algorithms

MDPI

Article

A Hybrid Grasshopper Optimization Algorithm Applied to the Open Vehicle Routing Problem

Valeria Soto-Mendoza [1,†], Irma García-Calvillo [1,†] and Efraín Ruiz-y-Ruiz [2,*,†] and Jaime Pérez-Terrazas [2,†]

[1] Centro de Investigación en Matemáticas Aplicadas, Universidad Autónoma de Coahuila Edificio "S" Camporredondo, Saltillo, Coahuila 25280, Mexico; vsoto@uadec.edu.mx (V.S.-M.); irma.garcia@uadec.edu.mx (I.G.-C.)

[2] División de Estudios de Posgrado e Investigación, Tecnológico Nacional de México/Instituto Tecnológico de Saltillo. Blvd. Venustiano Carranza 2400, Saltillo, Coahuila 25280, Mexico; jenrique@itsaltillo.edu.mx

* Correspondence: eruiz@itsaltillo.edu.mx; Tel.: +52-646-144-0208

† These authors contributed equally to this work.

Received: 3 March 2020; Accepted: 14 April 2020; Published: 16 April 2020

Abstract: This paper presents a hybrid grasshopper optimization algorithm using a novel decoder and local search to solve instances of the open vehicle routing problem with capacity and distance constraints. The algorithm's decoder first defines the number of vehicles to be used and then it partitions the clients, assigning them to the available routes. The algorithm performs a local search in three neighborhoods after decoding. When a new best solution is found, every route is locally optimized by solving a traveling salesman problem, considering the depot and clients in the route. Three sets containing a total of 30 benchmark problems from the literature were used to test the algorithm. The experiments considered two cases of the problem. In the first, the primary objective is to minimize the total number of vehicles and then the total distance to be traveled. In the second case, the total distance traveled by the vehicles is minimized. The obtained results showed the algorithm's proficient performance. For the first case, the algorithm was able to improve or match the best-known solutions for 21 of the 30 benchmark problems. For the second case, the best-known solutions for 18 of the 30 benchmark problems were found or improved by the algorithm. Finally, a case study from a real-life problem is included.

Keywords: optimization; combinatorial optimization; open vehicle routing problem; grasshopper optimization algorithm

1. Introduction

In recent years, the development of metaheuristic methodologies for the solution of complex optimization problems has increased, both in the continuous and discrete space versions. Some of the most significant metaheuristics are: particle swarm optimization (PSO) [1], ant colony optimization (ACO) [2], genetic algorithms (GA) [3], artificial bee colony algorithm (ABC) [4], tabu search (TS) [5], variable neighborhood search (VNA) [6], the bees algorithm (BA) [7], fish school search (FSS) [8], swallow swarm optimization (SSO) [9], hybrid particle swallow swarm optimization (HPSSO) [10], grasshopper optimization algorithm (GOA) [11], bat algorithm (BAT) [12], whale optimization algorithm (WOA) [13] just to mention a few. Within the metaheuristics mentioned above it can be observed that they are developed using two main ingredients: (1) the basic principles of physics such as velocity, the force of

attraction and acceleration and (2) the behavior of certain groups of animals. There are many applications in which GOA has been used. Among such applications the most recent and relevant are [14–18].

The open vehicle routing problem (OVRP) is a variant of the classic vehicle routing problem (VRP) first introduced in [19]. Many variants of the VRP have been studied in the literature, and recent research focuses on modeling very specialized real-world variants of the VRP. Some of these recent specialized real-world variants can be found in [20–29].

Although new routing problems arise everyday, classic problems such as the OVRP are still studied thanks to their potential applications. The research presented in this paper was motivated by the case study presented in Section 5, which required the solution of OVRP instances.

The OVRP was formally introduced in [30], although the first work related to the problem was first presented in [31] for an air express courier problem. The OVRP is an NP-Hard combinatorial optimization problem (as mentioned in [32]), which makes it difficult to solve by exact methods. Therefore, many heuristic approaches have been developed to solve the problem. Among the most important heuristic algorithms used to solve the OVRP are those in [32–39].

The applications of the OVRP are not limited to the context of vehicle route design. In [31], the OVRP is used to solve an air express courier problem, and in [40], the train services planning at the Channel tunnel is made by solving an OVRP.

In this work, a novel hybrid grasshopper optimization algorithm with local search (HGOALS) method is used to solve instances of the open vehicle routing problem. To our knowledge, GOA has not been previously used for solving the OVRP. In fact, there is not much literature on GOA in the context of vehicle routing problems. The proposed solution method is based on the grasshopper optimization algorithm (GOA) and uses local search to improve solutions and accelerate the optimization process. The GOA is a swarm optimization algorithm that mimics the behavior of a grasshopper swarm to solve optimization problems.

The HGOALS combines the GOA framework with a decodification procedure and local search to obtain solutions for the OVRP instances. The decodification procedure incorporates a feasibility recovery phase in case the decoded solutions are infeasible. Once the solutions are feasible, the best ones are selected for improvement using local search. Every time a new best solution is found, it is subjected to an improvement phase. The algorithm finishes when two given stop criteria are reached.

To demonstrate the efficiency of the proposed methodology, experiments were carried out using 30 benchmark problems taken from the scientific literature and the results were compared against previous algorithms. The experiments considered two objective functions for the problem. The first is the hierarchical objective function, which has as a first goal the minimization of the total number of vehicles to be used, and as a second goal the minimization of the total traveling cost of such vehicles. In the second objective function, only the total traveling cost is minimized. The results show that the proposed methodology is competitive. Additionally, the algorithm was utilized to solve a case study from a security company, obtaining excellent results. The rest of this paper is organized as follows: Section 2 describes the OVRP, while Section 3 describes the proposed algorithm. The computational experiments are reported in Section 4, where the used benchmark instances, the corresponding numerical results and a numerical comparison concerning other algorithms are presented. Finally, conclusions and future research are discussed in Section 6.

2. Problem Description

As mentioned before, two types of objective functions have been taken from the literature for the problem: (i) the hierarchical one (minimize the number of routes and then minimize the total traveling cost) and (ii) the minimization of the total traveling cost.

Consider a complete directed graph $G = (V, A)$, with $V = \{0, 1, \ldots, n\}$ the set of vertices and A the set of arcs. The reason to consider a directed graph is that in real-life applications, cost (distance) matrices are not symmetric (See Section 5). Notice that vertex 0 (depot) is the departure point of all vehicles. Each vertex (except the depot) $i \in V \setminus \{0\}$, has an associated demand d_i. Also, each arc $a = (i, j) \in A$, has an associated cost $c_{ij} > 0$. Feasible solutions to the OVRP are open routes originated at the depot, that satisfy capacity and route length constraints.

Before formally defining the OVRP, some notation is introduced. Let $R \subset A$ denote a given set of open routes of G originated at the depot (0):

- $V^+ = \{1, \ldots, n\}$, the set of vertices without the depot.
- Cost of R, $c(R) = \sum_{a=(i,j)\in R} c_{ij}$.
- st, the time required to service any client.
- First vertex, a vertex directly connected to 0 in R ($a = (0, i) \in R$).
- Subroute ($R_i \subseteq R$), an open route in R originated at vertex $i \in V$.
- s-route ($R_i \subseteq R : a = (0, i) \in R$), a subroute originated at i, where i is the first vertex of R_i.
- $V(R_i) \subset V$, the set of vertices that are part of subroute R_i.
- $d(R_i) = \sum_{j\in V(R_i)} d_j$, the sum of the demands of the vertices in R_i.
- $l(R_i) = \sum_{a=(i,j)\in R_i} (c_{ij} + st)$, the total length of R_i.

The capacity constraints for the OVRP impose that for a given open route originated at 0 the total demand of any subroute can not exceed a given value $Q > 0$. That is $d(R_i) \leq Q$ for any subroute R_i of R. Open routes are also subjected to length constraints, which impose that route duration can not exceed a given value $L > 0$. That is $l(R_i) \leq L$ for any subroute R_i of R.

The OVRP can be defined as the problem of finding a set of open routes originated at the depot, visiting all clients and satisfying the capacity and route length constraints while minimizing one of the above mentioned objective functions.

Mathematical Formulation

As mentioned before the OVRP is a variant of the VRP and formulations for the VRP can be adapted to solve the OVRP. Following a formulation for the OVRP is presented. This formulation is useful for solving symmetric and asymmetric OVRP and considers capacity and route length constraints. It is based on a classic formulation of the VRP and the Miller–Tucker–Zemlin subtour elimination constraints for the VRP [41]. Therefore, the OVRP can be formulated using the following variables:

$$x_{ij} = \begin{cases} 1, & \text{if arc } a = (i, j) \text{ is part of the solution,} \quad i \in V, j \in V^+, \\ 0, & \text{otherwise.} \end{cases}$$

$$u_i = \text{Load on the vehicle after visiting vertex i}$$

$$y_i = \text{Length travelled after visiting vertex i}$$

The hierarchical objective function is defined using the variables mentioned above, as follows:

$$\min \sum_{j\in V^+, i\neq j} M x_{0j} + \sum_{i\in V^+} \sum_{j\in V^+, i\neq j} c_{ij} x_{ij} \tag{1}$$

where M is a large number. The first term in Equation (1) minimizes the total number of vehicles, while the second term minimizes the total traveling cost.

The second objective function only minimizes the total traveling cost and is defined as follows:

$$\min \sum_{i \in V} \sum_{j \in V^+, i \neq j} c_{ij} x_{ij} \tag{2}$$

The following constraints are required to complete the model for the OVRP:

$$\sum_{i \in V, i \neq j} x_{ij} = 1, \qquad \forall j \in V^+ \tag{3}$$

$$\sum_{i \in V_0, i \neq j} x_{ji} \leq 1, \qquad \forall j \in V^+ \tag{4}$$

$$u_i - u_j + Q x_{ij} \leq Q - d_j, \qquad \forall i \; inV, j \in V^+, i \neq j, d_i + d_j \leq Q \tag{5}$$

$$y_i - y_j + Q x_{ij} \leq Q - st, \qquad \forall i \; inV, j \in V^+, i \neq j \tag{6}$$

$$d_i \leq u_i \leq Q, \qquad \forall i \in V, j \in V^+ \tag{7}$$

$$st \leq y_i \leq L, \qquad \forall i \in V, j \in V^+ \tag{8}$$

The constraints in Equation (3) guarantee that all clients are visited, and the constraints in Equation (4) guarantee that routes leave the client or finish the route at the client. Route capacity is controlled with the constraints in Equations (5) and (6). The constraints in Equations (7) and (8) control the route length. Note that the constraints in Equation (5) are the Miller–Tucker–Zemlin subtour elimination constraints for the VRP [41], and the constraints of Equation (7) are the same constraints adapted to control route length.

3. Hybrid Grasshopper Optimization Algorithm with Local Search

This section describes the proposed hybrid grasshopper optimization algorithm with local search (HGOALS) for solving OVRP's instances. The grasshopper optimization algorithm (GOA) has been proposed initially in [11].

The behavior of a grasshopper's swarm of size P can be simulated using $X_i = r_3 A_i + r_2 G_i + r_1 S_i$, where X_i is the current position of the $i-th$ member of the swarm, A_i the effect of the wind on the $i-th$ grasshoppers movement, G_i the gravity force and S_i the social interaction between the *i-th* grasshopper and the rest of the members of the swarm. The coefficients r_1, r_2 and r_3 are random numbers in the interval $[0, 1]$ used to introduce random behavior among the members of the swarm. As mentioned in [11], the problem with this model is that grasshoppers tend to stay in a comfort zone, which is not useful when solving optimization problems. Therefore the authors proposed to use:

$$X_i = c_1 \left[\sum_{j=1, j \neq i}^{P} c_2 \frac{ub - lb}{2} s\left(|X_j - X_i| \right) \frac{X_j - X_i}{dist_{ij}} \right] + T \tag{9}$$

In this equation, ub and lb are vectors containing respectively, the upper and lower bounds for the variables considered for optimization. T is the target solution, which is the best solution found so far by the algorithm, $dist_{ij}$ is the distance between grasshoppers i and j. Finally, the social interaction between grasshoppers is computed using $s(|X_j - X_i|) = f \exp(-r/l) - \exp(-r)$, where f is the intensity of attraction, and l the attractive length scale. Constant c_1 determines how likely is the grasshopper to explore around the target T (equivalent to w in the PSO algorithm), balancing the exploration and exploitation of the area around T. Constant c_2 defines the attraction, repulsion and comfort zone among the

grasshoppers. For a problem of dimension n, the position of the grasshopper on dimension h ($h > 0, h \leq n$) is given by:

$$X_i^h = c_1 \left[\sum_{j=1, j \neq i}^{P} c_2 \frac{ub^h - lb^h}{2} s\left(|X_j^h - X_i^h| \right) \frac{X_j - X_i}{dist_{ij}} \right] + T^h \quad (10)$$

The movement of the grasshopper can be simulated in a multidimensional space using Equation (10). For further details about the GOA, the reader is encouraged to read [11].

Since the GOA is a continuous optimization algorithm, it is necessary to implement a decodification procedure in order to solve non-continuous optimization problems. Therefore, a decodification procedure is required to transform the continuous values from the GOA into solutions of the non-continuous problem.

3.1. HGOALS Description

A mentioned previously, HGOALS combines the GOA framework with a decodification procedure and local search to solve OVRP instances. Figure 1 shows a flow diagram for the HGOALS. The proposed algorithm receives as an input graph $G = (V, A)$, the associated cost matrix c, the service time parameter st, the vehicle capacity parameter Q, the maximum length parameter L, the grasshopper population size P and the grasshopper dimension n. HGOALS has three essential components: (i) The route decoder (decodification procedure), (ii) the local search and (iii) the improvement phase. To facilitate comprehension the above mentioned components are described first, and the complete process of the proposed HGOALS algorithm is explained after.

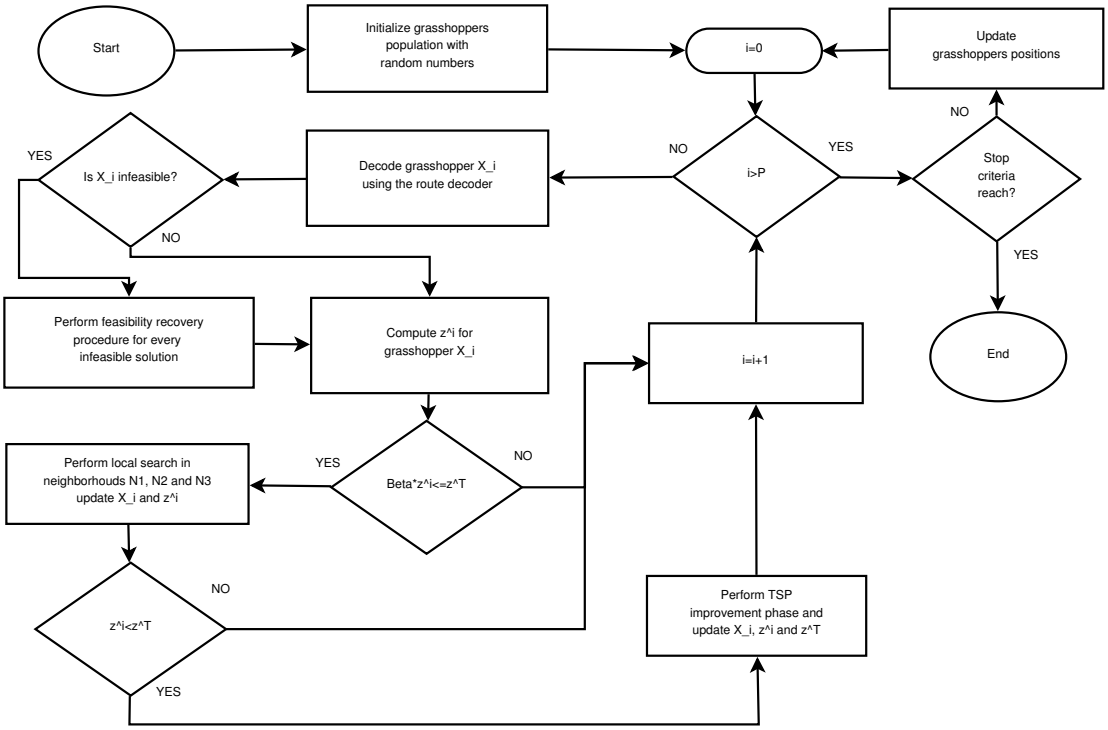


Figure 1. Flow diagram for the hybrid grasshopper optimization algorithm with local search (HGOALS).

3.2. The Route Decoder

As mentioned before, the HGOALS requires the implementation of a decoder. Other decoders for similar problems have been previously proposed. The objective of proposing a new decoder is to find

better results than those obtained by previous decoders and other metaheuristics. In this section, the new decoder is described.

The route decoder (RD) requires as input the associated cost to every arc $c_{ij}, \mid a \in A$, the random-key vector $\mathcal{X}$ and the demand vector d. The random-key vector $\mathcal{X}$ contains $n+1$ fractional values. $\mathcal{X}_0$, the first value in vector $\mathcal{X}$, is used to define the total number of vehicles available in the solution. The other n values are used to assign vertices to different vehicles. The decoder determines the total number of vehicles, by first computing the minimum number of vehicles required to deliver all the clients demand using:

$$MV = \left\lceil \frac{d(V^+)}{Q} \right\rceil$$

where:

$$d(V^+) = \sum_{i \in V^+} d_i$$

Using the random-key vector value $\mathcal{X}_0$ the number of available vehicles in the solution is given by:

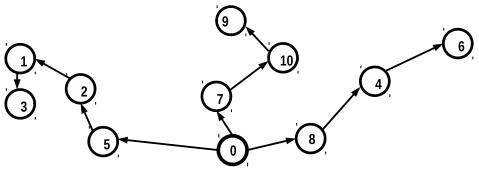
$$NV = \lceil MV(1 + \mathcal{X}_0) \rceil$$

Then the interval $[0, 1]$ is divided into NV subintervals. Each client is assigned to a vehicle using these subintervals and the keys associated with each client. Once all clients have been assigned to a vehicle, they are sorted in ascending order according to their key. The routes are built using the sorted keys. Figure 2 illustrates an example of how the proposed decodification procedure works. In this example, using the demand vector d, the capacity parameter $Q = 20$ and the first key-value $\mathcal{X}_0$ = 0.28, the minimum and the available number of vehicles are obtained: $MV = 2$ and $NV = 3$ respectively. Therefore, the interval $[0, 1]$ is divided into three subintervals which are used to assign clients to vehicles. Clients 4, 6 and 8 are assigned to vehicle number 1; clients 1, 2, 3 and 5 are assigned to vehicle 2; and clients 7, 9 and 10 are assigned to vehicle number 3. For vehicle 1, the assigned vertices are transformed into the route $0-8-4-6$; for vehicle 2, the resulting route is $0-5-2-1-3$. Finally, for vehicle 3, the resulting route is $0-7-10-9$.

Client	1	2	3	4	5	6	7	8	9	10
Demand	2	3	1	10	2	4	3	4	5	4

$$Q=20 \qquad MV=\left\lceil \frac{38}{20} \right\rceil=2 \qquad NV=\lceil 2(1+0.28) \rceil=3$$

Vertex	0	1	2	3	4	5	6	7	8	9	10
Key	0.28	0.52	0.47	0.63	0.23	0.36	0.28	0.81	0.17	0.93	0.88
Route assigned	-	2	2	2	1	2	1	3	1	3	3
Position in route	-	3	2	4	2	1	3	1	1	3	2



$d(R_8)=18$

$d(R_7)=12$

$d(R_5)=8$

Figure 2. Route decoder example.

It is relevant to notice that this decodification/codification method can lead to infeasible solutions. In Figure 3, an example of an infeasible solution is presented. The violation is made over the capacity and maximum distance constraints since for s-route R_8, $d(R_8) > Q$ and $l(R_8) > L$.

Client	1	2	3	4	5	6	7	8	9	10
Demand	2	3	1	10	2	4	3	4	5	4

$$Q=20 \qquad MV=\left\lceil \frac{38}{20} \right\rceil=2 \qquad NV=\left\lceil 2(1+0.28) \right\rceil=3$$

Vertex	0	1	2	3	4	5	6	7	8	9	10
Key	0.28	0.52	0.47	0.63	0.23	0.36	0.28	0.81	0.17	0.93	0.31
Route assigned	-	2	2	2	1	2	1	3	1	3	1
Position in route	-	3	2	4	2	1	3	1	1	3	4

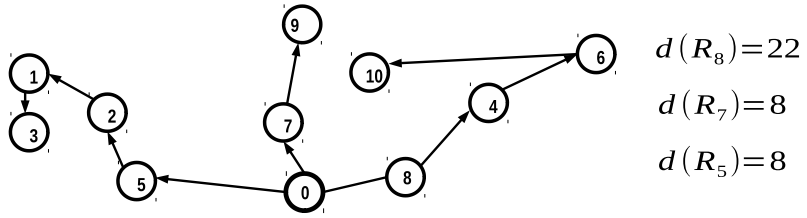


Figure 3. Infeasible solution example.

Whenever a solution is infeasible after decoding, it enters the feasibility recovery procedure. The feasibility recovery procedure reassigns subroutes or vertices from the infeasible route to other routes with available capacity and without violating the maximum length constraint. If there are no suitable routes for reassigning a given subroute or vertex, a new route is created by connecting such subroute or vertex directly to the depot. The reassignment is made by replacing arcs on the solution, using the cheapest available that reduces or eliminates the route's infeasibility. Figure 4 shows how the feasibility recovery procedure repairs an infeasible solution. In this example, s-route R_8 violates the capacity and maximum distance constraints ($d(R_8) > Q$ and $l(R_8) > L$). For recovering feasibility, subroute R_5 is removed from R_8. Since annexing R_5 to s-route R_7 will violate the maximum length constraint, R_5 is directly connected to the depot. At this point, R_8 only violates the capacity constraint. Therefore, vertex 10 is removed from R_8 and annexed to s-route R_7, leading to a feasible solution.

3.3. Local Search

Once a solution is feasible, it is improved through a local search phase if its objective function value z^i, is close to the objective function value z^T of the current best solution T ($\beta \cdot z^i <= z^T$, $0.75 <= \beta <= 0.95$). In the local search, three neighborhoods are explored: (i) $N1$ vertex exchange, (ii) $N2$ vertex reassignment and (iii) $N3$ route merging. Consider two vertices i and j belonging respectively, to two different s-routes k and m. In the $N1$ neighborhood, vertex i is removed from s-route k and connected to s-routes m, while vertex j is removed from s-route m and connected to s-routes k (Figure 5). The $N2$ neighborhood is explored by removing vertex i from s-routes k and connecting i to s-route m (Figure 6). Finally, s-routes k and m

are merged into one *s*-route (*k*) in the $N3$ neighborhood (Figure 7). The three neighborhoods are explored sequentially ($N1 \rightarrow N2 \rightarrow N3$) until no further improvement is found.

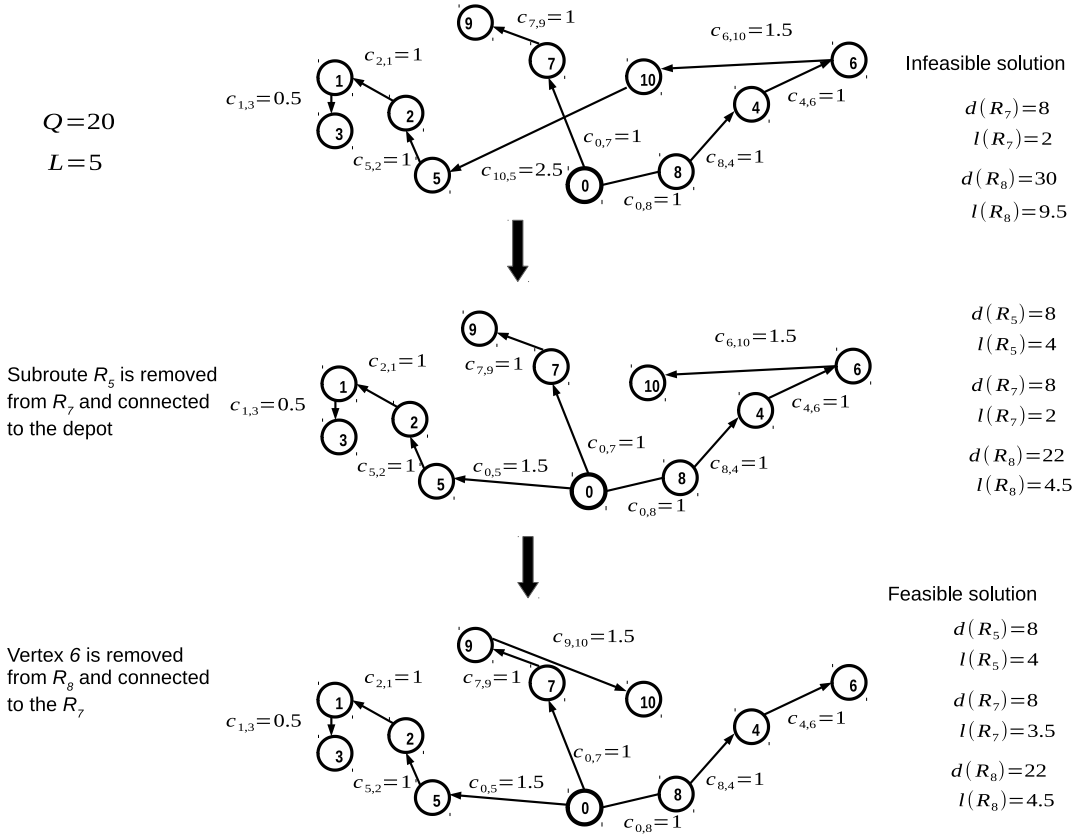


Figure 4. Feasibility recovery procedure example of an infeasible solution.

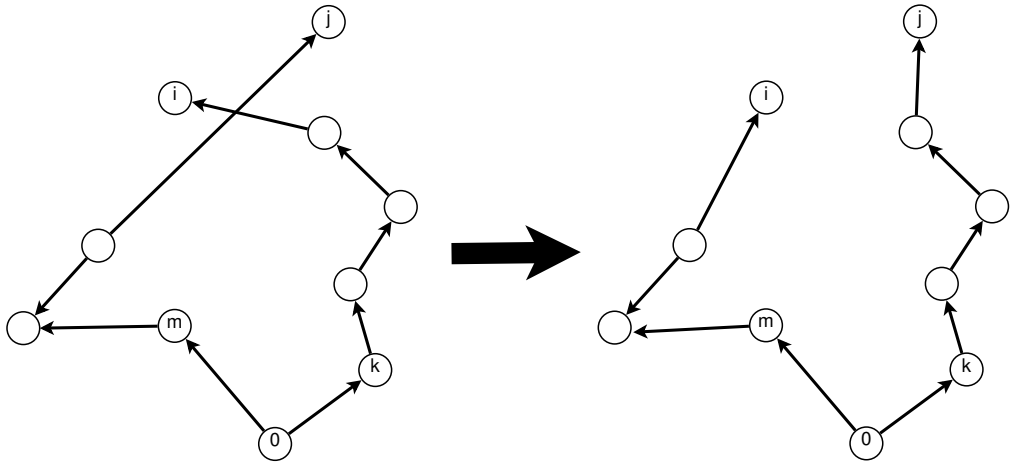


Figure 5. Neighborhood $N1$, vertex exchange.

3.4. Improvement of Best Solutions

If after the local search, the algorithm finds a new best solution, such a solution is subjected to an improvement phase. Consider the subset of vertices $S_R i = V(R_i) \cup \{0\}$ that contains the vertices assigned to a given *s*-route R_i and the depot. Using $S_R i$ and the associated subset of arcs $A_R i \subset A, a =$

$(i,j) \mid i \in S_R i, j \in S_R i$, the sub-graph $G_R = (S_R i, A_R i)$ is built. Associated costs c_{ij} for every arc $a = (i,j) \in A_R i \mid i \in S_R i, j \in S_R i, j \neq 0$ are obtained from the original problem. The cost for the remaining arcs $a = (i,0), i \in S_R i$ are substituted using $c_{i0} = 0, \mid i \in S_R i$. Using G_R and its associated costs, a traveling salesman problem (TSP) can be solved in order to minimize the cost of route R_i since every route respects the capacity and maximum length constraints. Notice that the solution obtained by the TSP using graph G_R and its associated costs are equivalent to an open route, since the costs for returning to the depot are equal to zero. Therefore, a TSP problem is solved for every s-route using the formulation in [41], leading to a locally optimized solution.

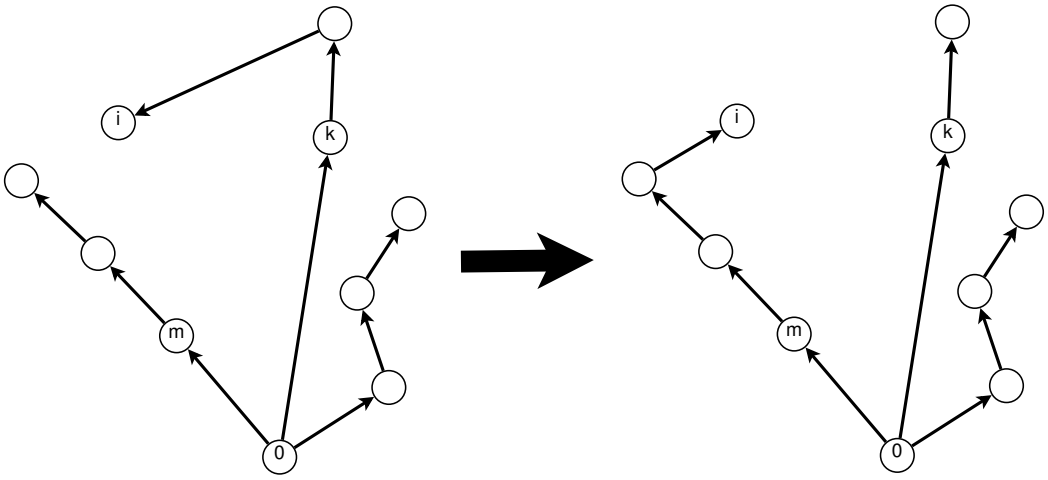


Figure 6. Neighborhood $N2$, vertex reassignment.

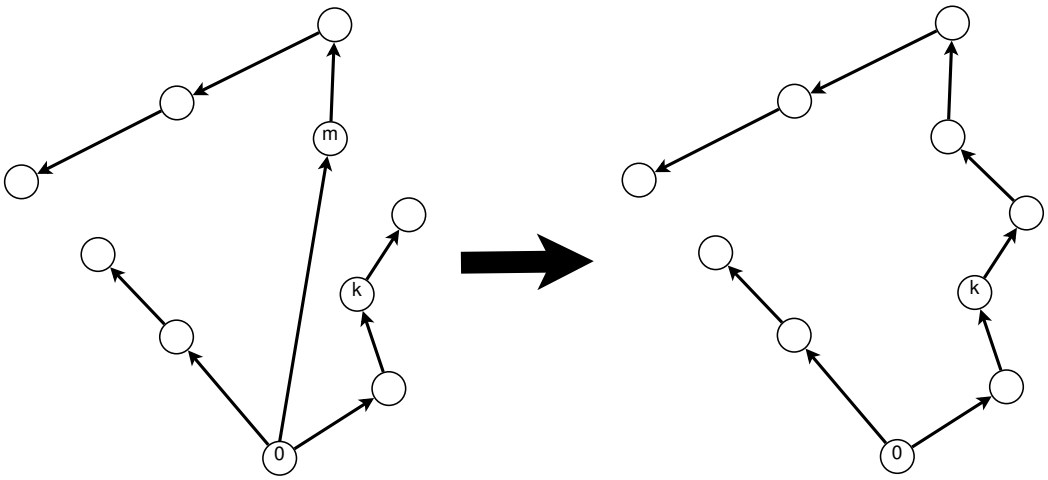


Figure 7. Neighborhood $N3$, route merging.

Since all HGOALS components have been explained, it is appropriate to describe the algorithm with the aid of the flow diagram presented in Figure 1. The first step is to initialize the P grasshoppers with $n+1$ continuous random values, and the target solution T's objective function is initialized $z^T = \infty$. Then, the algorithm enters a loop to transform grasshoppers into OVRP solutions. As every grasshopper X_i enters the loop, the route decoder (comprehensively explained in Section 3.2) transforms the grasshopper's continuous values to obtain OVRP solutions in a non-continuous space. If the obtained solution is infeasible, then it enters the feasibility recovery procedure. Once the solution is feasible, its objective function z_i is computed. If the solution is among the best solutions ($\beta \cdot z^i <= z^T$, $0.75 <= \beta <= 0.95$) then

it enters local search. In the local search, three neighborhoods $N1$, $N2$ and $N3$ are explored sequentially. After the local search phase, the solution's objective function z^i is computed. If $z^i < z^T$ then T is updated with X_i. The loop finishes once all grasshoppers have been decoded. At this point, the grasshoppers' positions are updated using Equation (10), and the stop criteria are checked. The algorithm finishes when one stop criterion is reached.

4. Computational Experiments

In this Section, test instances are described first, and second, the results from the computational experiments for both of the objective functions previously mentioned are presented.

4.1. Test Instances

For testing the proposed algorithm, three sets of instances were used: group C, group O and group K. The first two sets of instances have been widely used in previous research, while the third set was only used in [42,43]. Table 1 shows the characteristics for every group of instances; n specifies the size of the instances in the group (excluding the depot), Q specifies the vehicle's capacity and L specifies the range for the maximum length allowed for the routes.

Table 1. Test instances.

Instance	*n*	*Q*	*L*
C1–C5	50–200	140–200	∞
C6–C10	50–200	140–200	160–230
C11–C12	100–120	200	∞
C13–C14	100–120	200	720–1040
O1–O8	200–440	550–1000	∞
K01–K08	200–440	550–1000	650–1800

Table 1 shows that instances C1–C5, C11–C12 and O1–O8 do not consider route length constraints, while instances C6–C10, C13–C14 and K01–K08 do. For instances C1–C10, O1–O8 and K01–K08 the depot is located at the center of the customers, while for instances C11–C14 the depot is located on an extreme side. Additionally, for instances C11–C14, clients are distributed in clusters over the plane. For instances C1–C10, clients are randomly distributed around the depot. Finally, clients for K and O instances follow a ring-like distribution over the plane.

4.2. Implementation Details

The proposed algorithm was implemented in Matlab. Two criteria were used to stop the algorithm. The first was set to a maximum running CPU time of 1000 s, while the second was to reach 25 iterations without improving the current best solution found. The machine used for the experiments was running Linux with an Intel Core2 Duo, a CPU clock speed of 3.1 GHz and 4 GB of RAM.

4.3. Results with the Hierarchical Objective Function

Table 2 presents the results obtained by the HGOALS for the benchmark problems using a hierarchical objective function. In this case, the first goal is to minimize the number of vehicles, and the second, the total traveling cost. Column n shows the size of the instance, Q shows the vehicles capacity, NV presents the number of vehicles and BKS the best-known solution value. Column Avg shows the average solution value after five runs, $Best$ presents the value of the best solution value after five runs, $\%gap$ shows the percentage

difference between the best-known solution value and the best solution value obtained by the HGOALS, and CPU presents the average CPU time per run in seconds, after five runs. Row Avg presents the average for columns CPU and $\%gap$. Results in bold format show that the algorithm found the best know solution for the instance. Results in bold and underlined show the algorithm improved the best-known solution for the benchmark problem. The symbol '**' is used to indicate that the algorithm found solutions with a new minimum number of vehicles. When the number of vehicles from the best-known solution is different of the best solution obtained by the HGOALS, the number of vehicles used by the best-known solution is shown in column NV. The results show that the algorithm was able to improve the best-known solutions for seven instances in group C, five on group K and one in group O.

Table 2. Results obtained by the HGOALS for all the test instances (hierarchical objective function).

Instance	n	Q	NV	BKS	Avg	$Best$	CPU	$\%gap$
C1	50	160	5	416.06	416.95	**416.06**	3.96	0.21
C2	75	140	10	567.14	563.40	**560.19**	20.31	−1.23
C3	100	200	8	639.74	643.65	**639.74**	23.28	0.00
C4	150	200	12	733.13	734.16	733.64	94.81	0.07
C5	199	200	16	893.39	885.25	895.49	158.32	0.24
C6	50	160	5 (6)	412.96	417.75	**422.25** **	5.70	2.25
C7	75	140	10	583.19	567.96	**567.77**	9.45	−2.64
C8	100	200	9	644.63	647.74	646.75	24.12	0.33
C9	150	200	13	757.69	743.19	**742.35**	135.98	−2.02
C10	199	200	17	874.71	878.73	**873.70**	243.11	−0.12
C11	120	200	7	682.12	695.83	690.194	62.00	1.18
C12	100	200	10	534.24	535.99	**534.24**	30.37	0.00
C13	120	200	10 (11)	899.16	877.21	**865.80** **	122.00	−3.71
C14	100	200	10 (11)	591.87	557.74	**556.32** **	39.06	−6.01
Avg							69.46	-0.82
O1	200	900	5	6018.52	6035.89	**6018.52**	519.37	0.00
O2	240	550	9	4557.38	4571.43	4560.29	820.07	0.06
O3	280	900	7	7731.00	7735.45	7734.70	825.82	0.05
O4	320	700	9(10)	7251.74	7274.23	**7367.91** **	740.80	1.60
O5	360	900	8	9193.15	9198.73	**9193.15**	919.40	0.00
O6	400	900	9	9793.72	9807.46	**9793.72**	1000.00	0.00
O7	440	900	10	10,344.37	10,387.18	10,363.80	1000.00	0.19
O8	400	1000	10	12,415.36	12,465.65	12,420.81	943.79	0.04
Avg							846.16	0.24
K01	240	550	9	4629.21	4642.75	**4622.58**	760.49	−0.14
K02	320	700	9	7285.98	7303.10	7293.50	1000.00	0.10
K03	400	900	10	9837.79	9879.08	**9831.07**	1000.00	−0.07
K04	480	1000	10	12,431.50	12,475.34	**12,422.20**	1000.00	−0.07
K05	200	900	5	6342.66	6052.56	**6041.33**	595.51	−4.75
K06	280	900	7	7731.21	7734.46	**7731.21**	756.28	0.00
K07	360	900	9	9188.11	9172.43	**9162.29**	1000.00	−0.28
K08	440	900	10	10,488.50	10,472.46	**10,457.60**	1000.00	−0.29
Avg							889.04	−0.69

Therefore, the HGOALS is capable of improving the best-known solution for 15 of the 30 benchmark problems. Notice that 11 of the improved best-known solutions where from instances considering route length constraints. The algorithm also finds the best-known solutions for the other five instances. Other relevant results are the new best solutions for instances C6, C13, C14 and O4 that use fewer vehicles

(5, 10, 10 and 9, respectively) than the solutions found in previous research (6, 11, 11 and 10). Finally, the algorithm struggles with the C11 instance. The average percentage deviation from the best-known solution (gap) is the worst for all benchmark problems (1.18%). The reason is the cluster distribution of the clients and the tendency of the feasibility recovery procedure to create more routes than the minimum required. Note that instance C6 has a higher gap, but it uses fewer vehicles than the best-known solution.

The CPU times increase significantly as the benchmark problem size increases. The algorithm reached the 1000 s limit for seven of the benchmark instances, two from set O and five from set K. The algorithm performs the worst in the set O with an average deviation from the best-known solution of 0.24. For group C, the algorithm performs the best having an average deviation from the best-known solutions of −0.82, while for group K this metric is −0.69.

Table 3 compares the proposed algorithm with the results from previous research. Considering all instances in group C, the HGOALS outperforms the other algorithms. The Broad Local Search Algorithm (BLSA) performs better than the HGOALS in instances without route length constraints in this group. The HGOALS performs the best in instances with route length constraints, improving the best-known solutions for six of the seven instances. The HGOALS obtains new best solutions for C2, C6, C7, C9, C10, C13 and C14. For instances C6, C13 and C14 the algorithm found solutions using one less vehicle. Finally, it is relevant to mention that the algorithm struggles with C11 instance obtaining the largest gap (1.18). The latter instance has a cluster like distribution.

Table 3. Comparison between the HGOALS results and previous research for instances in group C (hierarchical objective function).

Instance	*n*	*Q*	*BKS*	*NV*	HGOALS	BBMO [37]	BLSA [38]	HBMO [44]	ORTR [35]
C1	50	160	416.06	5	**416.06**	**416.06**	**416.06**	**416.06**	**416.06**
C2	75	140	567.14	10	**560.19**	567.14	567.14	567.14	567.14
C3	100	200	639.74	8	**639.74**	**639.74**	**639.74**	640.25	**639.74**
C4	150	200	733.13	12	733.64	735.18	**733.13**	738.49	**733.13**
C5	199	200	893.39	16	895.49	895.19	**893.39**	902.17	924.96
C6	50	160	412.96	6	**422.25 (5)**	412.96	-	412.96	412.96
C7	75	140	583.19	10	**567.77**	583.19	-	583.19	568.49
C8	100	200	644.63	9	646.75	**644.63**	-	644.63	**644.63**
C9	150	200	757.69	13	**742.35**	759.35	-	765.95	**756.38**
C10	199	200	874.71	17	**873.70**	879.38	-	884.28	876.02
C11	120	200	682.12	7	690.194	**682.12**	**682.12**	683.15	682.54
C12	100	200	534.24	10	**534.24**	535.28	**534.24**	536.37	**534.24**
C13	120	200	899.16	11	**865.80 (10)**	904.04	-	905.18	896.50
C14	100	200	591.87	11	**556.32 (10)**	592.16	-	593.95	591.87
Instance	***n***	***Q***	***BKS***	**MVNS [34]**	**HES [36]**	**TSB [32]**	**ALNS [45]**	**ACO [46]**	**HPSO [47]**
C1	50	160	416.06	5	**416.06**	**416.06**	**416.06**	**416.06**	**416.06**
C2	75	140	567.14	10	567.14	567.14	574.05	567.14	567.14
C3	100	200	639.74	8	**639.74**	**639.74**	641.60	641.76	639.74
C4	150	200	733.13	12	**733.13**	**733.13**	740.80	**733.13**	**733.13**
C5	199	200	893.39	16	905.96	894.11	953.40	896.08	977.95
C6	50	160	412.96	6	412.96	412.96	412.96	412.96	412.96
C7	75	140	583.19	10	596.47	584.15	634.54	583.19	568.49
C8	100	200	644.63	9	**644.63**	**644.63**	**644.63**	645.16	**644.63**
C9	150	200	757.69	13	760.06	764.56	785.20	757.84	758.97
C10	199	200	874.71	17	**875.67**	888.46	884.63	**875.67**	886.08
C11	120	200	682.12	7	682.12	682.12	683.40	682.12	683.11
C12	100	200	534.24	10	**534.24**	**534.24**	535.10	**534.24**	**534.24**
C13	120	200	899.16	11	904.04	910.26	943.66	909.80	904.01
C14	100	200	591.87	11	591.87	591.87	597.30	591.87	591.87

The comparison of the results with previous research for instances in group O is made in Table 4. In this set of instances, the BLSA performs the best obtaining the best-known solutions for six of the eight benchmark problems. On the other hand, the HGOALS performs better than the other algorithms (except BLSA), and finds a new best solution for instance O4 using nine vehicles instead of ten. Notice that test problems in this set do not consider route length constraints.

Table 4. Comparison between route decoder and previous research for instances in group O (hierarchical objective function).

Instance	*n*	*Q*	*NV*	*BKS*	HGOALS	ORTR [35]	BLSA [38]
O1	200	900	5	6018.52	**6018.52**	**6018.52**	**6018.52**
O2	240	550	9	4557.38	4560.29	4584.55	**4557.38**
O3	280	900	7	7731.00	7734.70	7732.85	**7731.00**
O4	320	700	10	7251.74	**7367.91 **(9)**	7291.89	7253.20
O5	360	900	8	9193.15	**9193.15**	9197.61	**9193.15**
O6	400	900	9	9793.72	**9793.72**	9803.80	**9793.72**
O7	440	900	10	10,344.37	10,363.80	10,374.97	10347.70
O8	400	1000	10	12,415.36	12,420.81	12,429.56	**12,415.36**
Instance	***n***	***Q***	***NV***	***BKS***	**HES [36]**	**HBMO [44]**	**BBMO [37]**
O1	200	900	5	6018.52	**6018.52**	6023.48	6021.11
O2	240	550	9	4557.38	4583.70	4561.18	**4557.38**
O3	280	900	7	7731.00	7733.77	7745.16	7735.14
O4	320	700	10	7251.74	7271.24	7287.49	7267.18
O5	360	900	8	9193.15	9254.15	9201.25	9198.25
O6	400	900	9	9793.72	9821.09	9809.48	9798.19
O7	440	900	10	10,344.37	10,363.40	10,401.24	10,351.18
O8	400	1000	10	12,415.36	12,428.20	12,429.57	12,418.57

Finally, Table 5 compares results for instances in group K. The comparison is made with upper bounds obtained by the GRASP algorithm proposed in [42] and the BRKGA in [48]. The results show that the HGOALS outperforms the other two metaheuristics by obtaining six new best solutions for the eight instances in the set. In this set, all benchmark problems consider route length constraints.

Table 5. Comparison between route decoder results and previous research for instances in group K (hierarchical objective function).

Instance	*n*	*Q*	*NV*	*BKS*	HGOALS	GRASP [42]	BRKGA [43]
K01	240	550	9	4629.21	**4622.58**	4790.31	4629.21
K02	320	700	10	7285.98	7293.50	8407.07	**7285.98**
K03	400	900	10	9837.79	**9831.07**	11,286.40	9837.78
K04	480	1000	10	12,431.50	**12,422.20**	14,325.87	12,431.50
K05	200	900	5	6342.66	**6041.33**	6342.66	6004.65 *(12)
K06	280	900	7	7731.21	**7731.21**	8557.82	**7731.21**
K07	360	900	9	9188.11	**9162.29**	10,624.48	9188.11
K08	440	900	10	10,488.50	**10,457.60**	12,129.07	10,488.50

4.4. Results with the Traveling Cost Minimization Objective Function

The obtained results by the HGOALS for the benchmark problems with the second objective function of minimizing the total traveling cost are presented in Table 6. The results show that the algorithm was able to improve three of the 14 instances of group C, one of group O and five on group K. The HGOALS also reaches the best-known solutions for five instances in group C and one for group K.

Table 6. Results obtained by the HGOALS for all the test instances (minimizing traveling cost).

Instance	n	Q	BKS	Avg	$Best$	CPU	$\%gap$
C1	50	160	412.96	412.96	**412.96**	3.12	0.00
C2	75	140	564.06	564.06	**564.06**	4.97	0.00
C3	100	200	639.26	644.36	**639.26**	21.84	0.00
C4	150	200	733.13	733.84	733.84	54.50	0.10
C5	199	200	869.00	876.33	875.49	147.48	0.75
C6	50	160	412.96	412.96	**412.96**	2.94	0.00
C7	75	140	567.59	567.59	**567.59**	4.91	0.00
C8	100	200	644.63	646.35	644.63	20.88	0.00
C9	150	200	743.51	743.22	**742.35**	71.62	−0.16
C10	199	200	873.89	875.83	**873.69**	140.66	−0.02
C11	120	200	676.40	679.80	678.70	85.84	0.34
C12	100	200	534.24	536.20	536.20	10.22	0.37
C13	120	200	862.74	849.64	**841.92**	122.42	−2.41
C14	100	200	554.67	554.67	**554.67**	11.79	0.00
Avg						50.23	−0.07
O1	200	900	5988.35	6003.08	6003.08	277.41	0.25
O2	240	550	4549.46	4557.38	4550.21	603.40	0.02
O3	280	900	7731.00	7735.43	**7731.00**	1000.00	0.00
O4	320	700	7251.30	7254.18	7253.20	1000.00	0.03
O5	360	900	9152.47	9163.25	**9132.88**	1000.00	−0.21
O6	400	900	9793.72	9803.41	**9793.72**	1000.00	0.00
O7	440	900	10,344.37	10,349.57	10,347.70	1000.00	0.03
O8	400	1000	12,412.26	12,426.58	12,415.36	1000.00	0.02
Avg						860.10	0.02
K01	240	550	4629.21	4637.86	**4622.58**	901.95	−0.14
K02	320	700	7285.98	7304.73	7296.15	957.26	0.14
K03	400	900	9837.78	9848.97	**9831.07**	817.50	−0.07
K04	480	1000	12,431.50	12,449.32	**12,422.20**	806.17	−0.07
K05	200	900	6004.74	6006.72	**6004.74**	275.42	0.00
K06	280	900	7731.21	7735.93	7735.16	807.58	0.05
K07	360	900	9188.11	9183.54	**9162.29**	1000.00	−0.28
K08	440	900	10,488.50	10,486.70	**10,457.60**	873.22	−0.29
Avg						804.89	−0.08

The results show that the proposed algorithm is capable of improving the best-known solution in nine of the 30 instances used to test the algorithm. Notice that eight of these instances consider route length constraints. It is also capable of finding the best-known solutions in other nine test instances. In brief, the HGOALS matches or improves the best-known solutions for 18 of the 30 test instances. On the other hand, the CPU times required by the algorithm reach 1000 s for six instances in group O. It is also in this group where the algorithm performs the worst with an average deviation from the best solution of 0.02. For group K, the algorithm performs the best having an average deviation from the best solution of −0.08, while for group C this metric is of −0.07.

Table 7 compares the proposed algorithm with previous research for benchmark problems in group C. For instances with no route length constraints, the HGOALS performs worse than the BLSA and BRKGA but better than the other algorithms. For the benchmark problems including route length constraints, the HGOALS performs better than the rest of the algorithms obtaining new best solutions for instances C9, C10 and C13 and reaching the best-known solutions for instances C6, C7 and C14.

Table 7. Comparison between the HGOALS and previous research for instances in group C (minimizing traveling cost).

Instance	*n*	*Q*	*BKS*	HGOALS	BRKGA	BBMO [37]	BLSA [38]	HBMO [44]
C1	50	160	412.96	**412.96**	**412.96**	**412.96**	**412.96**	**412.96**
C2	75	140	564.06	**564.06**	**564.06**	**564.06**	**564.06**	**564.06**
C3	100	200	639.26	**639.26**	**639.26**	**639.26**	**639.26**	640.08
C4	150	200	733.13	**733.13**	**733.13**	735.18	**733.13**	738.49
C5	199	200	869.00	875.49	871.21	872.15	**869.00**	878.25
C6	50	160	412.96	**412.96**	**412.96**	**412.96**	-	**412.96**
C7	75	140	567.59	**567.59**	**567.59**	568.95	-	568.49
C8	100	200	644.63	646.75	646.75	**644.63**	-	647.26
C9	150	200	743.51	**742.35**	743.51	757.24	-	761.28
C10	199	200	873.89	**873.69**	873.89	879.38	-	903.10
C11	120	200	676.40	678.70	680.23	678.54	678.54	680.15
C12	100	200	534.24	536.20	536.20	535.28	**534.24**	536.37
C13	120	200	862.74	**841.92**	862.74	897.10	-	922.28
C14	100	200	554.67	**554.67**	**554.67**	592.16	-	600.66

The comparison of the results with previous research for instances of group O is made in Table 8. In this set of instances, the BLSA performs the best obtaining the best-known solutions for six of the eight instances, followed by the HGOALS obtaining the best-known solution for one of the eight instances.

For the benchmark problems in group K, the comparison with results obtained in previous research is made in Table 9. The results show that the HGOALS outperforms the other two metaheuristics by obtaining five new best solutions for the eight instances in the group.

Table 8. Comparison between route decoder and previous research for instances in group O (minimizing traveling cost).

Instance	*n*	*Q*	*BKS*	HGOALS	%gap	BLSA [38]	BRKGA [43]
O1	200	900	5988.35	6003.08	0.25	**5988.35**	6003.08
O2	240	550	4549.46	4621.41	1.58	**4549.46**	4609.25
O3	280	900	7731.00	7734.70	0.05	**7731.00**	7735.16
O4	320	700	7251.30	7269.20	0.25	**7251.30**	7280.98
O5	360	900	9152.47	**9132.88**	−0.22	9152.47	9178.13
O6	400	900	9793.72	9833.41	0.41	**9793.72**	9847.00
O7	440	900	10,344.37	10,463.80	1.16	10347.70	10,444.70
O8	400	1000	12,412.26	12,457.10	0.36	**12,412.26**	12,449.60

Table 9. Comparison between the HGOALS results and previous research for instances in group K (minimizing traveling cost).

Instance *n*	*Q*	HGOALS	GRASP [42]	BRKGA [43]	
K01	240	550	**4622.58**	4790.31	4629.21
K02	320	700	7296.15	8407.07	**7285.98**
K03	400	900	**9831.07**	11,286.40	9837.78
K04	480	1000	**12,422.20**	14,325.87	12,431.50
K05	200	900	6004.74	6342.66	**6004.65**
K06	280	900	7735.16	8557.82	**7731.21**
K07	360	900	**9162.29**	10,624.48	9188.11
K08	440	900	**10,457.60**	12,129.07	10,488.50

4.5. Comparison of Average CPU Times with Previous Research

Finally, a comparison of the computational effort required by each of the solution methods is presented in Tables 10 and 11. Rows C, O and K present the mean for the different sets of test instances of the average CPU time per run reported by the different solutions methods. Row *BKS* shows the number of best-known solutions found and the number of tested instances by each algorithm. Finally, row *Model* presents the CPU model used by each algorithm and row *Factor*, the SPECint2000 (https://www.spec.org/cpu2000/results/) and SPECint2006 (http://www.spec.org/cpu2006/results/cint2006.html) base score, along with an estimate of the computing times reduction factor.

Table 10. Average CPU times comparison for the different solution approaches (hierarchical objective function).

Group	**HGOALS**	**BBMO [37]**	**BLSA [38]**	**HBMO [44]**	**ORTR [35]**
C	69.46	69.21	122.29	73.33	112.73
O	846.16	210.75	1057.75	217.95	709.95
K	889.04	–	–	–	–
BKS	20/30	5/22	12/15	1/22	7/22
Model	Core-2	T9550	T5550	T9550	Athlon 1Ghz
Factor	22.1(1)	18.1(0.82)	14(0.63)	18.1(0.82)	5.75(0.26)
Group	**HGOALS**	**MVNS [34]**	**HES [36]**	**TSB [32]**	**ALNS [45]**
C	69.46	485.71	581.50	559.41	1255.00
O	846.16	–	1051.50	–	–
K	889.04	–	–	–	–
BKS	20/30	6/14	6/22	2/14	4/14
Model	Core-2	Pentium M	Pentium 4	Pentium 3	Pentium 4
Factor	22.1(1)	9.04(0.41)	11.05(0.52)	5.11(0.23)	11.05(0.52)

Table 11. Average CPU times comparison for the different solution approaches (minimizing traveling cost).

Group	**HGOALS**	**BBMO [37]**	**BLSA [38]**	**HBMO [44]**	**BRKGA [43]**
C	50.23	59.77	110.29	61.72	326.70
O	860.10		1017.50	–	6163.79
K	804.89	–	–	–	6496.05
BKS	18/30	5/14	11/15	3/14	10/30
Model	Core-2		T5550	T9550	T9550
Factor	22.1(1)	18.1(0.82)	14(0.63)	18.1(0.82)	22.1(1)

In terms of CPU times, the HGOALS has slightly higher average CPU times, when compared with the BBMO, BLSA, HBMO and ORTR, although it obtains better results. It has lower average CPU times when compared with MVNS, HES, TSB, ALNS and BRKGA.

5. A Case Study

A security and alarm company requires its payment collection routes to be more efficient for two reasons: the first is to reduce the days required to collect payments, and the second is to reduce the fuel consumption used by the company representative. The company provides alarm and security services on a monthly or yearly basis. Customers choose the type of service they require and pay on the same basis. Therefore, at the beginning of each month, the company requires to collect payment from around 100 customers. Payment collection routes were decided by the company representative based on his experience and usually took between three to four weeks to be completed (15 to 20 working days). The company

provided the addressees of customer required to pay in the next two months. They also set a maximum number of customers to be visited per day (15 per day) and a maximum of working hours per day (seven hours). Finally, they provided the routes planned by the company representative in charge of collecting the payments for the next two months. A geographic information system (GIS), customers' addresses, and a web-based tool were used to build cost(distance) and time matrices to feed the algorithm. Such matrices were non-symmetric. The capacity parameter was set to $Q = 15$, the service time was set to 0.25 hours and the maximum route length to $L = 7.0$ hours. The algorithm was run five times using this information and the best solution was chosen. The results are presented in Table 12, where columns *Dist CRP*, *Days CRP*, *Dist HGOALS* and *Days HGOALS* show the total distance and required days for the routes planned by the company representative and the routes obtained by the HGOALS respectively. Columns *Dist % red* and *Days % red* show the percentaje reduction in total distance and working days obtained with the use of HGOALS proposed routes instead of the planned routes by the company representative. Finally, column *CPU* denotes the CPU time in seconds required by the algorithm and column *%gap* the gap with the best solution obtained by using a modified version (directed) of the formulation in [49] on CPLEX. This last column was added to verify the performance of the algorithm on real-life instances. For the exact method, a maximum CPU time was set to 10,800 s.

Table 12. Case study results.

Month	n	Dist CRP	Days CRP	Dist HGOALS	Days HGOALS	Dist %red	Days %red	CPU	$\%gap$
1	107	293.25	21	158.56	12	45.93	42.85	5.43	0.00
2	98	272.86	17	123.54	9	54.72	47.05	7.81	0.00

The results show a significant reduction in terms of total distance and working days. The routes obtained by the HGOALS are more efficient than those planned by the company representative. Moreover, fuel consumption is also importantly reduced due to the reduction in the total distance to be traveled. Note that the associated demand for each customer is the same ($d_i = 1$). Considering unitary demands helps the HGOALS reach the optimal solution with a small computational effort.

6. Concluding Remarks

A new hybrid grasshopper algorithm is presented and used to solve the open vehicle routing problem with capacity and maximum distance constraints. The OVRP is a variant of the classic VRP and has multiple applications, especially in transportation and distribution services. It is also a difficult combinatorial optimization problem of the NP-Hard type. The novel decoder first defines the number of vehicles to be used in the solution and then assigns clients to them. Such an approach allows the algorithm to find a balance between transmitting genetic information and allowing diversification of the population. The combination of the decoder with the local search and improvement phase grants robustness to the proposed heuristic. The last because the algorithm is capable of solving instances of up to 480 vertices without changing any parameter. The computational experiments considered two different objective functions. In the first, the prime objective is to minimize the number of vehicles to be used. In the second case, the total traveled distance is minimized. The results also show the excellent performance of the algorithm, especially in benchmark problems considering maximum distance constraints. For the first objective function, the algorithm improves or finds the best-known solutions for 21 of the 30 benchmark problems. For the second objective function, 18 of the 30 best-known solutions were found or improved by the algorithm. For the 30 benchmark problems considering distance constraints (including both objective functions), the algorithm found or improved the best-known solution for 21 of them. A comparison of the proposed algorithm with other heuristics in the literature shows its efficiency in terms of solutions quality,

particularly in benchmark problems with maximum distance constraints. Additionally, the algorithm was used to solve instances in a case study with excellent results. Finally, in future research, the algorithm can incorporate constraints related to time windows, heterogeneous fleet or hiring costs, among others.

Author Contributions: Conceptualization, E.R.-y.-R. and V.S.-M.; methodology, E.R.-y.-R. and I.G.-C.; validation, V.S.-M., I.G.-C. and J.P.-T.; formal analysis, E.R.-y.-R.,V.S.-M. and I.G.-C.; investigation, E.R.-y.-R. and V.S.-M.; resources, J.P.-T.; writing—original draft preparation, V.S.-M. and E.R.-y.-R.; writing—review and editing, I.G.-C. and J.P.-T.; visualization, J.P.-T.; supervision, E.R.-y.-R.. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: The authors thank the Tecnológico Nacional de México/Instituto Tecnológico de Saltillo and Universidad Autónoma de Coahuila for their support during the development of this research project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kennedy, J.; Eberhat, R. Particle swarm optimization. In Proceedings of the IEEE international conference on neural networks, Perth, WA, Australia, 27 November–1 December 1995; pp. 1942–1948.
2. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **1996**, *26*, 29–41.
3. Holland, J.H. *Adaptation in Natural and Artificial Systems*; University of Michigan Press: Ann Arbor, MI, USA, 1975.
4. Karaboga, D. An Idea Based on Honey Bee Swarm for Numerical Optimization, 2005. Available online: https://pdfs.semanticscholar.org/015d/f4d97ed1f541752842c49d12e429a785460b.pdf (accessed on 20 March 2020).
5. Glover, F.; Laguna, M. Tabu search. In *Handbook of Combinatorial Optimization*; Springer: New York, NY, 1998; pp. 2093–2229.
6. Mladenović, N.; Hansen, P. Variable neighborhood search. *Comput. Oper. Res.* **1997**, *24*, 1097–1100.
7. Pham, D.; Ghanbarzadeh, A.; Koc, E.; Otri, S.; Rahim, S.; Zaidi, M. *The Bees Algorithm Technical Note*; Manufacturing Engineering Centre, Cardiff University: Cardiff, UK, 2005; p. 40.
8. Filho, C.J.A.B.; de Lima Neto, F.B.; Lins, A.J.C.C.; Nascimento, A.I.S.; Lima, M.P. A novel search algorithm based on fish school behavior. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, SMC 2008, Singapore, 12–15 October 2008; pp. 2646–2651.
9. Neshat, M.; Sepidnam, G.; Sargolzaei, M. Swallow swarm optimization algorithm: A new method to optimization. *Neural Comput. Appl.* **2013**, *23*, 429–454.
10. Kaveh, A.; Bakhshpoori, T.; Afshari, E. An efficient hybrid Particle Swarm and Swallow Swarm Optimization algorithm. *Comput. Struct.* **2014**, *143*, 40–59.
11. Saremi, S.; Mirjailili, S.; Lewis, A. Grasshopper Optimisation Algorithm: Theory and application. *Adv. Eng. Softw.* **2017**, *105*, 30–47.
12. Yang, X.S. A New Metaheuristic Bat-Inspired Algorithm. *Nat. Inspired Coop. Strateg. Optim.* **2010**, *284*, 65–74.
13. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67.
14. Cubukcuoglu, C.; Ekici, B.; Tasgetiren, M.F.; Sariyildiz, S. OPTIMUS: Self-Adaptive Differential Evolution with Ensemble of Mutation Strategies for Grasshopper Algorithmic Modeling. *Algorithms* **2019**, *12*, 141.
15. Mafarja, M.; Aljarah, I.; Heidari, A.A.; Hammouri, A.I.; Faris, H.; Ala'M, A.Z.; Mirjalili, S. Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. *Knowl.-Based Syst.* **2018**, *145*, 25–45.
16. Wu, J.; Wang, H.; Li, N.; Yao, P.; Huang, Y.; Su, Z.; Yu, Y. Distributed trajectory optimization for multiple solar-powered UAVs target tracking in urban environment by Adaptive Grasshopper Optimization Algorithm. *Aerosp. Sci. Technol.* **2017**, *70*, 497–510.
17. Zhang, X.; Miao, Q.; Zhang, H.; Wang, L. A parameter-adaptive VMD method based on grasshopper optimization algorithm to analyze vibration signals from rotating machinery. *Mech. Syst. Signal Process.* **2018**, *108*, 58–72. [CrossRef]

18. Barman, M.; Choudhury, N.D.; Sutradhar, S. A regional hybrid GOA-SVM model based on similar day approach for short-term load forecasting in Assam, India. *Energy* **2018**, *145*, 710–720. [CrossRef]
19. Dantzig, G.B.; Ramser, J.H. The truck dispatching problem. *Manag. Sci.* **1959**, *6*, 80–91. [CrossRef]
20. Goel, A.; Gruhn, V. A general vehicle routing problem. *Eur. J. Oper. Res.* **2008**, *191*, 650–660. [CrossRef]
21. Sicilia, J.; Pellicer, E.L.; Agustin, B.R.; Finol, D.E. Siatema inteligente de planificación de rutas para la distribución de mercanciasías basado en las técnicas metaheurísticas, búsqueda de entorno variable y búsqueda tabú. *DYNA* **2013**, *88*, 414–423. [CrossRef]
22. Sicilia, J.A.; Royo-Agustin, B.; Quemada-Mayoral, C.; Olivares-Colay, M.J.; Larrodé, E. A decision support system to long haul freight transportation by means of ant colony optimization. *Transp. Syst. Technol.* **2015**. [CrossRef]
23. Cassettari, L.; Demartini, M.; Mosca, R.; Revetria, R.; Tonelli, F. A multi-stage algorithm for a capacitated vehicle routing problem with time constraints. *Algorithms* **2018**, *11*, 69. [CrossRef]
24. Sicilia, J.A.; Quemada, C.; Royo, B.; Escuín, D. An optimization algorithm for solving the rich vehicle routing problem based on Variable Neighborhood Search and Tabu Search metaheuristics. *J. Comput. Appl. Math.* **2016**, *291*, 468–477. [CrossRef]
25. Nepomuceno, N.; Barboza Saboia, R.; Rogério Pinheiro, P. A Fast Randomized Algorithm for the Heterogeneous Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Algorithms* **2019**, *12*, 158. [CrossRef]
26. Ait Haddadene, S.R.; Labadie, N.; Prodhon, C. Bicriteria Vehicle Routing Problem with Preferences and Timing Constraints in Home Health Care Services. *Algorithms* **2019**, *12*, 152. [CrossRef]
27. Zhao, M.; Lu, Y. A heuristic approach for a real-world electric vehicle routing problem. *Algorithms* **2019**, *12*, 45. [CrossRef]
28. Juárez Pérez, M.A.; Pérez Loaiza, R.E.; Flores, Q.; Malaquias, P.; Atriano Ponce, O.; Flores Peralta, C. A Heuristic Algorithm for the Routing and Scheduling Problem with Time Windows: A Case Study of the Automotive Industry in Mexico. *Algorithms* **2019**, *12*, 111. [CrossRef]
29. Liang, Y.C.; Minanda, V.; Gunawan, A.; Chen, A.H.L. Harmony search algorithm for time-dependent vehicle routing problem with time windows. *J. Appl. Sci.* 2013, **13**, 633–638.
30. Sariklis, D.; Powell, S. A heuristic method for the open vehicle routing problem. *J. Oper. Res. Soc.* **2000**, *51*, 564–573. [CrossRef]
31. Schrage, L. Formulation and structure of more complex/realistic routing and scheduling problems. *Networks* **1981**, *11*, 229–232. [CrossRef]
32. Brandão, J. A tabu search algorithm for the open vehicle routing problem. *Eur. J. Oper. Res.* **2004**, *157*, 552–564. [CrossRef]
33. Tarantilis, C.D.; Ioannou, G.; Kiranoudis, C.T.; Prastacos, G.P. Solving the open vehicle routeing problem via a single parameter metaheuristic algorithm. *J. Oper. Res. Soc.* **2005**, *56*, 588–596. [CrossRef]
34. Fleszar, K.; Osman, I.H.; Hindi, K.S. A variable neighbourhood search algorithm for the open vehicle routing problem. *Eur. J. Oper. Res.* **2009**, *195*, 803–809. [CrossRef]
35. Li, F.; Golden, B.; Wasil, E. The open vehicle routing problem: Algorithms, large-scale test problems, and computational results. *Comput. Oper. Res.* **2007**, *34*, 2918–2930. [CrossRef]
36. Repoussis, P.P.; Tarantilis, C.D.; Bräysy, O.; Ioannou, G. A hybrid evolution strategy for the open vehicle routing problem. *Comput. Oper. Res.* **2010**, *37*, 443–455. [CrossRef]
37. Marinakis, Y.; Marinaki, M. A Bumble Bees Mating Optimization algorithm for the Open Vehicle Routing Problem. *Swarm Evol. Comput.* **2014**, *15*, 80–94. [CrossRef]
38. Zachariadis, E.E.; Kiranoudis, C.T. An open vehicle routing problem metaheuristic for examining wide solution neighborhoods. *Comput. Oper. Res.* **2010**, *37*, 712–723. [CrossRef]
39. Salari, M.; Toth, P.; Tramontani, A. An ILP improvement procedure for the open vehicle routing problem. *Comput. Oper. Res.* **2010**, *37*, 2106–2120. [CrossRef]
40. Fu, Z.; Eglese, R.; Li, L.Y. A new tabu search heuristic for the open vehicle routing problem. *J. Oper. Res. Soc.* **2005**, *56*, 267–274. [CrossRef]
41. Miller, C.E.; Tucker, A.W.; Zemlin, R.A. Integer programming formulation of traveling salesman problems. *J. ACM (JACM)* **1960**, *7*, 326–329. [CrossRef]

42. Ruiz-y Ruiz, E.; Soto-Mendoza, V. *A GRASP Algorithm for the Open Vehicle Routing Problem*; Technical Report; Instituto Tecnológico de Saltillo: Saltillo, Mexico, 2017.
43. Ruiz, E.; Soto-Mendoza, V.; Barbosa, A.E.R.; Reyes, R. Solving the Open Vehicle Routing Problem with Capacity and Distance Constraints with a Biased Random Key Genetic Algorithm. *Comput. Ind. Eng.* **2019**, *133*, 207–219. [CrossRef]
44. Marinakis, Y.; Marinaki, M. A honey bees mating optimization algorithm for the open vehicle routing problem. In *2011 Genetic and Evolutionary Computation Conference*; Taylor and Francis: Abingdon, UK, 2011.
45. Pisinger, D.; Ropke, S. A general heuristic for vehicle routing problems. *Comput. Oper. Res.* **2007**, *34*, 2403–2435. [CrossRef]
46. Li, X.; Tian, P.; Leung, S.C. An ant colony optimization metaheuristic hybridized with tabu search for open vehicle routing problems. *J. Oper. Res. Soc.* **2009**, *60*, 1012–1025. [CrossRef]
47. Marinakis, Y.; Marinaki, M. A hybrid particle swarm optimization algorithm for the open vehicle routing problem. In *International Conference on Swarm Intelligence*; Springer: Berlin, Germany, 2012; pp. 180–187.
48. Ruiz-y Ruiz, E.; Ruiz-Barbosa, A. *A Cutting Plane Algorithm for the Open Vehicle Routing Problem with Capacity and Distance Constraints*; Technical Report; Instituto Tecnológico de Saltillo: Saltillo, Mexico, 2019.
49. Letchford, A.N.; Lysgaard, J.; Eglese, R.W. A branch-and-cut algorithm for the capacitated open vehicle routing problem. *J. Oper. Res. Soc.* **2007**, *58*, 1642–1651. [CrossRef]