CARPATHIAN J. MATH.
Volume **38** (2022), No. 1,
Pages 117 - 128

Online version at `https://www.carpathian.cunbm.utcluj.ro/`
Print Edition: ISSN 1584 - 2851; Online Edition: ISSN 1843 - 4401
DOI: https://doi.org/10.37193/CJM.2022.01.09

Dedicated to the memory of Academician Mitrofan M. Choban (1942-2021)

A two-level based genetic algorithm for solving the soft-clustered vehicle routing problem

OVIDIU COSMA[1], PETRICĂ C. POP[1] and CORINA POP SITAR[2]

ABSTRACT. The soft-clustered vehicle routing problem (Soft-CluVRP) is a relaxation of the clustered vehicle routing problem (CluVRP), which in turn is a variant of the generalized vehicle routing problem (GVRP). The aim of the Soft-CluVRP is to look for a minimum cost group of routes starting and ending at a given depot to a set of customers partitioned into a priori defined, mutually exclusive and exhaustive clusters, satisfying the capacity constraints of the vehicles and with the supplementary property that all the customers from the same cluster have to be supplied by the same vehicle. The considered optimization problem is NP-hard, that is why we proposed a two-level based genetic algorithm in order to solve it. The computational results reported on a set of existing benchmark instances from the literature, prove that our novel solution approach provides high-quality solutions within acceptable running times.

1. INTRODUCTION

The Vehicle Routing Problem (VRP) is one of the best-known, intensively studied and important combinatorial optimization problems, being investigated in the literature for a long period. In 1959, Dantzig and Ramser [6] studied a fuel distribution problem which led to the VRP. Since then, the VRP received a lot of attention, many results being obtained concerning the mathematical formulations of the problem and the solving methods (exact, heuristic, metaheuristic and hybrid algorithms). Several variants and extensions of the VRP have been considered in the literature. For further information on VRP and its variations, we refer to Mor and Speranza [19].

In the last two decades, a lot of attention was focused on VRP variants defined on graphs whose nodes (customers) are divided into a given number of subsets, called clusters. The first problem of this type was considered by Ghiani and Improta [14] and it was called the Generalized Vehicle Routing Problem (GVRP). GVRP looks for a minimum cost collection of routes beginning and finishing at the depot in such a way that every cluster is visited exactly once, the incoming and the outgoing customer at every visited cluster is the same, and the total demands for any route does not surpass the vehicle capacity. Different mathematical models of the GVRP based on integer programming and mixed integer programming have been introduced by Bektas et al. [3] and Pop et al. [23]. Due to the complexity of the problem, heuristic and metaheuristic algorithms have been proposed for dealing with the GVRP. See for more information in [20, 21, 24]. The clustered vehicle routing problem (CluVRP) is a version of the GVRP in which we look for minimum cost group of routes beginning and finishing at the depot, satisfying the capacity constraints, visiting all the customers, except the depot, precisely once, and with the

Received: 04.02.2021. In revised form: 19.10.2021. Accepted: 26.10.2021
2010 *Mathematics Subject Classification.* 90C27, 68T20.
Key words and phrases. *vehicle routing problem, soft-clustered vehicle routing problem, genetic algorithms, two-level solution approaches.*
Corresponding author: Petrică C. Pop; petrica.pop@cunbm.utcluj.ro

supplementary condition that once a vehicle reaches a cluster, all the customers belonging to that cluster are visited before the vehicle departs. As far as we know, the CluVRP was first considered by Pop et al. [23]. The current literature regarding the CluVRP is rather scarce: Pop et al. [23] presented three mathematical models of the problem based on mixed integer programming, Battara et al. [2] described exact algorithms for solving small and medium instances of the problem, Vidal et al. [32] proposed a hybrid metaheuristic algorithm, Defryn and Sörensen [7] developed a heuristic algorithm obtained by combining two variable neighborhood search algorithms and Exposito et al. [9] and Pop et al. [26] described two different approaches obtained by splitting the problem into two sub-problems.

The soft-clustered vehicle-routing problem (Soft-CluVRP) is a relaxation of the CluVRP obtained by relaxing the condition that once a vehicle reaches a cluster, all the vertices belonging to that cluster are visited before the vehicle departs, by the following soft-clustered constraint: all the customers belonging to the same cluster must be supplied by the same vehicle. This variant was first considered by Defryn and Sörensen [7] inspired by a real application in parcel delivery in which the customers are scattered in geographical regions called clusters, to simplify the sorting process. The same authors [7] developed a heuristic approach that solves the Soft-CluVRP problem by combining two variable neighborhood search algorithms. Recently, Hintsch and Irnich [16] provided an exact solution approach based on branch-and-cut and Heßler and Irnich [15] described a branch-and-cut algorithm which solved problems of small and medium sizes. For dealing with larger instances, Hintsch [17] developed a large multiple neighborhood search algorithm.

Recently, Posada et al. [29, 30] and Sabo et al. [31] investigated a new variant of the GVRP in which the nodes of the graph may belong to one or more clusters.

These variants of the VRP belong to the class of generalized network design problems. These problems naturally extend the classical combinatorial optimization problems, having the following main features: the nodes of the graph are divided into a predefined number of clusters and, when taking into consideration the feasibility constraints of the initial problem, these are expressed in relation to the clusters rather than to individual nodes. For more information on this class of problems, we refer to [4, 5, 8, 11, 13, 22, 28].

The aim of our paper is to present a novel solution approach for solving the Soft-CluVRP, namely a two-level based genetic algorithm, obtained by splitting the problem into two smaller subproblems: a macro-level subproblem and a micro-level subproblem. The macro-level subproblem looks for determining the clusters supplied by the same vehicle, using the corresponding macro graph (i.e. the graph achieved after substituting all the nodes of a cluster with a supernode defining that cluster), while the scope of the micro-level subproblem is to find the order in which customers are visited by each vehicle. The second subproblem reduces to finding a collection of minimum cost hamiltonian tours visiting all the customers corresponding to the clusters supplied by one vehicle. These hamiltonian tours are calculated optimally using the Concorde TSP solver [1]. The computational results obtained by testing our solution approach on the benchmark instances from the specific literature are reported and interpreted.

The remaining of our paper is organized as follows: in Section 2, we define the Soft-CluVRP, the two-level based genetic algorithm for solving the investigated problem is presented in Section 3, the computational experiments and the obtained results are displayed in Section 4 and finally, the conclusions are presented in Section 5.

2. Definition of the soft-clustered vehicle-routing problem

We consider $G = (V, E, c)$ an undirected, connected and weighted graph characterized by the set of nodes $V = \{v_0, v_1, v_2, ..., v_n\}$, the set of edges E, defined as follows:

$$E \subseteq \{\{v_i, v_j\} |\, v_i, v_j \in V,\ i \neq j \in \{0, 1, 2, ..., n\}\}.$$

and the cost function $c : V \to R_+$ which assigns to every edge $e = (u, v) \in E$ of the graph, a positive number $c(e) = c_e = c_{(u,v)} \in R_+$, called the cost of the edge e.

The edges are divided into two categories: edges defined between nodes that pertain to the same cluster, known as intra-cluster edges, and edges defined between nodes that pertain to different clusters, known as inter-cluster edges.

The nodes $v_1, ..., v_n$ represent the customers and the node v_0 represents the depot. The whole set of nodes $\{v_0, v_1, ..., v_n\}$ is divided into $k + 1$ mutually exclusive nonempty subsets, called clusters, denoted by $C_0, C_1, ..., C_k$, and satisfying the following conditions:

1. $V = C_0 \cup C_1 \cup ... \cup C_k$
2. $C_l \cap C_p = \emptyset$ for all $l, p \in \{0, 1, ..., k\}$ and $l \neq p$.
3. $C_0 = \{v_0\}$

The total cost of one route is the total amount incurred by all the edges pertaining to that specific route. Every customer v_i ($i \in \{1, ..., n\}$) has an established non-negative demand d_i to be shipped or picked up. There are m homogeneous vehicles; every one of them has a capacity Q and with the aim of ensuring feasibility, one assumes that $d_i \leq Q$ for each $i \in \{1, ..., n\}$. Moreover, the subsequent assumption is made: every vehicle carries out one route, one particular cluster is invariably visited by only one single vehicle and one vehicle can visit more than one cluster if the restrictions regarding its capacity are satisfied.

The *soft-clustered vehicle-routing problem* (Soft-CluVRP) aims in identifying the minimum cost group of routes visiting all the customers in such a way that the following constraints hold:

- every route begins and finishes at the depot;
- all the customers belonging to a certain cluster have to be supplied by exactly one vehicle;
- the sum of the demands of the visited nodes by a route does not surpass the vehicle capacity, Q.

An illustrative scheme of the Soft-CluVRP defined on an undirected graph with 25 nodes divided into 8 clusters, denoted by $C_0, C_1, ..., C_7$, where C_0 represents the depot, and a feasible solution of the problem are presented in Figure 1. We can observe that the feasible solution of the Soft-CluVRP defined on Figure 1 consists of two routes.

The routes satisfying the above conditions are called *soft-clustered routes*. Consequently, a feasible solution of the Soft-CluVRP is made up by a group of soft-clustered routes.

Evidently, the Soft-CluVRP is an NP-hard optimization problem because it contains the classical Capacitated Vehicle Routing Problem as a special case when all the clusters are singletons.

3. Description of the two-level based genetic algorithm

Our innovative solution approach for solving the Soft-CluVRP is achieved by splitting the problem into two smaller subproblems:

1. a macro-level subproblem which looks for determining the clusters supplied by each vehicle;

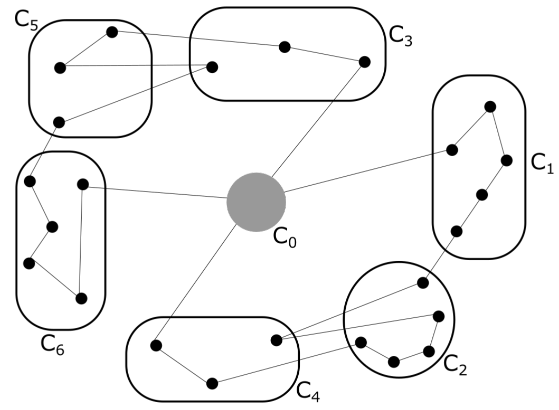

FIGURE 1. A feasible solution of the Soft-CluVRP residing in two soft-clustered routes

2. a micro-level subproblem whose aim is to find the visiting order of the nodes belonging to the clusters visited by the same vehicle.

3.1. **The macro-level subproblem.** The macro-level subproblem is defined on the graph G' achieved from G after we replaced all the nodes belonging to a given cluster C_i with a super-node corresponding to C_i, $\forall\ i \in \{1, ..., k\}$, the cluster C_0 representing the depot contains already one node. This graph G' will be the called *macro-graph*. Edges of G' link different pairs of super-nodes $C_0, C_1, \ldots, C_k$.

In order to supply a set of routes for visiting the clusters, called macro-routes, we use the macro-graph G'. A principal characteristic of this approach is that in this way we are reducing substantially the solution space of the original problem.

In Figure 2, we illustrate a set of two macro-routes corresponding to the example provided in Figure 1.

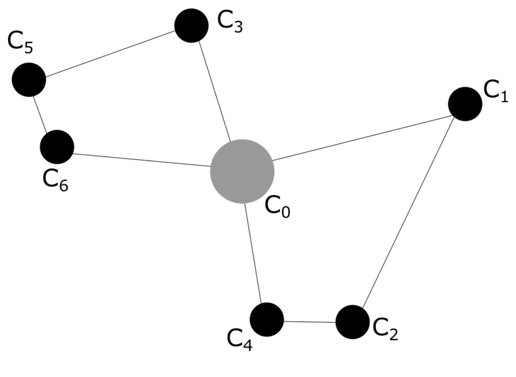

FIGURE 2. A feasible solution in the macro-graph corresponding to the example provided in Figure 1

There exist several soft-clustered routes associated to a macro-route visiting a number of clusters satisfying the capacity constraints of the vehicles. Among these soft-clustered routes there is one, which will be called the best soft-clustered route (w.r.t. cost minimization), that is going to be found using the Concorde TSP solver [1].

3.2. **The micro-level subproblem.** The macro-level subproblem applied on the macro-graph G' provides us for each vehicle, a collection of clusters to be visited, fulfilling the capacity constraints of the vehicles. Next we present how we can transform a collection of clusters visited by a vehicle into a TSP.

Let us suppose that we have a given collection of p clusters visited by a vehicle and satisfying the capacity constraints. We will denote by G_p the corresponding subgraph of G with the set of vertices belonging to the considered p clusters, including the depot and the corresponding edges, making no difference between intra-cluster and inter-cluster edges. Then finding the optimal way of visiting the p clusters reduces to solving a TSP on the subgraph G_p.

In Figure 3, we illustrate the optimal solution corresponding to the macro graph G', presented in Figure 2.

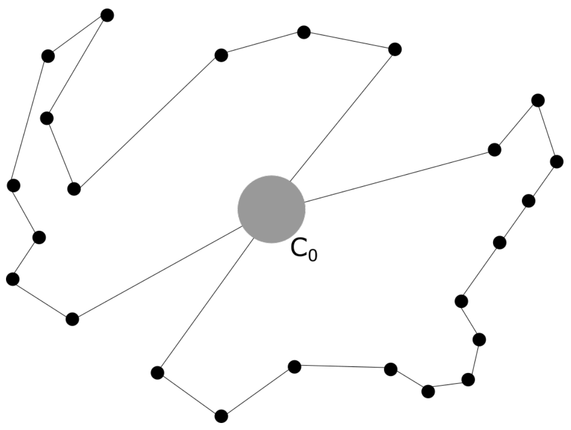

FIGURE 3. The optimal solution corresponding to the feasible solution presented in Figure 2

Evidently, the optimal solution corresponding to the feasible solution in the macro graph G' is a feasible solution of the Soft-CluVRP. In the illustrated example the feasible solution of the Soft-CluVRP consists of two soft-clustered routes which are the optimal Hamiltonian cycles visiting the following collection of clusters: $C_0 - C_1 - C_2 - C_4 - C_0$ and $C_0 - C_6 - C_5 - C_3 - C_0$

3.3. **The proposed Genetic algorithm.** In this subsection, we present the GA that supplies us good quality Soft-CluVRP solutions. The GA includes a macro-level solver for building valid chromosomes and a micro-level solver for evaluating their fitness.

3.3.1. *Representation.* In our developed algorithm we make use of an efficient representation at the level of the macro-graph, in which the chromosome for each candidate solution is represented as an array of fixed dimension, m and each gene contains the collection of the clusters visited by a certain vehicle. Our proposed representation restricts the route of each vehicle and contains the collection of clusters. In general, the chromosome has the following structure:

$$(g_1, g_2, ..., g_m)$$

where each gene g_i consists of a collection of clusters visited by vehicle i, $i \in \{1, ..., m\}$.

For example, for the feasible solution illustrated in Figure 2 made up by two macro-routes $C_0 - C_1 - C_2 - C_4 - C_0$ and $C_0 - C_6 - C_5 - C_3 - C_0$, the associated chromosome is represented by the following array of lists:

$$(\{C_0, C_1, C_2, C_4\}, \{C_0, C_3, C_5, C_6\}).$$

For each chromosome representing the collection of clusters visited by each vehicle, we can provide the corresponding best soft-clustered routes by solving a certain Traveling Salesman Problem, whose nodes are the depot and the nodes belonging to the collection of clusters visited by a given vehicle.

3.3.2. *Initial population.* It is known that the initial population has a substantial effect on the efficiency of the GA. In general, the initial population is generated randomly from the entire space of the candidate solutions, thus supplying an unbiased initial population. This variant of generating the initial population is very useful in bench-marking GAs, but in the case of real-world applications we might need additional information that helps to construct the initial population. In our case this additional information is supplied by the demands of the customers and the capacity Q of the vehicles.

Our initial population, which contains the sets of macro-routes is constructed as follows:

- The construction of each chromosome requires k steps, where k is the number of clusters.
- In the first m steps, a gene is created in which a cluster is added at random. In this way we assure that at the end there will be no empty genes.
- In the next steps, an unassigned cluster C_j, $j \in \{1, ..., k\}$ is randomly chosen, then a gene g_i, $i \in \{1, ..., m\}$ that has sufficient remaining capacity to include C_j is randomly chosen. If there is no such gene, then g_i is picked from all the genes. Then the cluster C_j is allocated to the gene g_i.

This generating mechanism could create supercharged genes, resulting invalid chromosomes. The supercharge of a gene g_i is: $s_{g_i} = \textit{total demand}_{g_i} - Q$ if $\textit{total demand}_{g_i} - Q > 0$, otherwise $s_{g_i} = 0$. A gene g_i is supercharged if $s_{g_i} > 0$. The supercharge of the chromosome is the sum of the supercharges of its genes. So a chromosome is invalid if it has a positive supercharge. If the chromosome is invalid, then the following repairing moves are repeated untill its supercharge becomes 0.

- Randomly chose two genes g_1 and g_2 and a cluster from each of them: C_{g_1} and C_{g_2}.
- Perform the first operation from the following list, that reduces the supercharge of the chromosome. If none of the operations has this property, then no changes are performed to the chromosome.
    1. Move C_{g_2} from g_2 to g_1.
    2. Move C_{g_1} from g_1 to g_2.
    3. Swap g_1 and g_2 between C_{g_1} and C_{g_2}.

3.3.3. *Fitness evaluation.* The fitness function associated to every individual chromosome from the solution space is provided by the total cost of the best soft-clustered routes corresponding to the set of nodes specified by its genes. Our scope is to minimize the total distance. Each gene g_i of the chromosome defines a subgraph G_{g_i} of G, which corresponds to a specific micro-level subproblem. The fitness evaluation implies solving the micro-level subproblem for each of the genes, and adding the results.

3.3.4. *Selection.* The selection mechanism merges the newly created population with the current population, removes the duplicates, then sorts the resulting population by fitness value. Then the best D chromosomes are selected for the new current population. All the other chromosomes are discarded.

3.3.5. *Crossover.* The crossover mechanism selects from the current population two parents P_1 and P_2, which are used to create an offspring. The first parent is always chosen randomly from the best 20% chromosomes in the current population, and the second parent is chosen randomly from the entire population. The crossover operator used in our GA works as follows:

- In offspring O are added genes from either P_1 or P_2 with equal probabilities. After this operation the chromosome may need repairs because the same clusters may appear in several genes and there may be clusters that do not appear in any gene.
- The first repairing step involves removing the duplicates. If a cluster C_j, $j \in \{1, ..., k\}$ appears in two genes g_1 and g_2, it will be deleted from either g_1 or g_2. The gene from which it is deleted is chosen at random, modified genes having priority. A modified gene is a gene that was already changed by the crossover operator. All the repairing steps try to preserve (as much as possible) the original genes inherited from the parrents.
- The second repairing step involves adding missing clusters to the chromosome. If a cluster C_j, $j \in \{1, ..., k\}$ is missing, it is added to a randomly selected gene, that is searched in order in the following cathegories.
  1. Empty genes;
  2. Modified genes with sufficient remaining capacity to include cluster C_j;
  3. Genes with sufficient remaining capacity to include cluster C_j;
  4. Modified genes;
  5. All the genes;
- The third step involves repairing the empty genes. If the g_1 gene has no assigned cluster, except the depot, then another gene g_2 that contains at least two clusters is chosen at random and one of its clustres it is moved to g_1.
- If the chromosome still needs repairs, then it is processed by the repairing moves described in the Initial population section.

3.3.6. *Mutation.* For some of the resulted offspring from the crossover algorithm, we may apply a mutation operator with a given probability which was determined based on preliminary computational experiments.

Our GA uses an exchanging inter-cluster mutation operator which acts as follows:

- Pick randomly two genes g_1 and g_2.
- Pick randomly one cluster C_1 from g_1, and one cluster C_2 from g_2.
- Move C_1 from g_1 to g_2, and C_2 from g_2 to g_1, only if such a move is possible, i.e. g_1 and g_2 do not become supercharged. If the move is not possible, the operator ends without changing the chromosome.

3.3.7. *Genetic parameters.* The parameters of the algorithm have an important impact on the efficiency of the GAs. Based on preliminary computational experiments, we have chosen the following values for the genetic parameters: the population size μ has been set at 50, the mutation probability was set at 5% and our GA stops when there has been no improvement in the population for 30 iterations.

4. COMPUTATIONAL RESULTS

This section contains the computational results achieved by our two-level based genetic algorithm for solving the Soft-CluVRP.

In order to asses the performance of the proposed two-level based genetic algorithm, we tested our solution approach on a set of 72 instances provided by Bektas et al. [3] and adapted from the CVRP benchmark instances. The names of the instances have the following format: **X–n**n**–k**z**–C**k**–V**m , where X is the type of the instance $X \in \{A, B, P\}$, n refers to the number of vertices , z represents the number of vehicles in the original CVRP instance , k is the number of clusters and m is the number of vehicles in the instance. There are two sets of instances denoted by GVRP-2 and GVRP-3, which differ by the desired average number of customers per cluster, that takes two possible values 2 and 3.

The proposed two-level based genetic algorithm for solving the Soft-CluVRP was implemented in Java, and in our experiments we performed 30 independent runs for each instance. The testing machine was an Intel(R) Core i3-8100 @ 3.6 GHz, 8 GB RAM. In order to solve the micro-level problem we used the Concorde TSP solver [1].

In Tables 1 and 2, we report the computational results achieved by our two-level based genetic algorithm, in comparison to the best existing heuristic method from the literature for solving the Soft-CluVRP, provided by Hintsch [17] using the developed large multiple neighborhood search algorithm, denoted by (LMNS). We point out that the computational results reported by Hintsch [17] were obtained using a standard PC equipped with MS Windows 7, an Intel(R) Core(TM) i7-5930K CPU processor clocked at 3.5 GHz, and with 64GB of main memory. The LMNS algorithm was implemented in C++ and compiled in 64-bit single-thread code with MS Visual Studio 2015 in release mode. For each instance, the LMNS algorithm was run with ten different random seeds, while in our GA we performed 30 runs for each instance.

Tables 1 and 2 have the following structure: the first column contains the name of the instance, followed by four columns that contain the results achieved by the LMNS algorithm [17]: the cost of the best found solution (Best), the average cost solution (Avg.), the average time calculated as the average over 30 runs, reported in seconds and the percentage gap (Gap) calculated as follows: $\%gap = 100 \times (Avg. - Best)/Best.$ *The "=" symbol indicates a value identical to the best known solution displayed in column 2.* The last four columns contain our obtained results: the cost of the best found solution (Best), the average cost solution (Avg.), the average time calculated as the average over 30 runs, reported in seconds and the percentage gap (Gap). To facilitate the comparison between the two algorithms, the best results are marked in bold. Regarding the efficiency of the algorithms, it is difficult to make a fair comparison, because they were implemented in different programming languages, and in our experiments we used a weaker processor. A comparison between the two programming languages in terms of efficiency can be found in [33]: the C++ time factor is 1, and the 64 bit Java time factor is 5.8. Therefore, our programming language is 5.8 times slower. Nevertheless, for 7 instances, we actually obtained better computational times.

Analyzing the computational results displayed in Table 1 in the case of GVRP-2 instances, we can observe that our two-level based genetic algorithm delivered the optimal solutions in all the 30 runs for 17 out of 32 instances, while the LMNS algorithm provided the optimal solutions in all the 30 runs for 29 out of 32 instances. In the case of our proposed solution approach the gap is at most 1.42% and the average computational time is up to 176 seconds.

When taking a closer look at the computational results displayed in Table 2, one can notice that: our two-level based genetic algorithm delivered the optimal solutions in all

TABLE 1. Computational results for solving the Soft-CluVRP in the case of GVRP-2 instances

Instance	**LMNS** [17]				**Our approach**			
	Best	Avg.	Time	Gap (%)	Best	Avg.	Time	Gap [%]
A-n32-k5-C16-V2	595	606.1	1.2	1.87	=	**603.44**	1.8	**1.42**
A-n33-k5-C17-V3	528	=	1.7	0	=	=	22.2	0
A-n33-k6-C17-V3	561	563.1	1.6	0.37	=	**=**	14.9	**0**
A-n34-k5-C17-V3	568	**=**	1.8	**0**	=	569.08	26.3	0.19
A-n36-k5-C18-V2	596	**=**	1.6	**0**	=	598.24	7.5	0.38
A-n37-k5-C19-V3	573	=	2.1	0	=	=	37.8	0
A-n37-k6-C19-V3	660	=	1.4	0	=	=	39.0	0
A-n38-k5-C19-V3	547	=	2.2	0	=	=	34.4	0
A-n39-k5-C20-V3	659	=	2.1	0	=	=	44.9	0
A-n39-k6-C20-V3	676	**=**	2.1	**0**	=	676.76	44.0	0.11
A-n44-k6-C22-V3	723	**=**	2.3	**0**	=	723.80	59.0	0.11
A-n45-k6-C23-V4	679	**=**	2.5	**0**	=	680.56	77.8	0.23
B-n31-k5-C16-V3	451	=	1.4	0	=	=	24.0	0
B-n34-k5-C17-V3	495	=	2.1	0	=	=	35.7	0
B-n35-k5-C18-V3	654	**=**	1.9	**0**	=	656.64	42.0	0.40
B-n38-k6-C19-V3	479	**=**	2.0	**0**	=	479.55	43.0	0.11
B-n39-k5-C20-V3	378	**=**	1.7	**0**	=	381.36	51.5	0.89
B-n41-k6-C21-V3	514	**=**	1.9	**0**	=	514.55	76.2	0.11
B-n43-k6=C22-V3	522	**=**	2.4	**0**	=	523.18	88.9	0.23
B-n44-k7-C22-V4	562	=	1.8	0	=	=	94.3	0
B-n45-k5-C23-V3	542	**=**	2.8	**0**	=	544.64	74.9	0.49
B-n45-k6-C23-V4	506	**=**	2.5	**0**	=	506.36	135.0	0.07
B-n50-k7-C25-V4	495	=	3.3	0	=	=	91.5	0
B-n50-k8-C25-V5	954	**=**	2.6	**0**	=	960.14	169.9	0.64
P-n16-k8-C8-V5	299	=	0.2	0	=	=	**0.0**	0
P-n19-k2-C10-V2	195	=	0.7	0	=	=	1.6	0
P-n20-k2-C10-V2	208	=	0.8	0	=	=	1.4	0
P-n21-k2-C11-V2	208	=	1.0	0	=	=	2.8	0
P-n22-k2-C11-V2	209	**=**	1.0	**0**	=	212.00	3.4	1.44
P-n22-k8-C11-V5	397	=	0.4	0	=	=	4.5	0
P-n23-k8-C12-V5	369	=	0.5	0	=	=	1.8	0
P-n40-k5-C20-V3	401	=	2.5	0	=	=	46.7	0
P-n45-k5-C23-V3	443	**=**	2.9	**0**	=	444.00	78.4	0.23
P-n50-k6C25-V4	464	**464.4**	3.4	**0.09**	=	465.08	151.3	0.23
P-n51-k10-C26-V6	548	**=**	2.3	**0**	=	548.75	176.0	0.14

the 30 runs for 26 out of 32 instances, while the LMNS algorithm provided the optimal solutions in all the 30 runs as well for 26 out of 32 instances. In the case of our proposed solution approach the gap is at most 1.04% and the average computational time is up to 276.5 seconds.

Overall, we can remark that our proposed solution approach delivers high-quality solutions of the Soft-CluVRP within reasonable running times.

TABLE 2. Computational results for solving the Soft-CluVRP in the case of GVRP-3 instances

Instance	LMNS [17]				Our approach			
	Best	Avg.	Time	Gap (%)	Best	Avg.	Time	Gap [%]
A-n32-k5-C11-V2	515	=	1.5	0	=	=	9.0	0
A-n33-k5-C11-V2	461	=	1.7	0	=	=	2.1	0
A-n33-k6-C11-V2	554	=	1.7	0	=	=	**0.1**	0
A-n34-k5-C11-V2	538	=	1.9	0	=	=	3.7	0
A-n37-k6-C13-V2	605	=	1.8	0	=	=	**0.0**	0
A-n38-k5-C13-V2	507	=	2.2	0	=	=	12.2	0
A-n39-k6-C13-V2	603	=	2.1	0	=	=	2.1	0
A-n44-k6-C15-V2	691	691.8	2.0	0.12	=	**=**	**0.0**	**0**
A-n45-k6-C15-V3	652	=	2.6	0	=	=	14.4	0
A-n53-k7-C18-V3	627	=	3.3	0	=	=	22.4	0
A-n55-k9-C19-V3	645	=	3.3	0	=	=	25.7	0
A-n61-k9-C21-V4	671	672.6	3.4	0.24	=	**672.47**	85.4	**0.22**
B-n31-k5-C11-V2	375	=	1.6	0	=	=	5.91	0
B-n34-k5-C12-V2	415	**=**	2.0	**0**	=	419.31	9.34	1.04
B-n35-k5-C12-V2	557	557.3	2.1	0.05	=	**=**	11.69	**0**
B-n38-k6-C13-V2	427	=	1.8	0	=	=	5.43	0
B-n41-k6-C14-V2	469	**=**	2.3	**0**	=	469.59	24.0	0.12
B-n43-k6-C15-V2	405	=	2.6	0	=	=	8.29	0
B-n44-k7-C15-V3	443	=	1.8	0	=	=	15.22	0
B-n50-k8-C17-V3	661	=	2.7	0	=	=	32.72	0
B-n52-k7-C18-V3	427	=	3.6	0	=	=	48.81	0
B-n56-k7-C19-V3	420	=	3.8	0	=	=	56.95	0
B-n63-k10-C21-V3	685	=	3.2	0	=	=	65.03	0
B-n66-k9-C22-V3	683	**685.5**	4.2	**0.37**	=	686.33	276.55	0.49
P-n16-k8-C6-V4	251	=	0.4	0	=	=	**0.0**	0
P-n22-k8-C8-V4	365	=	0.8	0	=	=	**0.7**	0
P-n23-k8-C8-V3	270	=	0.7	0	=	=	**0.0**	0
P-n50-k8-C17-V3	441	441.3	2.8	0.07	=	**441.20**	19.47	**0.05**
P-n50-k10-C17-V4	471	=	2.8	0	=	=	26.6	0
P-n51-k10-C17-V4	493	=	2.6	0	=	=	22.27	0
P-n55-k8-C19-V3	454	454.8	3.8	0.18	=	**=**	76.6	**0**
P-n55-k15-C19-V6	572	=	2.4	0	=	=	50.2	0
P-n60-k15-C20-V5	591	=	2.5	0	=	=	53.0	0
P-n65-k-10-C22-V4	575	=	4.7	0	=	=	84.3	0
P-n70-k10-C24-V4	602	**=**	5.1	**0**	=	602.23	133.2	0.04

5. CONCLUSION

In this paper, we presented a novel solution approach for solving the soft-clustered vehicle-routing problem, namely a two-level based genetic algorithm, obtained by decomposing the optimization problem into two smaller subproblems: a macro-level subproblem and a micro-level subproblem.

The computational results reported on a set of 72 benchmark instances from the literature prove that our novel solution approach based on genetic algorithms is comparable with the best existing methods from the literature and provides high-quality solutions within acceptable running times.

In the future, we plan to improve the developed two-level based genetic algorithm by combining with local search methods and to evaluate the generality and scalability of the proposed solution approach by testing it on larger instances.

References

[1] Applegate, D.; Bixby, R.; Chvatal, V.; Cook, W. *Concorde tsp solver*. http://www.tsp.gatech.edu/concorde/index.html, 2001.

[2] Battarra, M.; Erdogan, G.; Vigo, D. Exact algorithms for the clustered vehicle routing problem. *Oper. Res.* **62** (2014), no. 1, 58–71.

[3] Bektas, T.; Erdogan, G.; Ropke, S. Formulations and Branch-and-Cut Algorithms for the Generalized Vehicle Routing Problem. *Transportation Science* **45** (2011), no. 3, 299–316.

[4] Cosma, O.; Pop, P. C.; Zelina, I. A novel genetic algorithm for solving the clustered shortest-path tree problem. *Carpathian J. Math.* **36** (2020), no. 3, 401–414.

[5] Cosma, O.; Pop, P. C.; Zelina, I. An effective genetic algorithm for solving the clustered shortest-path tree problem. *IEEE Access* **9** (2021), 15570–15591.

[6] Dantzig, G. B.; Ramser, J. H. The Truck Dispatching Problem. *Management Science* **6** (1959), no. 1, 80–91.

[7] Defryn, C.; Sörensen, K. A fast two-level variable neighborhood search for the clustered vehicle routing problem. *Comput. Oper. Res.* **83** (2017), 78–94.

[8] Demange, M.; Monnot, J.; Pop, P. C.; Ries, B. On the complexity of the selective graph coloring problem in some special classes of graphs. *Theoretical Computer Science* **540-541** (2014), 82–102.

[9] Exposito-Izquierdo, C.; Rossi, A.; Sevaux, M. A two-level solution approach to solve the clustered vehicle routing problem. *Computers & Industrial Engineering* **91** (2016), 274–289.

[10] Feremans, C.; Labbe, M.; Laporte, G. Generalized network design problems. *Eur. J. Oper. Res.* **148** (2003), no. 1, 1–13.

[11] Fidanova, S.; Pop, P. C. An improved hybrid ant-local search for the partition graph coloring problem. *J. Comput. Appl. Math.* **293** (2016), 55–61.

[12] Fischetti, M.; Salazar-Gonzales, J. J.; Toth, P. A Branch-and-Cut Algorithm for the Symmetric Generalized Traveling Salesman Problem. *Operations Research* **45** (1997), no. 3, 378–394.

[13] Fuksz, L.; Pop, P. C. *A hybrid genetic algorithm with variable neighborhood search approach to the number partitioning problem.* in Proc. of HAIS 2013, Lecture Notes in Computer Science, **8073** (2013), 649–658.

[14] Ghiani, G.; Improta, G. An efficient transformation of the generalized vehicle routing problem. *Eur. J. Oper. Res.* **122** (2000), 11–17.

[15] Heßler, K.; Irnich, S. A branch-and-cut algorithm for the soft-clustered vehicle-routing problem. *Discret. Appl. Math.* **288** (2021), 218–234.

[16] Hintsch, T.; Irnich, S. Exact solution of the soft-clustered vehicle-routing problem. *Eur. J. Oper. Res.* **280** (2020), 164–178.

[17] Hintsch, T. Large multiple neighborhood search for the soft-clustered vehicle-routing problem. *Comput. Oper. Res.* in Press, (2020).

[18] Hintsch, T.; Irnich, S. Large multiple neighborhood search for the clustered vehicle-routing problem. *Eur. J. Oper. Res.* **270** (2018), no. 1, 118–131

[19] Mor. A.; Speranza, M. G. Vehicle routing problems over time: a survey. 4OR *Springer* **18** (2020), 129–149.

[20] Pintea, C.; Chira, C.; Dumitrescu, D.; Pop, P. C. Sensitive ants in solving the generalized vehicle routing problem. *Int. J. Comput. Commun. Control.* **6** (2011), no. 4, 734–741.

[21] Pop, P. C.; Pop Sitar, C.; Zelina, I.; Lupşe, V.; Chira, C. Heuristic algorithms for solving the generalized vehicle routing problem. *Int. J. Comput. Commun. Control.* **6** (2011), no. 4, 158–166, 2011.

[22] Pop, P. C. *Generalized network design problems: Modelling and optimization*, De Gruyter, Germany, 2012.

[23] Pop, P. C.; Kara, I.; Horvat Marc, A. New Mathematical Models of the Generalized Vehicle Routing Problem and Extensions. *Appl. Math. Model.* **36** (2012), no. 1, 97–107.

[24] Pop, P. C.; Matei, O.; Pop Sitar, C. An improved hybrid algorithm for solving the generalized vehicle routing problem. *Neurocomputing* **109** (2013), 76–83.

[25] Pop, P. C.; Matei, O.; Sabo, C. *A hybrid diploid genetic based algorithm for solving the generalized traveling salesman problem.* in Proc. of HAIS 2017, Lect. Notes Comput. Sci., **10334** (2017) 149–160.

[26] Pop, P. C.; Fuksz, L.; Horvat-Marc, A.; Sabo, C. A novel two-level optimization approach for clustered vehicle routing problem. *Computers & Industrial Engineering* **115** (2018), 304–318.

[27] Pop, P. C.; Matei, O.; Sabo, C.; Petrovan, A. A two-level solution approach for solving the generalized minimum spanning tree problem. *Eur. J. Oper. Res.* **265** (2018), no. 2, 478–487.

[28] Pop, P. C. The generalized minimum spanning tree problem: an overview of formulations, solution procedures and latest advances. *Eur. J. Oper. Res.* **283** (2020), no. 1, 1–15.
[29] Posada A.; Rivera J. C.; Palacio J. D. *A Mixed-Integer Linear Programming Model for a Selective Vehicle Routing Problem.* in Proc. of WEA 2018, Communications in Computer and Information Science, **916** (2018), 108–119.
[30] Posada A.; Rivera J. C.; Palacio J. D. *Selective Vehicle Routing Problem: A Hybrid Genetic Algorithm Approach.* in Proc. of EA 2019, Lecture Notes in Computer Science, **12052** (2020), 148–161.
[31] Sabo, C.; Pop, P. C.; Horvat-Marc, A. On the Selective Vehicle Routing Problem. *Mathematics* **8** (2020), no. 5, 771.
[32] Vidal, T.; Battarra, M.; Subramanian, A.; Erdogan, G. Hybrid metaheuristics for the clustered vehicle routing problem *Comput. Oper. Res.* **58** (2015), 87–99
[33] Hundt, R. *Loop recognition in c++/java/go/scala*, Proceedings of Scala Days https://days2011.scala-lang.org/sites/days2011/files/ws3-1-Hundt.pdf, last accessed 16.06.2021, (2011)

[1]DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
TECHNICAL UNIVERSITY OF CLUJ-NAPOCA
NORTH UNIVERSITY CENTER OF BAIA MARE
VICTORIEI 76, BAIA MARE, ROMANIA
Email address: `ovidiu.cosma@cunbm.utcluj.ro`
Email address: `petrica.pop@cunbm.utcluj.ro`

[2]DEPARTMENT OF ECONOMICS
TECHNICAL UNIVERSITY OF CLUJ-NAPOCA
NORTH UNIVERSITY CENTER OF BAIA MARE
VICTORIEI 76, BAIA MARE, ROMANIA
Email address: `corina.pop.sitar@cunbm.utcluj.ro`