIS)*, in Lecture Notes in Computer Science, vol. 9121. Cham, Switzerland: Springer, 2015, pp. 221–233.

[2] S. Shamshirband, M. Shojafar, A. R. Hosseinabadi, and A. Abraham, "A solution for multi-objective commodity vehicle routing problem by NSGA-II," in *Proc. Int. Conf. Hybrid Intell. Syst. (HIS)*, 2014, pp. 12–17.

[3] G. Laporte, "Fifty years of vehicle routing," *Transp. Sci.*, vol. 43, no. 4, pp. 408–416, Nov. 2009.

[4] G. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Oper. Res.*, vol. 12, no. 4, pp. 568–581, 1964.

[5] M. Flood, "The traveling-salesman problem," *Oper. Res.*, vol. 4, no. 1, pp. 61–75, Feb. 1956.

[6] J. Clarke, V. Gascon, and J. A. Ferland, "A capacitated vehicle routing problem with synchronized pick-ups and drop-offs: The case of medication delivery and supervision in the DR Congo," *IEEE Trans. Eng. Manag.*, vol. 64, no. 3, pp. 327–336, Aug. 2017.

[7] M. Kargari and M. M. Sepehri, "Stores clustering using a data mining approach for distributing automotive spare-parts to reduce transportation costs," *Expert Syst. Appl.*, vol. 39, no. 5, pp. 4740–4748, 2012.

[8] H. M. Moftah, A. T. Azar, E. T. Al-Shammari, N. I. Ghali, A. E. Hassanien, and M. Shoman, "Adaptive k-means clustering algorithm for MR breast image segmentation," *Neural Comput. Appl.*, vol. 24, nos. 7–8, pp. 1917–1928, 2014.

[9] W. Zhang, H. He, and B. Cao, "Identifying and evaluating the Internet opinion leader community based on *k*-clique clustering," *Neural Comput. Appl.*, vol. 25, pp. 595–602, Sep. 2014.

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

[11] A. N. Letchford and J.-J. Salazar-González, "The capacitated vehicle routing problem: Stronger bounds in pseudo-polynomial time," *Eur. J. Oper. Res.*, vol. 272, no. 1, pp. 24–31, 2019.

[12] V. Leggieri and M. Haouari, "A matheuristic for the asymmetric capacitated vehicle routing problem," *Discrete Appl. Math.*, vol. 234, pp. 139–150, Jan. 2018.

[13] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Hoboken, NJ, USA: Wiley, 2009.

[14] M. Melanie, *An Introduction to Genetic Algorithms*, 5th ed. Cambridge, MA, USA: MIT Press, 1999.

[15] M. Dorigo and T. Stutzle, *Ant Colony Optimization*. Cambridge, MA, USA: MIT Press, 2004.

[16] Y. Yonezawa and T. Kikuchi, "Ecological algorithm for optimal ordering used by collective honey bee behavior," in *Proc. 7th Int. Symp. Micro Mach. Hum. Sci.*, 1996, pp. 249–256.

[17] Q. Bai, "Analysis of particle swarm optimization algorithm," *Comput. Inf. Sci.*, vol. 3, no. 1, pp. 180–184, 2010.

[18] F. Glover and M. Laguna, "Tabu search," in *Handbook of Combinatorial Optimization*, vols. 1–3. Norwell, MA, USA: Kluwer, 1997.

[19] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[20] E. B. Tirkolaee, M. Alinaghian, A. R. Hosseinabadi, M. B. Sasi, and A. K. Sangaiah, "An improved ant colony optimization for the multi-trip capacitated arc routing problem," *Comput. Elect. Eng.*, vol. 77, pp. 457–470, Jul. 2019.

[21] B. L. Webster, "Solving combinatorial optimization problems using a new algorithm based on gravitational attraction," Ph.D. dissertation, Florida Inst. Technol., Melbourne, FL, USA, 2004.

[22] Z. Pooranian, M. Shojafar, R. Tavoli, M. Singhal, and A. Abraham, "A hybrid metaheuristic algorithm for job scheduling on computational grids," *Informatica*, vol. 37, no. 2, pp. 157–164, 2013.

[23] Z. Pooranian, A. Harounabadi, M. Shojafar, and J. Mirabedini, "Hybrid PSO for independent task scheduling in grid computing to decrease makespan," in *Proc. Int. Conf. Future Inf. Technol.*, vol. 13, 2011, pp. 327–331.

[24] B. Barzegar and H. Shirgahi, "Advanced reservation and scheduling in grid computing systems by gravitational emulation local search algorithm," *Amer. J. Sci. Res.*, vol. 18, no. 18, pp. 62–70, May 2011.

[25] M. A. Khandke, R. B. Pawar, and A. R. Chinchawade, "A comparative study of advanced reservation algorithms in optical grid," *Int. J. Eng. Res. Technol.*, vol. 2, pp. 1–11, Mar. 2013.

[26] V. G. Rahmati, S. E. Alavi, and I. Attarzadeh, "A reliable and hybrid scheduling algorithm based on cost and time balancing for computational grid," *Adv. Comput. Sci., Int. J.*, vol. 3, no. 3, pp. 22–31, 2014.

[27] A. R. Hosseinabadi, H. Siar, S. Shamshirband, M. Shojafar, and M. M. Nasir, "Using the gravitational emulation local search algorithm to solve the multi-objective flexible dynamic job shop scheduling problem in small and medium enterprises," *Ann. Oper. Res.*, vol. 229, pp. 451–474, 2015.

[28] S. Shamshirband, M. Shojafar, A. R. Hosseinabadi, M. Kardgar, M. H. Nizam, M. Nasir, and R. Ahmad, "OSGA: Genetic-based open-shop scheduling with consideration of machine maintenance in small and medium enterprises," *Ann. Oper. Res.*, vol. 229, no. 1, pp. 743–758, 2015.

[29] A. R. Hosseinabadi, A. B. Farahabadi, M. S. Rostami, and A. F. Lateran, "Presentation of a new and beneficial method through problem solving timing of open shop by random algorithm gravitational emulation local search," *Int. J. Comput. Sci. Issues*, vol. 10, no. 1, pp. 745–752, 2013.

[30] A. B. Farahabadi and A. R. Hosseinabadi, "Present a new hybrid algorithm scheduling flexible manufacturing system consideration cost maintenance," *Int. J. Sci. Eng. Res.*, vol. 4, no. 9, pp. 1870–1875, 2013.

[31] S. Mirzayi and V. Rafe, "A hybrid heuristic workflow scheduling algorithm for cloud computing environments," *J. Exp. Theor. Artif. Intell.*, vol. 27, no. 6, pp. 721–735, 2015.
[32] A. R. Hosseinabadi, J. Vahidi, V. E. Balas, and S. S. Mirkamali, "OVRP_GELS: Solving open vehicle routing problem using the gravitational emulation local search algorithm," *Neural Comput. Appl.*, vol. 29, no. 10, pp. 955–968, 2018.
[33] A. S. Rostami, F. Mohanna, H. Keshavarz, and A. R. Hosseinabadi, "Solving multiple traveling salesman problem using the gravitational emulation local search algorithm," *Appl. Math. Inf. Sci.*, vol. 9, no. 2, pp. 699–709, 2015.
[34] A. R. Hosseinabadi, M. Kardgar, M. Shojafar, S. Shamshirband, and A. Abraham, "GELSGA: Hybrid metaheuristic algorithm for solving multiple travelling salesman problem," in *Proc. Int. Conf. Intell. Syst. Design Appl. (ISDA)*, 2014, pp. 76–81.
[35] S. Mazzeo and I. Loiseau, "An ant colony algorithm for the capacitated vehicle routing," *Electron. Notes Discrete Math.*, vol. 18, pp. 181–186, Dec. 2004.
[36] D. Mester and O. Bräysy, "Active guided evolution strategies for large-scale vehicle routing problems with time windows," *Comput. Oper. Res.*, vol. 32, no. 6, pp. 1593–1614, 2005.
[37] S. Lin, Z. Lee, K. Ying, and C. Lee, "Applying hybrid meta-heuristics for capacitated vehicle routing problem," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 1505–1512, 2009.
[38] P. Chen, H. Huang, and X. Dong, "Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem," *Expert Syst. Appl.*, vol. 37, no. 2, pp. 1620–1627, 2010.
[39] T. Zhen, Y. Zhu, and Q. Zhang, "A hybrid ant colony algorithm for the capacitated vehicle routing problem," in *Proc. IEEE (ITME)*, Dec. 2008, pp. 935–939.
[40] M. A. Mohammed, M. S. Ahmad, and A. Mostafa, "Using genetic algorithm in implementing capacitated vehicle routing problem," in *Proc. IEEE (ICCIS)*, Jun. 2012, pp. 257–262.
[41] B. Stanojević, M. Vujošević, and M. Stanojević, "Enhanced savings calculation and its applications for solving capacitated vehicle routing problem," *Appl. Math. Comput.*, vol. 219, pp. 10302–10312, Jun. 2013.
[42] C. Duhamel, P. Lacomme, A. Quilliot, and H. Toussaint, "A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle routing problem," *Comput. Oper. Res.*, vol. 38, no. 3, pp. 617–640, 2011.
[43] T. Tlili, S. Faiz, and S. Krichen, "A hybrid metaheuristic for the distance-constrained capacitated vehicle routing problem," *Procedia-Social Behav. Sci.*, vol. 109, pp. 779–783, 2014.
[44] P. J. S. Cardoso, G. Schütz, A. Mazayev, E. Ey, and T. Corrêa, "A solution for a real-time stochastic capacitated vehicle routing problem with time windows," *Procedia Comput. Sci.*, vol. 51, no. 1, pp. 2227–2236, 2015.
[45] J. Jin, T. G. Crainic, and A. Løkketangen, "A cooperative parallel metaheuristic for the capacitated vehicle routing problem," *Comput. Oper. Res.*, vol. 44, pp. 33–41, Apr. 2014.
[46] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "A unified solution framework for multi-attribute vehicle routing problems," *Eur. J. Oper. Res.*, vol. 234, no. 3, pp. 658–673, 2014.
[47] J. J. Dongarra, "Performance of various computers using standard linear equations software," Dept. Comput. Sci., Univ. Manchester, Knoxville, TN, USA, Tech. Rep. CS-89-85, 2014.
[48] N. Christofides, A. Mingozzi, and P. Toth, "Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations," *Math. Program.*, vol. 20, pp. 255–282, Dec. 1981.
[49] É. Taillard, "Parallel iterative search methods for vehicle routing problems," *Networks*, vol. 23, no. 8, pp. 661–673, 1993.
[50] B. L. Golden, E. A. Wasil, J. P. Kelly, and I.-M. Chao, "Metaheuristics in vehicle routing," in *Fleet Management and Logistics*, T. G. Crainic and G. Laporte, Eds. Boston, MA, USA: Kluwer, 1998, pp. 33–56.
[51] I. H. Osman, "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Ann. Oper. Res.*, vol. 41, no. 4, pp. 421–451, 1993.
[52] J. Berger and M. Barkaoui, "A hybrid genetic algorithm for the capacitated vehicle routing problem," in *Proc. Int. Genetic Evol. Comput. Conf. (GECCO)*, vol. 2723, 2003, pp. 646–656.
[53] X. Zhang and L. Tang, "A new hybrid ant colony optimization algorithm for the vehicle routing problem," *Pattern Recognit. Lett.*, vol. 30, no. 9, pp. 848–855, 2009.
[54] T. J. Ai and V. Kachitvichyanukul, "A particle swarm optimization for the heterogeneous fleet vehicle routing problem," *Int. J. Logistics SCM Syst.*, vol. 3, no. 1, pp. 32–39, 2009.
[55] Y. Marinakis and M. Marinaki, "A hybrid genetic—Particle swarm optimization algorithm for the vehicle routing problem," *Expert Syst. Appl.*, vol. 37, pp. 1446–1455, Mar. 2010.
[56] 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," *Inf. Sci.*, vols. 334–335, pp. 354–378, Mar. 2016.
[57] A. A. R. Hosseinabadi, N. S. H. Rostami, M. Kardgar, S. Mirkamali, and A. Abraham, "A new efficient approach for solving the capacitated vehicle routing problem using the gravitational emulation local search algorithm," *Appl. Math. Model.*, vol. 49, pp. 663–679, Sep. 2017.
[58] S. R. Balachandar and K. Kannan, "Randomized gravitational emulation search algorithm for symmetric traveling salesman problem," *Appl. Math. Comput.*, vol. 192, pp. 413–421, Sep. 2007.
[59] H. Nazif and L. S. Lee, "Optimised crossover genetic algorithm for capacitated vehicle routing problem," *Appl. Math. Model.*, vol. 36, no. 5, pp. 2110–2117, Sep. 2012.
[60] Y. Xiao, Q. Zhao, I. Kaku, and N. Mladenovic, "Variable neighbourhood simulated annealing algorithm for capacitated vehicle routing problems," *Eng. Optim.*, vol. 46, pp. 562–579, Apr. 2014.
[61] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems," *Oper. Res.*, vol. 60, no. 3, pp. 611–624, 2012.
[62] C. D. Tarantilis, "Solving the vehicle routing problem with adaptive memory programming methodology," *Comput. Oper. Res.*, vol. 32, pp. 2309–2327, Sep. 2005.
[63] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Comput. Oper. Res.*, vol. 31, no. 12, pp. 1985–2002, 2004.
[64] E. Alba and B. Dorronsoro, "Computing nine new best-so-far solutions for capacitated VRP with a cellular genetic algorithm," *Inf. Process. Lett.*, vol. 98, no. 6, pp. 225–230, 2006.
[65] P. Toth and D. Vigo, "The granular tabu search and its application to the vehicle-routing problem," *Inf. J. Comput.*, vol. 15, no. 4, pp. 333–346, Dec. 2003.

ALI ASGHAR RAHMANI HOSSEINABADI received the M.Sc. degree in software engineering from Islamic Azad University Ayatollah Amoli, Iran, in 2016. He has been a Research Assistant with the Engineering Faculty, Tamishan University, since 2013, and faculty member, since 2018. He is the author of more than 100 research articles in refereed journals and international conferences. His research interests include VRP, TSP, WSN, the IoT, and Scheduling. He is a member of several editorial boards and a member of several national and international journals and conferences.

ADAM SLOWIK (M'07–SM'12) was born in Warsaw, Poland, in 1977. He received the B.Sc. and M.Sc. degrees in computer engineering from the Department of Electronics and Computer Science, Koszalin University of Technology, Poland, in August 2001, the Ph.D. degree in electronics from the Department of Electronics and Computer Science, in March 2007, and the Dr. Habilitation degree (D.Sc.) in computer science from the Department of Mechanical Engineering and Computer Science, Czestochowa University of Technology, Poland, in June 2013. Since October 2013, he has been an Associate Professor with the Department of Electronics and Computer Science, Koszalin University of Technology. He is the author or coauthor of over 70 articles and two books (in Polish). His research interests include soft computing, computational intelligence, machine learning, and bio-inspired global optimization algorithms and their engineering applications. He is also a member of the Program Committee of several International conferences in the area of artificial intelligence and evolutionary computation. He is an Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS and a reviewer for many international scientific journals.

MEHDI SADEGHILALIMI is currently a faculty member of the Industrial Engineering Department, Islamic Azad University. His research interests include supply chain management, multiobjective optimization, uncertain optimization, and meta-heuristic algorithms.

MOHAMMAD FAROKHZAD received the M.Sc. degree in software engineering from the University of Kurdistan, Sanandaj, Iran, in 2017. His research interests include VRP, WSN, optimization, game theory, mechanism design, and data Science.

MORTEZA BABAZADEH SHAREH received the bachelor's and master's degrees from Islamic Azad University, Iran, in 2007. He is currently pursuing the Ph.D. degree with the Science and Research Branch, IAU. He is a faculty member of the Computer Department, Babol Branch, IAU. His current research interests include scheduling problems, optimization, game theory, and mechanism design.

ARUN KUMAR SANGAIAH received the M.E. degree in computer science and engineering from the Government College of Engineering, Anna University, Tirunelveli, India, and the Ph.D. degree in computer science and engineering from VIT University, Vellore, India. He is currently an Associate Professor with the School of Computer Science and Engineering, VIT University. His area of interest includes software engineering, computational intelligence, wireless networks, bio-informatics, and embedded systems. He has authored more than 100 publications in different journals and conference of national and international repute. He also registered a one Indian patent in the area of computational intelligence. His current research interests include global software development, wireless ad hoc and sensor networks, machine learning, cognitive networks, and advances in mobile computing and communications. Besides, he is responsible for Editorial Board Member/Associate Editor of various international journals.