Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Bilevel Memetic Search Approach to the Soft-Clustered Vehicle Routing Problem

Yangming Zhou, Yawen Kou, MengChu Zhou

To cite this article:
Yangming Zhou, Yawen Kou, MengChu Zhou (2022) Bilevel Memetic Search Approach to the Soft-Clustered Vehicle Routing Problem. Transportation Science

Published online in Articles in Advance 09 Dec 2022

. https://doi.org/10.1287/trsc.2022.1186 https://pubsonline.informs.org/journal/trsc

TRANSPORTATION SCIENCE
Articles in Advance, pp. 1–16
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

Bilevel Memetic Search Approach to the Soft-Clustered Vehicle Routing Problem

Yangming Zhou,[a,b,c] **Yawen Kou,**[d] **MengChu Zhou**[c,e,*]

[a] Data-Driven Management Decision Making Lab, Shanghai Jiao Tong University, Shanghai 200030, China; [b] Sino-US Global Logistics Institute, Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai 200030, China; [c] Macau Institute of Systems Engineering, Macau University of Science and Technology, Macau 999078, China; [d] Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China; [e] Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, New Jersey 07102

*Corresponding author

Contact: yangming.zhou@sjtu.edu.cn, https://orcid.org/0000-0002-4254-6517 (YZ); y30201066@mail.ecust.edu.cn, https://orcid.org/0000-0003-1670-4750 (YK); zhou@njit.edu, https://orcid.org/0000-0002-5408-8752 (MZ)

Received: April 1, 2022
Revised: June 22, 2022; September 5, 2022
Accepted: September 19, 2022
Published Online in Articles in Advance: December 9, 2022

https://doi.org/10.1287/trsc.2022.1186

Copyright: © 2022 INFORMS

Abstract. This work addresses a soft-clustered vehicle routing problem that extends the classical capacitated vehicle routing problem with one additional constraint, that is, customers are partitioned into clusters and all customers of the same cluster must be served by the same vehicle. Its potential applications include parcel delivery in courier companies and freight transportation. Due to its NP-hard nature, solving it is computationally challenging. This paper presents an efficient bilevel memetic search method to do so, which explores search space at both cluster and customer levels. It integrates three distinct modules: a group matching-based crossover (to generate promising offspring solutions), a bilevel hybrid neighborhood search (to perform local optimization), and a tabu-driven population reconstruction strategy (to help the search escape from local optima). Extensive experiments on three sets of 390 widely used public benchmark instances are conducted. The results convincingly demonstrate that the proposed method achieves much better overall performance than state-of-the-art algorithms in terms of both solution quality and computation time. In particular, it is able to find 20 new upper bounds for large-scale instances while matching the best-known upper bounds for all but four of the remaining instances. Ablation studies on three key algorithm modules are also performed to demonstrate the novelty and effectiveness of the proposed ideas and strategies.

Funding: This work was supported by the Macau Young Scholars Program [Grant AM2020011], Fundo para o Desenvolvimento das Cienciase da Tecnologia (FDCT) [Grant 0047/2021/A1], the National Natural Science Foundation of China [Grants 61903144, 71871142, and 71931007], and the Open Project of the Shenzhen Institute of Artificial Intelligence and Robotics for Society [Grant AC01202005002].
Supplemental Material: The online appendix is available at https://doi.org/10.1287/trsc.2022.1186.

Keywords: vehicle routing problem • bilevel optimization • heuristic • memetic search • variable neighborhood search

1. Introduction

The soft-clustered vehicle routing problem (SoftCluVRP) (Defryn and Sörensen 2017) extends the well-known capacitated vehicle routing problem (CVRP) (Lysgaard, Letchford, and Eglese 2004; Accorsi and Vigo 2021), where customers are partitioned into clusters, and all customers in a cluster must be served by a same vehicle. Note that SoftCluVRP allows the visits to customers of the same cluster to be interrupted by visits to customers of another different cluster. Such interruption is not allowed in the clustered vehicle routing problem (CluVRP) (Sevaux and Sörensen 2008), and all customers of a cluster must be served by the same vehicle in consecutive visits. Therefore, SoftCluVRP can be considered as a relaxation of CluVRP.

SoftCluVRP and CluVRP are the CVRP variants. They arise in many practical applications (Sevaux and Sörensen 2008, Battarra, Erdoğan, and Vigo 2014, Wang et al. 2020, Hintsch 2021, Li et al. 2022). The most representative example is parcel delivery in courier companies (Sevaux and Sörensen 2008), where customers are grouped into districts and parcels are sorted into containers according to their corresponding district by postal codes. Then, the containers filled with parcels are assigned to vehicles and drivers who deliver them to the recipients. Because the districting and sorting policy are typically made at a tactical planning level and altered only occasionally, they are often fixed before the true demand distribution is known. According to the accessibility of all parcels within a delivery vehicle, the resulting problem can be soft- or hard-clustered vehicle routing problems. If the delivery person has always all containers in the vehicle, deliveries to customers of the same district can be interrupted by deliveries to customers of another district, thus resulting in a problem corresponding to SoftCluVRP (Defryn and Sörensen 2017). Otherwise, it is CluVRP (Hoogeboom et al. 2016, Hintsch and Irnich 2018).

Another application can be found in freight transportation, where several containers are loaded to vehicles at a warehouse or harbor, and the goods inside the containers have to be delivered to different customers (Hintsch 2021). Similarly, application scenarios arise in transportation of passengers to recreation centers (Battarra, Erdoğan, and Vigo 2014), where customers are required to be picked up by the same vehicle.

Both CVRP and CluVRP are all NP-hard. They have attracted increasing attention from researchers and practitioners. Compared with CluVRP and CVRP, few efforts have been made to solve SoftCluVRP. Existing exact algorithms can only find optimal solutions on 81 out of 220 large-scale Golden instances given one hour. Heuristic algorithms (Defryn and Sörensen 2017; Hintsch 2021; Cosma, Pop, and Sitar 2022) are developed to find high-quality solutions for large-scale SoftCluVRP instances in a reasonable computation time. To the best of our knowledge, best-performing SoftCluVRP algorithms are local search-based algorithms. A genetic algorithm (Whitley 1994) is recently proposed to solve SoftCluVRP (Cosma, Pop, and Sitar 2022), but it cannot outperform local search-based algorithms. To enrich the set of solution methods for solving the computationally challenging SoftCluVRP, we propose a bilevel memetic search (BMS) algorithm. This work intends to make the following unique contributions:

- It proposes a bilevel memetic search (BMS) approach for SoftCluVRP, which decomposes the original problem into two subproblems: cluster level and customer level ones. BMS consists of five main modules. Besides the general population initialization and updating modules, it integrates three novel and effective modules: a group matching-based crossover to generate promising offspring solutions, a bilevel hybrid neighborhood search to find high-quality local optima, and a tabu-driven population reconstruction strategy to help the search jump out of the local optima, which explores the search space at both cluster and customer levels.
- It performs extensive experiments to evaluate the performance of BMS and compare it with state-of-the-art algorithms on 390 benchmark instances. Computational results show that BMS can find new upper bounds for 20 large-scale instances, and match previous best-known upper bounds on 366 instances. It also demonstrates its superiority over state-of-the-art algorithms in terms of both solution quality and computation time. In addition, ablation studies on three modules of the proposed method are investigated to confirm the importance of the proposed ideas and techniques.

The rest of this paper is organized as follows. Section 2 provides the problem description of SoftCluVRP. Section 3 discusses the related work of SoftCluVRP. Section 4 presents an efficient BMS approach for SoftCluVRP. Section 5 conducts extensive performance comparisons between BMS and state-of-the-art algorithms. Ablation studies are performed in Section 6, which is followed by conclusion and future work in Section 7.

2. Problem Description

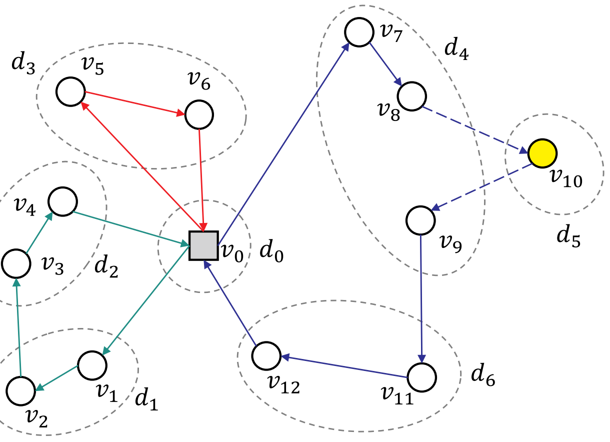
Given a complete undirected graph $G = (V, E)$ with the vertex set $V = \{0, 1, 2, \ldots, n\}$ and the edge set E. The vertex 0 denotes a *depot* and the remaining vertices $\{1, 2, \ldots, n\}$ denote n customers. There is a homogeneous fleet of m vehicles with capacity Q located at depot 0. For each edge $e(i, j) \in E$, $i, j \in V$ denotes a connection in the graph between vertices i and j, a non-negative routing cost w_{ij} is associated with it. A route $r = \{v_0, v_1, \ldots, v_s, v_{s+1}\}$ is a cycle in G passing the depot, that is, $v_0 = v_{s+1} = 0$ and $v_1, v_2, \ldots,$ and v_s are customers ($s \geq 1$). A route of CVRP is feasible if all customers $v_1, v_2, \ldots,$ and v_s are different and the capacity constraint $\sum_{j=1}^{s} d_{v_j} \leq Q$ is satisfied for given positive customer-specific demands d_i for $i \in V$.

Both SoftCluVRP and CluVRP are two clustered variants of CVRP. They partition the vertex set V into $N + 1$ clusters, that is, $V = C_0 \cup C_1 \cup C_2 \cup \cdots \cup C_N$, where $C_0 = \{0\}$ is a depot cluster and $C_1, C_2, \ldots,$ and C_N are N disjoint customer clusters, that is, $C_i \cap C_j = \emptyset$, $\forall i \neq j \in \{1, 2, \ldots, N\}$. Each customer cluster C_i is associated with a positive demand $d_i > 0$. The demand $d_i, i \in \{1, 2, \ldots, N\}$, can be considered as the sum of demands of all customers belonging to C_i. We assume that C_0 has zero demand, that is, $d_0 = 0$.

The objective of SoftCluVRP is to determine a cost-minimal set of m tours over G such that all customers of the same cluster must be served exactly once by the same vehicle and the total demand of all clusters served by one vehicle does not exceed the capacity Q of this vehicle. To better understand it, we illustrate it with an example shown in Figure 1.

Figure 1 presents a feasible solution of a given SoftCluVRP instance with one depot and 12 customers divided into seven clusters, denoted by $C_0 = \{v_0\}, C_1 = \{v_1, v_2\}, C_2 = \{v_3, v_4\}, C_3 = \{v_5, v_6\}, C_4 = \{v_7, v_8, v_9\}$, $C_5 = \{v_{10}\}$, and $C_6 = \{v_{11}, v_{12}\}$. This solution consists of three routes: $r_1 = \{v_0, v_1, v_2, v_3, v_4, v_0\}$, $r_2 = \{v_0, v_5, v_6, v_0\}$, and $r_3 = \{v_0, v_7, v_8, v_{10}, v_9, v_{11}, v_{12}, v_0\}$. Note that SoftCluVRP allows a vehicle to leave the current cluster and returns to it after serving the customer of other clusters. Specifically,

Figure 1. (Color online) A SoftCluVRP Solution



in r_3, vehicle 3 first visits v_7 and v_8 of C_4, and then leaves C_4 and serves v_{10} of C_5. Afterward, it returns to serve v_9 of C_4, followed by serving v_{11} and v_{12} of C_6. Finally, it returns to C_0. Note that the visits to v_9 of C_4 is interrupted in r_3, which is not allowed in the CluVRP that requires all customers of a cluster must be visited continuously before any customers in other clusters are visited.

3. Related Work

The SoftCluVRP generalizes CVRP (Miller 1995). In particular, it extends CVRP by adding one additional constraint, that is, all customers of a cluster must be served by the same vehicle (Defryn and Sörensen 2017). It is also a relaxed variant of CluVRP (Sevaux and Sörensen 2008) in which all customers of a cluster must be served by the same vehicle and in consecutive visits, and no interruption is allowed. Because SoftCluVRP was proposed by Defryn and Sörensen (2017), it has attracted much attention (Hintsch and Irnich 2020; Hintsch 2021; Cosma, Pop, and Sitar 2022). Several solution methods have been proposed and fall into the classes of exact and heuristic algorithms.

The former ones are theoretically able to guarantee the optimality of the solutions they find. Hintsch and Irnich (2020) designed and analyzed different branch-and-price (BP) algorithms for the exact solution of SoftCluVRP. They proved that BP equipped with an integer programming-based approach outperforms dynamic programming labeling-based algorithms by one order of magnitude in solution speed. As reported in Hintsch and Irnich (2020), their BP algorithms obtained excellent performance on 158 small- and medium-scale generalized vehicle routing problem (GVRP) instances. In particular, they were able to solve optimally 142 out of 158 instances with 16 to 262 vertices and six to 131 clusters in one hour. However, they can only find optimal solutions on 68 out of 220 large-scale Golden instances with 201 to 484 vertices and 14 to 97 clusters. Recently, Heßler and Irnich (2021) proposed a branch-and-cut (BC) algorithm for SoftCluVRP, which is complementary to BP. Compared with BP, BC is much simpler to implement. As indicated in Heßler and Irnich (2021), BC is particularly useful for SoftCluVRP instances that are truly clustered. It finds new provably optimal solutions to five GVRP and 13 Golden benchmark instances that could not solved before 2021. We observe that only 81 out of 220 large-scale Golden instances have been solved optimally by exact algorithms given one hour. To deal with these large-scale instances, heuristic algorithms constitute a valuable alternative that are used to solve SoftCluVRP approximately in reasonable computation time, for example, the allowable maximum number of iterations is 10,000 in Hintsch (2021). The results reported by Hintsch (2021) were obtained using a standard PC equipped with MS Windows 7, an Intel(R) Core(TM) i7-5930K CPU processor clocked at 3.5GHz, and with 64GB memory.

Because SoftCluVRP is a relatively new problem, only three heuristic algorithms have been reported in the literature (Defryn and Sörensen 2017; Hintsch 2021; Cosma, Pop, and Sitar 2022) to our knowledge. Defryn and Sörensen (2017) designed a fast two-level variable neighborhood search (TVNS) for both CluVRP and SoftCluVRP. Their low-level routing problem considers intra-route moves of customers and high-level routing problem considers inter-route moves of complete clusters. Both levels are solved by variable neighborhood search (VNS). Following the algorithm framework of VNS, Hintsch (2021) proposed a large multiple neighborhood search (LMNS) for SoftCluVRP. They combined multiple *destroy* and *repair* operators with two variable neighborhood descents for post-optimization. The former allows for relocating and swapping complete clusters among routes, whereas the latter improves single routes by classical and Balas-Simonetti neighborhoods search (Balas and Simonetti 2001). Recently, Cosma, Pop, and Sitar (2022) presented a two-level genetic algorithm (TGA) to solve SoftCluVRP, which decomposes the original problem into two smaller subproblems: macro-level and micro-level ones. As reported in Cosma, Pop, and Sitar (2022), both TGA and LMNS reached the same best results on 72 small- and medium-scale GVRP instances. In terms of average result, TGA found better results on seven out of 72 instances, the same results on 44 instances, and the worse results on the remaining 21 instances. Based on our literature review, no memetic algorithms have been explored for SoftCluVRP. This work represents the first try to develop such algorithm to solve SoftCluVRP.

4. Bilevel Memetic Search

4.1. Solution Representation and Evaluation

Given a SoftCluVRP instance with n customers divided into N customer clusters and m vehicles, we resort to a two-step method to describe its solution. We represent a SoftCluVRP solution as two sets of size m, that is, cluster and customer sets. The former is a set of clusters visited, while the latter indicates the order to visit customers. We use m-tour representation to represent a customer set. A feasible solution demonstrated in Figure 1 can be represented as $S = \{g, r\}$, where $g = \{g_1, g_2, g_3\}$ and $r = \{r_1, r_2, r_3\}$. For this solution representation, g_i indicates the set of clusters visited by the i-th vehicle, that is, $g_1 = \{0, 1, 2\}$, $g_2 = \{0, 3\}$, and $g_3 = \{0, 4, 5, 6\}$, whereas r_i denotes the order of customers visited by the i-th vehicle, that is, $r_1 = \{0, 1, 2, 3, 4, 0\}$, $r_2 = \{0, 5, 6, 0\}$, and $r_3 = \{0, 7, 8, 10, 9, 11, 12, 0\}$.

The target of SoftCluVRP is to determine a cost-minimal set of m routes that together visit all customers exactly once, and herewith also serve all clusters. Therefore, the objective function $f(S)$ can be defined as follows:

$$f(S) = \sum_{k=1}^{m} \left(\sum_{i=0}^{|r_k|-1} w_{r_k(i) r_k(i+1)} + w_{r_k(|r_k|) r_k(0)} \right), \qquad (1)$$

where $|r_k|$ is the number of customers in the k-th route. Note that $w_{r_k(i) r_k(i+1)}$ indicates the distance between two customers or the distance between the depot and the first customer in the k-th route, and $w_{r_k(|r_k|) r_k(0)}$ presents the distance between the last customer and the depot. Given a solution S, $f(S)$ calculates the total distance of all m routes in S.

4.2. General Framework

We present a bilevel memetic search (BMS) algorithm for solving SoftCluVRP. BMS is composed of five key modules: population initialization, group matching-based crossover (GMC), bilevel hybrid neighborhood search (BHNS), population updating, and tabu-driven population reconstruction (TPR). It is realized in Algorithm 1. Starting from an initial population P of η individuals generated by a population initialization procedure, BMS enters a "while" loop to iteratively perform evolutionary search. Following the general practice (Cordeau et al. 2006; Fu and Hao 2015; Zhou, Hao, and Duval 2017; Zhou, Hao, and Glover 2019), at each generation, two parent solutions are randomly selected from P to generate an offspring solution based on GMC operator. The offspring solution is then improved by the BHNS procedure, followed by a population updating strategy to decide whether to accept or discard it. Once a search stagnation is detected, it switches to use a TPR strategy to reconstruct the population. This process repeats until a stopping condition (e.g., a time limit or maximal generation count) is satisfied. We present each key algorithmic module of BMS next.

Algorithm 1 (Bilevel Memetic Search)

```
Input: Instance $I$, and population size $\eta$
output: The best solution found $S^*$
   /* Initialize*/
 1: $\tau \leftarrow 0$;/* idle update count $\tau$ at Page 7, line 15 */
 2: $Flag \leftarrow false$;/* stagnation state $Flag$ at Page 7, line 17 */
 3: $T[i] \leftarrow false$, $\forall i \in \{1, 2, \ldots, \eta\}$;/* tabu state $T[i]$ at
    Page 7, line 34 */
   /* Build an initial population */
 4: $P \leftarrow$ PopulationInitialization($\eta$);
 5: $S^* \leftarrow \arg\min\{f(S_i) \mid i = 1, 2, \ldots, \eta\}$;
 6: while Stopping condition is not met do
 7:    randomly select two parents $S_i$ and $S_j$ from $P$;
 8:    $S' \leftarrow$ GMC($S_i, S_j$);
 9:    $S' \leftarrow$ BHNS($S'$);
10:   if $f(S') < f(S^*)$ then
11:      $S^* \leftarrow S'$;
12:   end if
13:   $P \leftarrow$ PopulationUpdating($P, S', Flag, \tau, T$);
      /* Rebuild the population */
14:   if $Flag = true$ then
15:       $P \leftarrow$ TPR($P, S^*, T$);
16:   end if
17: end while
18: return The best solution found $S^*$
```

4.3. Population Initialization

BMS starts its search with an initial population P of η feasible solutions. For each feasible initial solution $S = \{g, r\}$, we construct it from the viewpoint of both the cluster level and the customer level based on the characteristics of SoftCluVRP. Specifically, we first generate a solution g at the cluster level. It means that all individual customers belonging to each cluster are disregarded and the cluster as a whole is assigned to an available vehicle. Once the solution at the cluster level g is obtained, we convert it to the corresponding solution at the customer level r.

The cluster level subproblem of SoftCluVRP aims to determine the assignment of clusters to vehicles, which can be regarded as a grouping problem (Ramos-Figueroa et al. 2020). To evaluate the quality of a grouping configuration, we define an inter-cluster distance matrix. Given two clusters C_i and C_j, their inter-cluster distance is approximated as the shortest edge cost, that is, $\min_{(i,j) \in C_i \times C_j} w_{ij}$. Therefore, the shorter Hamiltonian tours of g, the better the grouping configuration.

To generate a feasible solution at the cluster level g, we first randomly sort all customer clusters, and then iteratively add a customer cluster to a vehicle in two steps. At the first step, we consider the latest cluster that was added to all the vehicles, and try to identify a vehicle whose latest cluster is closest to the current one while satisfying its capacity constraint Q. At the second step, the current cluster is added into the identified vehicle. The process repeats until all customer clusters have been assigned to vehicles. It can be modeled as a one-dimensional bin packing problem. A set of items (clusters) with a given weight (total demand) are packed into a set of bins (vehicles) with a predefined maximal load (vehicle capacity Q). To increase randomness, we do not consider minimizing the total distance of g. If no vehicle has enough capacity left to store the next cluster, a redistribution operator (Defryn and Sörensen 2017) is used to re-optimize the current capacity distribution, which is able to increase the fill rate of one of the vehicles as much as possible. The redistribution operator executes a cluster-swap operator by swapping the vehicle of two already allocated clusters. All cluster pairs are examined sequentially and performed the best move operation, leading to the highest possible fill rate for one of the vehicles.

Once a solution at the cluster level is obtained, it can be converted to the solution at the customer level by a random conversion operator. Specifically, for each cluster, a route of the intra cluster traveling salesman problem (TSP) (Flood 1956) is constructed in a random way, that is, all customers of the current cluster are randomly visited by a vehicle. The solution obtained after applying the random conversion operator is treated as the initial solution at the customer level r. It consists of m routes, where each individual route can be considered as a single TSP. To obtain a high-quality initial solution at the customer level, we improve it by a well-known TSP solver named Lin-Kernighan-Helsgaun (LKH) (Helsgaun 2017).

4.4. Group Matching-Based Crossover

The crossover defines the way in which information is transmitted from parents to offspring. In this work, we design a dedicated backbone-based crossover named a group matching-based crossover (GMC), which is performed at the cluster level.

As mentioned before, the cluster level subproblem of SoftCluVRP is a grouping problem. The backbone-based crossover is popular and has been successfully applied to solve many grouping problems, such as bin packing and graph coloring (Porumbel, Hao, and Kuntz 2010). To identify the relationship between two solutions (i.e., g^F and g^M) at the cluster level, we sequentially match each group of g^F with each group of g^M. Then we can build a complete bipartite graph $G' = (V'_F, V'_M, E')$, where V'_F and V'_M represent m groups of g^F and g^M, respectively. Each edge $(g_i^F, g_j^M) \in E'$ is associated with a weight $|g_i^F \cap g_j^M|$, which counts the number of common clusters in group g_i^F of g^F and g_j^M of g^M.

Definition 1. Given a pair of matching groups between g_i^F of g^F and g_j^M of g^M, if there exists one cluster $C_k \in g_i^F, C_k \notin g_j^M$ or $C_k \in g_j^M, C_k \notin g_i^F$, then g_i^F and g_j^M are considered as partly matching groups; otherwise, g_i^F and g_j^M are treated as completely matching groups.

Given two parent solutions $S^F = \{g^F, r^F\}$ and $S^M = \{g^M, r^M\}$ randomly selected from population P, our GMC operator constructs an offspring solution g^O at the cluster level in three phases.

1. Matching groups of clusters. To identify the relationship between g^F and g^M, we first build a complete bipartite graph and then find a maximum weight matching with an augmenting path algorithm implemented in Setubal (1996). After the group matching is made between two parents, all groups can be divided into two categories: completely matching and partly matching groups, as presented in Definition 1. Note that for all completely matching groups, their clusters remain unchanged in subsequent search.

2. Inheriting common clusters. After group matching, a partial offspring solution g^O is obtained based on the matched results. For completely matching groups, we directly inherit them into the offspring solution. For partly matching groups, we first remove all non-common clusters and then inherit the remaining common clusters in each group.

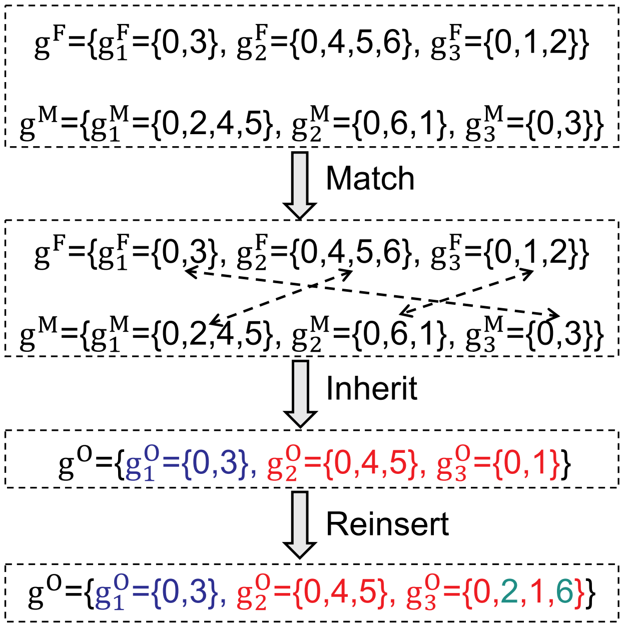
3. Reinserting removed clusters. Once a partial offspring solution g^O is obtained, we reinsert these removed clusters into the partly matching groups of g^O in a greedy way that leads to a minimal increase of the total distance at the cluster level. Due to the limitation of vehicle capacity Q, it is possible that no vehicle has enough capacity left to serve the customers in the next cluster. To deal with it, we use a redistribution operator (Defryn and Sörensen 2017) to repair it.

To demonstrate the basic idea of our GMC operator, we adopt an example shown in Figure 2. Two parents are $g^F = \{\{0,3\},\{0,4,5,6\},\{0,1,2\}\}$ and $g^M = \{\{0,2,4,5\},\{0,6,1\},\{0,3\}\}$. After maximum weight matching between g^F and g^M, we can identify the following group-to-group correspondence: $g_1^F \leftrightarrow g_3^M$, $g_2^F \leftrightarrow g_1^M$, and $g_3^F \leftrightarrow g_2^M$. According to Definition 1, we identify $g_1^F \leftrightarrow g_3^M$ is a completely matching group pair, whereas $g_2^F \leftrightarrow g_1^M$ and $g_3^F \leftrightarrow g_2^M$ are partly matching ones. Suppose that g^F is the donor parent. We can obtain an offspring solution g^O in three steps. Firstly, we can obtain a partial offspring solution by inheriting directly a completely matching group, that is, $g^O = \{\{0,3\}\}$. Secondly, we inherit two partly matching groups after removing two non-common clusters C_2 and C_6, that is, $g^O = \{\{0,3\},\{0,4,5\},\{0,1\}\}$. Finally, we obtain a complete solution by greedily inserting all removed clusters into the partly matching groups, which leads to a minimal increase of the total distance at the cluster level, that is, $g^O = \{\{0,3\},\{0,4,5\},\{0,2,1,6\}\}$.

4.5. Bilevel Hybrid Neighborhood Search (BHNS)

Local search is an important component in BMS to ensure the role of intensive exploitation of the search

Figure 2. (Color online) Diagram of Group Matching-Based Crossover


$g^F=\{g_1^F=\{0,3\}, g_2^F=\{0,4,5,6\}, g_3^F=\{0,1,2\}\}$
$g^M=\{g_1^M=\{0,2,4,5\}, g_2^M=\{0,6,1\}, g_3^M=\{0,3\}\}$
Match
$g^F=\{g_1^F=\{0,3\}, g_2^F=\{0,4,5,6\}, g_3^F=\{0,1,2\}\}$
$g^M=\{g_1^M=\{0,2,4,5\}, g_2^M=\{0,6,1\}, g_3^M=\{0,3\}\}$
Inherit
$g^O=\{g_1^O=\{0,3\}, g_2^O=\{0,4,5\}, g_3^O=\{0,1\}\}$
Reinsert
$g^O=\{g_1^O=\{0,3\}, g_2^O=\{0,4,5\}, g_3^O=\{0,2,1,6\}\}$

space by focusing on a limited search region. Due to the characteristic of SoftCluVRP, we propose a bilevel hybrid neighborhood search (BHNS) to perform search at both cluster level and customer level. It combines a random order-based variable neighborhood search (RO-VNS) and a TSP solver, that is, LKH (Helsgaun 2017). Given a starting solution $S=\{g,r\}$, it first employs an RO-VNS to improve g at the cluster level. Then, it is converted to a corresponding solution at the customer level. Finally, an LKH solver is applied to improve the solution at the customer level. BHNS is realized in Algorithm 2.

Algorithm 2 (Pseudo Code of Bilevel Hybrid Neighborhood Search)

```
Input: A candidate solution $S = \{g, r\}$
Output: The improved solution $S$
   /* Optimize $g$ by RO-VNS at the cluster level */
1: Sort the neighborhoods in $\mathcal{N}$ in a random order;
2: $k \leftarrow 1$;
3: while $k \le |\mathcal{N}|$ do
4:    Choose an improving neighbor $g'$ of $g$ in $\mathcal{N}_k$
      based on the first improvement strategy;
5:    if $f(g') < f(g)$ then
6:       $g \leftarrow g'$;
7:       $k \leftarrow 1$;
8:    else
9:       $k \leftarrow k+1$;
10:   end if
11: end while
   /* Convert $g$ at the cluster level to $r$ at the customer
   level */
12: $r \leftarrow \emptyset$;
13: for each route $g_i$ in $g$ do
14:    $r_i \leftarrow \emptyset$;
15:    for each cluster $C$ in $g_i$ do
16:       Sort each customer $v \in C$ in a random order;
17:       for each customer $v$ in $C$ do
18:          $r_i \leftarrow r_i \cup \{v\}$;
19:       end for
20:    end for
21:    $r \leftarrow r \cup r_i$;
22: end for
   /* Optimize $r$ by LKH at the customer level */
23: for each route $r_i$ in $r$ do
24:    $r_i \leftarrow \text{LKH}(r_i)$;
25: end for
26: return The improved solution $S$
```

BHNS uses an RO-VNS to optimize g at the cluster level. By ignoring all individual customers in the clusters, both the problem size and complexity of SoftCluVRP are dramatically reduced. RO-VNS integrates two kinds of neighborhoods: intra-route and inter-route ones. The former includes two classical operators, that is, two-opt and or-opt ones, which try to minimize the total distance of a single route. The latter also has two operators, relocate-sequence and swap-sequence, which operate on two different routes and try to reduce the total distance by relocating or swapping a set of consecutive clusters between two different routes.

1. Intra-route neighborhoods: The two-opt operator first removes two edges and then replaces them with two new edges to close the route; whereas the or-opt operator first removes $\nu \in \{1,2,3\}$ consecutive clusters and then inserts them into a new position in current route.

2. Inter-route neighborhoods: The relocate-sequence operator first removes $\nu \in \{1,2,3\}$ consecutive clusters from one route and then inserts them into another route; whereas the swap-sequence operator exchanges $\nu \in \{1,2,3\}$ consecutive clusters in one route with ν consecutive clusters in another route.

To increase the randomness of RO-VNS, we sort these four neighborhoods in a random order at the beginning. Starting from the first neighborhood, RO-VNS switches to the next neighborhood if no improvement can be found in the current neighborhood. Once an improvement is made, RO-VNS returns to the first neighborhood. To speed up the search, the first improvement neighbor selection strategy is applied in RO-VNS. The process repeats until all neighborhoods have been examined and the solution cannot be further improved, and then a local optimum is found at the cluster level.

Once the search is finished at the cluster level, a conversion procedure (Defryn and Sörensen 2017) is used to convert it to a solution r at the customer level. Solution r consists of m routes. Each individual route can be considered as a single TSP. To further optimize each individual route, we resort to a TSP heuristic known as LKH (Helsgaun 2017). Its source code is publicly available at http://akira.ruc.dk/keld/research/LKH/. Note that we keep all original parameters of LKH unchanged except for setting $\hat{\xi} = |V|/5$, where $\hat{\xi}$ denotes the maximum number of trials in each run.

4.6. Population Updating

For each improved offspring solution S, we resort to a quality-and-distance population updating strategy (Lü and Hao 2010; Fu and Hao 2015; Zhou, Hao, and Duval 2017; Zhou et al. 2021) to decide whether it should be accepted into population P. Our population updating strategy is beneficial to make a balance between solution quality and diversity. Because the subproblem at the cluster level of SoftCluVRP is a grouping problem, we use the well-known set-theoretic partition distance (Gusfield 2002) to evaluate the distance between two solutions. We first present the definition of the distance between two solutions, and then provide the distance between a solution and the population.

Definition 2. The distance between solutions $S_i = \{g^i, r^i\}$ and $S_j = \{g^j, r^j\}$ is $d_{ij} = N - \varrho(g^i, g^j)$, where $\varrho(g^i, g^j)$ is a similarity metric, which counts the maximum number of common customer clusters in all the matched groups of g^i and g^j.

To calculate $\varrho(g^i, g^j)$, we first use the well-known Hungarian algorithm (Kuhn 2005) to perform maximum weight matching between S_i and S_j. According to the Definition 2, two solutions are the same if all the corresponding routes at their cluster level are completely matching trips.

Definition 3. The distance between S_i and population $P = \{S_1, S_2, \ldots, S_\eta\}$ is $D(S_i, P) = \min\{d_{ij} \mid j = 1, 2, \ldots, \eta, j \neq i\}$.

To update the population with an improve offspring solution S, we first tentatively insert S into population P, that is, $P' \leftarrow P \cup \{S\}$. Then, all $\eta + 1$ individuals in P' are evaluated based on the following fitness score function:

$$\Psi(S_i, P') = \lambda * f'(S_i) + (1 - \lambda) * D'(S_i, P'), \tag{2}$$

where λ is a weight, and $f'(S_i)$ and $D'(S_i, P')$ are normalized values of $f(S_i)$ and $D(S_i, P')$, respectively, that is,

$$f'(S_i) = \frac{f(S_i) - \check{f}}{\hat{f} - \check{f}}, \tag{3}$$

$$D'(S_i, P') = \frac{\hat{D} - D(S_i, P')}{\hat{D} - \check{D}}, \tag{4}$$

where $\check{f} = \min\{f(S_i) \mid i = 1, 2, \ldots, \eta + 1\}$ and $\hat{f} = \max\{f(S_i) \mid i = 1, 2, \ldots, \eta + 1\}$ are the minimal and maximal solution costs in population P'. Correspondingly, $\check{D} = \min\{D(S_i, P') \mid i = 1, 2, \ldots, \eta + 1\}$ and $\hat{D} = \max\{D(S_i, P') \mid i = 1, 2, \ldots, \eta + 1\}$ are the minimal and maximal distances between a solution and P', respectively. According to (2), we can observe that the smaller the fitness score $\Psi(S_i, P')$, the better solution S_i for P'.

Given an improved offspring solution S returned by BHNS and a population $P = \{S_1, S_2, \ldots, S_\eta\}$, we use the following strategy to decide whether S should be added into the population or not, and detect whether the stagnation occurs or not. Firstly, S is temporarily inserted into P, leading to $P' = P \cup \{S\}$. Secondly, we evaluate each solution $S_i \in P'$ according to (2). Thirdly, we identify the worst solution with the maximal fitness score, that is, $S_w \leftarrow \arg_{S_i \in P'} \max \Psi(S_i, P')$. Finally, if the improved offspring solution S is not same to S_w, we replace S_w with S; otherwise, we discard it.

At the population updating phase, we record the idle update count τ. When it reaches an allowable maximal idle update count $\hat{\tau}$, the search is considered to be stagnated. Then, the stagnation state *Flag* is set to be true, which wakes up a population reconstruction procedure.

4.7. Tabu-Driven Population Reconstruction

Once the search is considered to be stagnated, a tabu-driven population reconstruction (TPR) strategy is launched to escape from a possible local optimum. TPR effectively combines the basic idea of tabu search (Glover 1989) and neighborhood decomposition strategy (Lai et al. 2021). The former attempts to forbid previously reconstructed solutions. The latter aims to accelerate the neighborhood search by only checking the promising neighborhood blocks. According to the characteristic of SoftCluVRP, TPR employs a neighborhood decomposition-based variable neighborhood search (ND-VNS) to improve a non-tabu solution at the customer level, and then replace it by its improved solution. More specifically, TPR first requires finding a non-tabu solution S_i (i.e., $T[i] = \mathit{false}$) in population P, where a binary vector T of size η is used to indicate the tabu state. Element $T[i] = \mathit{true}$ indicates that the corresponding solution S_i is a tabu solution; otherwise, a non-tabu solution. If such a solution S_i exists, then an additional search at the customer level is applied to improve it to a high-quality local optimum S, and it is marked as a tabu solution. Afterward, the best solution S^* is updated. If the improved solution S is different from any existing solutions in P, S_i is replaced by S; otherwise, S is discarded.

To improve a non-tabu solution at the customer level, we employ a ND-VNS, as realized in Algorithm 3. It combines two kinds of neighborhoods, Relocate (Relo for short) and Swap, in a VNS algorithm framework. Both operations treat all customers of a cluster a whole. Specifically, the former relocates all customers of one cluster and insert them in another route, whereas the latter tries to swap all customers of one cluster with all customers of one cluster in another route.

Due to the high complexity of examining Relo and Swap neighborhoods by LKH, we integrate a neighborhood decomposition strategy to speed up the search.

Inspired by Lai et al. (2021), our strategy works as follows. During the search, the current neighborhood (Relo or Swap) is first decomposed into $m \times (m-1)$ blocks, where m is the number of vehicles. Then, the search focuses only on the promising blocks, and skips those non-promising blocks that have been identified in previous iterations. To indicate whether a neighborhood block has been examined or not during the neighborhood search, we construct an $m \times m$ binary state matrix M, where $M_{ij} = 0$ if the block has been checked previously without finding any improving solution, and 1 otherwise.

Algorithm 3 (Pseudo Code of ND-VNS)

```
Input: A starting solution $S = \{g, r\}$ and proportion factor $\delta$
Output: The improved solution $S$
1: $Sign \leftarrow true$;
2: while $Sign = true$ do
3:     $Sign \leftarrow false$;
       /* Search in Relocate Neighborhood */
4:     for two different routes $g_1, g_2$ in $g$ do
5:         for each cluster $C_j^2$ in $g_2$ do
6:             $\hat{r}_1, \hat{r}_2 \leftarrow$ Relo$(r_1, C_j^2, r_2)$;
7:             if $f(\hat{r}_1) + f(\hat{r}_2) < \Phi_1(\delta) * (f(r_1) + f(r_2))$ then
8:                 $\hat{r}_1 \leftarrow$ LKH$(r_1)$, $\hat{r}_2 \leftarrow$ LKH$(r_2)$;
9:                 if $f(\hat{r}_1) + f(\hat{r}_2) < f(r_1) + f(r_2)$ then
10:                    $r_1 \leftarrow \hat{r}_1$, $r_2 \leftarrow \hat{r}_2$;
11:                    modify $g_1, g_2$;
12:                    $Sign \leftarrow true$;
13:                end if
14:            end if
15:        end for
16:    end for
       /* Search in Swap Neighborhood */
17:    if $Sign = false$ then
18:        for two different routes $g_1, g_2$ in $g$ do
19:            for each cluster $C_i^1$ in $g_1$, each cluster $C_j^2$ in $g_2$ do
20:                $\hat{r}_1, \hat{r}_2 \leftarrow$ Swap$(r_1, C_j^2, r_2, C_i^1)$;
21:                if $f(\hat{r}_1) + f(\hat{r}_2) < \Phi_2(\delta) * (f(r_1) + f(r_2))$ then
22:                    $\hat{r}_1 \leftarrow$ LKH$(r_1)$, $\hat{r}_2 \leftarrow$ LKH$(r_2)$;
23:                    if $f(\hat{r}_1) + f(\hat{r}_2) < f(r_1) + f(r_2)$ then
24:                        $r_1 \leftarrow \hat{r}_1$, $r_2 \leftarrow \hat{r}_2$;
25:                        modify $g_1, g_2$;
26:                        $Sign \leftarrow true$;
27:                    end if
28:                end if
29:            end for
30:        end for
31:    end if
32: end while
33: return The improved solution $S$
```

Starting from a solution $S = \{g, r\}$, ND-VNS explores Relo and Swap neighborhoods sequentially. Due to the higher cost of examining Swap neighborhood, ND-VNS first checks the Relo neighborhood. For any two different routes g_1 and g_2 in g and each cluster C_j^2 in g_2, we first remove all customers of C_j^2 from r_2. Then, all removed customers of C_j^2 are sorted randomly and reinserted into r_1 one by one with the best-insert procedure proposed in Hintsch (2021). The best-insert procedure aims to insert a customer into the current route by minimizing the insertion cost. Once the cluster C_j^2 has been inserted, two new routes $\hat{r}_1$ and $\hat{r}_2$ can be obtained. They are further examined by LKH if $f(\hat{r}_1) + f(\hat{r}_2) < \Phi_1(\delta) * (f(r_1) + f(r_2))$, where $\Phi_1(\delta)$ is a threshold, that is,

$$\Phi_1(\delta) = 1 + \delta * \frac{|C_j^2|}{|r_1|}, \tag{5}$$

where $|C_j^2|$ is the number of customers in cluster C_j^2, and $|r_1|$ is the number of customers in route r_1. These two original routes r_1 and r_2 are replaced by $\hat{r}_1$ and $\hat{r}_2$ if an improvement is found in the Relo neighborhood, that is, $f(\hat{r}_1) + f(\hat{r}_2) < f(r_1) + f(r_2)$.

Once no improved solution can be found in Relo neighborhood, ND-VNS switches to examine Swap neighborhood. Specifically, for any two different routes g_1 and g_2 in g and each cluster C_i^1 in g_1, we first remove all customers of C_i^1 from r_1, and then sort them randomly. All removed customers of C_i^1 are inserted into r_2 by the best-insert procedure. Similarly, for each cluster C_j^2 in g_2, we remove all customers of C_j^2 from r_2 and insert them in r_1. Then, we can have two new routes: $\hat{r}_1$ and $\hat{r}_2$, which are further improved by LKH if $f(\hat{r}_1) + f(\hat{r}_2) < \Phi_2(\delta) * (f(r_1) + f(r_2))$, where $\Phi_2(\delta)$ is a threshold, that is,

$$\Phi_2(\delta) = 1 + \delta * \left(k_1 \frac{|C_j^2|}{|r_1|} + k_2 \frac{|C_i^1|}{|r_2|}\right), \tag{6}$$

and $k_1 = f(r_1)/(f(r_1) + f(r_2))$ and $k_2 = f(r_2)/(f(r_1) + f(r_2))$ are two proportion factors. If an improved solution is found in the Swap neighborhood, that is, $f(\hat{r}_1) + f(\hat{r}_2) < f(r_1) + f(r_2)$, the original two routes r_1 and r_2 are updated by the two new routes $\hat{r}_1$ and $\hat{r}_2$, respectively. Otherwise ND-VNS ends the search, and returns the new improved solution S.

4.8. Computational Complexity of BMS

BMS starts its search from an initial population of η solutions generated by a population initialization procedure. For each initial solution, we first construct a solution at the cluster level by a randomized heuristic in $O(m \cdot N)$, and then converts it into a solution at the customer level in $O(n)$, followed by an LKH algorithm to improve it at the customer level. From Helsgaun (2000), the time complexity of LKH is approximately $O(n^{2.2})$. Hence, the population initialization procedure can be finished in $O(\eta \cdot m \cdot n^{2.2})$ time. In addition, we need to initialize the $\eta \times \eta$ distance matrix recording the distances between any two individuals in the population, which can be achieved in $\eta^2 \cdot m^3$.

At each generation of the main loop of Algorithm 1, BMS iteratively executes four search procedures: GMC, BHNS, population updating, and TPR. Specifically, an offspring solution is first generated by a GMC operator. Because the group matching operation can be achieved in $O(m^3)$, the time complexity of GMC is $O(m^3 + N \cdot |R|)$, where $|R|$ presents the number of removed clusters. Once an offspring solution is obtained, we improve it by BHNS. It is a two-level hybrid search by combining RO-VNS and LKH, whose time complexity is $O(N^2)$ and $O(m \cdot n^{2.2})$, respectively. The time complexity of BHNS is $O(N^2 + m \cdot n^{2.2})$. For an improved offspring solution, a population updating strategy is used to decide whether to accept or discard it, which can be achieved in $O(\eta \cdot m^3)$. Once search stagnation is detected, BMS switches to a TPR strategy for escaping from a possible local optimum. This procedure uses ND-VNS to improve a non-tabu solution at the customer level, and then tries to replace it by its improved solution, which can be done in $O(N^2 \cdot n^{2.2} + \eta \cdot m^3)$. To summarize, at each generation, the computational complexity of BMS is $O((m + N^2) \cdot n^{2.2} + \eta \cdot m^3)$.

5. Experimental Results

5.1. Benchmark Instances

Our experimental studies are performed on 390 widely used SoftCluVRP benchmark instances (Defryn and Sörensen 2017, Hintsch and Irnich 2020, Hintsch 2021). They were adapted from CVRP instances by defining θ as the desired average number of customers per cluster and building $N = \lceil (n+1)/\theta \rceil$ customer clusters. Note that the number of vehicles m is given for each instance, and it is not allowed to use fewer vehicles. These SoftCluVRP instances can be grouped into following three categories:

- GVRP instance set consists of 158 small- and medium-scale instances with $n \in [16, 262]$ and $N \in [6, \ 131]$ customer clusters. They are derived from the CVRP instances (Bektaş, Erdoğan, and Røpke 2011) called A, B, P, and GC by choosing $\theta \in \{2, 3\}$. They can be further divided into two groups: GVRP-2 and GVRP-3.
- Golden instance set is composed of 220 large-scale instances with $n \in [201, 484]$ and $N \in [14, 97]$. Battarra, Erdoğan, and Vigo (2014) generated them from the CVRP instances of Golden et al. (1998) by choosing $\theta = 4 + \{1, 2, \ldots, 11\}$ for each of the 20 original instances.
- Li instance set contains 12 large-scale instances with $n \in [561, 1201]$ and $\theta \in [113, 241]$. They were generated by Vidal et al. (2015) based on the CVRP instances of Li, Golden, and Wasil (2005) under $\theta = 5$.

5.2. Experimental Settings

All computational results are obtained using a standard PC with an AMD Ryzen 7 4800U CPU processor with 1.8GHz and 16GB RAM under the Windows 10 OS. Our algorithm is implemented in C++ and compiled in 32-bit single-thread code with MS Visual Studio 2019. Both our programs and results are available at https://github.com/YangmingZhou/Soft-clusteringVehicleRoutingProblem. Table 1 lists the detailed parameter settings of BMS. We empirically set the population size η to 20 and weight factor λ to 0.6, as suggested in Lü and Hao (2010) and Zhou, Hao, and Glover (2019). Our preliminary analysis indicates that the maximal idle update count $\hat{\tau}$ and proportional factor δ are two sensitive parameters, whereas η and λ are not. Following the general practice in heuristic algorithm design, we tune $\hat{\tau}$ and δ by experimentally determining them on some representative instances, as presented next.

5.3. Parameter Sensitivity Analyses

Following the general practice in heuristic algorithm design (Cordeau et al. 2006, Absi et al. 2015, Su et al. 2020), we tune $\hat{\tau}$ and δ based on some representative instances selected from benchmark instances. They are determined using a sequential process. At each step, the heuristic is executed by varying a single parameter within an interval. Sensitivity data are collected in terms of solution quality, including the best result $(\hat{f})$ and average result $(\bar{f})$. Furthermore, we define the performance gap in percentage between the solution value f and the best known value (BKV) as $\phi = (f - \text{BKV}) / \text{BKV} \times 100\%$. The smallest gap found in the 10 runs is denoted as $\check{\phi}$, whereas $\overline{\phi}$ presents the average gap over the 10 runs.

5.3.1. Test Instances. Our parameter sensitivity analyses are performed on 24 instances randomly selected from three instance sets. They include nine small and medium-scale GVRP instances (i.e., A-n45-k6-C15-V3, A-n63-k9-C32-V5, B-n66-k9-C22-V3, B-n68-k9-C34-V5, P-n50-k7-C25-V4, P-n101-k4-C34-V2, G-n262-k25-C88-V9, C-n151-k12-C76-V6, and C-n200-k16-C67-V6), 12 large-scale Golden instances (i.e., Golden-1-C22-N241, Golden-3-C27-N401, Golden-4-C69-N481, Golden-6-C32-N281, Golden-8-C63-N441, Golden-10-C65-N324, Golden-11-C37-N400, Golden-14-C54-N321, Golden-15-C31-N397, Golden-16-C35-N481, Golden-19-C73-N361, and Golden-20-C71-N421), and three large-scale Li instances (i.e., 640.vrp-C129-R5, 800.vrp-C161-R5, and 1040.vrp-C209-R5).

5.3.2. Parameter $\hat{\tau}$. This parameter denotes the maximal idle update count, which is used as a condition that wakes up the population reconstruction procedure. To study its influence on the performance of BMS, we test

Table 1. Parameter Settings of BMS

Parameter	Description	Value	Section
η	Population size	20	4.2
λ	Weight factor	0.6	4.6
$\hat{\tau}$	Maximal idle update count	10	4.6
δ	Proportion factor	0.5	4.7

different $\hat{\tau}$ values. Having set $\delta = 0.5$, we let $\hat{\tau}$ vary in the set $\{8, 9, 10, 11, 12\}$. Table 2 describes the detailed results of BMS with these values. At its bottom, we provide the average value of each performance indicator over 24 test instances. From Table 2, we can observe that BMS's performance in terms of $\check{\phi}$ and $\overline{\phi}$ is stable for $\hat{\tau} \in \{8, 9, 10, 11, 12\}$ and is slightly better for $\hat{\tau} = 10$. Hence, we set $\hat{\tau} = 10$.

5.3.3. Parameter δ.

This parameter is a proportional factor in Algorithm 3. After fixing $\hat{\tau} = 10$, we run BMS with five different proportional factors, that is, 0.3, 0.4, 0.5, 0.6, and 0.7. Detailed results of BMS with different δ values are summarized in Table 3. From it, we can find that its performance in terms of average $\overline{\phi}$ decreases as δ increases until $\delta = 0.5$. Then, for $\delta = 0.6$, we find an increase. In our tests, the minimum average $\overline{\phi}$ is obtained for $\delta = 0.5$, and this result validates our parameter choice.

5.4. Comparison with State-of-the-Art Algorithms

To evaluate BMS, we perform a detailed performance comparison between it and three state-of-the-art algorithms, that is, two-level variable neighborhood search (TVNS) (Defryn and Sörensen 2017), large multiple neighborhood search (LMNS) (Hintsch 2021), and two-level genetic algorithm (TGA) (Cosma, Pop, and Sitar 2022). Note that the source code of TVNS is available, whereas the source codes of LMNS and TGA are not available. Hence, we reimplement LMNS and TGA according to Hintsch (2021) and Cosma, Pop, and Sitar (2022), respectively. To guarantee a good implementation of both LMNS and TGA, we provide the detailed performance comparison between our implemented algorithms and their reported results in the GitHub Page mentioned earlier.

We run BMS and its three peers on our computational platform. Following Hintsch (2021), for each instance, we independently execute each algorithm with 20 different random seeds, and then record the best result ($\hat{f}$), average result ($\overline{f}$), and the average time ($\overline{t}$) in seconds needed to obtain the best result over 20 runs. Furthermore, we compute the performance gap in percentage. Comparative results of BMS and state-of-the-art algorithms are summarized in Tables 4–6. Detailed instance-by-instance results of BMS are provided in the online appendix.

In Table 4, columns 1 and 2 present the name of instance set (Set (#Inst.)) and the time limit ($\hat{t}$) in seconds, respectively. Columns 3–6 and 7–10 list the results of TVNS and LMNS, respectively, including the smallest gap ($\check{\phi}$), average gap ($\overline{\phi}$), average time ($\overline{t}$), and the number of best known values (#BKV) on each instance set. Columns 11–14 and 15–18 provide the corresponding results of TGA and BMS, respectively. At

Table 2. Sensitivity of BMS Performance to Parameter $\hat{\tau}$

		$\hat{\tau} = 8$		$\hat{\tau} = 9$		$\hat{\tau} = 10$		$\hat{\tau} = 11$		$\hat{\tau} = 12$	
Instances	$\hat{t}$	$\check{\phi}$	$\overline{\phi}$	$\check{\phi}$	$\overline{\phi}$	$\check{\phi}$	$\overline{\phi}$	$\check{\phi}$	$\overline{\phi}$	$\check{\phi}$	$\overline{\phi}$
A-n45-k6-C15-V3	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02
A-n63-k9-C32-V5	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
B-n66-k9-C22-V3	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
B-n68-k9-C34-V5	10	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01
P-n50-k7-C25-V4	10	0.00	0.02	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.03
P-n101-k4-C34-V2	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
G-n262-k25-C88-V9	30	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.03
C-n151-k12-C76-V6	30	0.00	0.16	0.00	0.22	0.00	0.08	0.00	0.15	0.00	0.24
C-n200-k16-C67-V6	30	0.00	0.03	0.00	0.05	0.00	0.03	0.00	0.05	0.00	0.10
Golden-1-C22-N241	180	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Golden-3-C27-N401	180	0.00	0.02	0.00	0.02	0.00	0.02	0.00	0.03	0.00	0.02
Golden-4-C69-N481	240	0.00	0.23	0.00	0.19	0.00	0.14	0.00	0.26	0.00	0.19
Golden-6-C32-N281	180	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Golden-8-C63-N441	240	0.00	0.15	0.00	0.10	0.00	0.10	0.00	0.07	0.00	0.17
Golden-10-C65-N324	240	0.00	0.63	0.00	0.57	0.00	0.52	0.00	0.48	0.00	0.60
Golden-11-C37-N400	180	0.00	0.05	0.00	0.09	0.00	0.07	0.00	0.05	0.00	0.07
Golden-14-C54-N321	240	0.00	0.12	0.00	0.12	0.00	0.14	0.00	0.15	0.00	0.12
Golden-15-C31-N397	180	0.00	0.07	0.00	0.02	0.00	0.05	0.00	0.07	0.00	0.10
Golden-16-C35-N481	180	0.00	0.07	0.00	0.04	0.00	0.01	0.00	0.03	0.00	0.02
Golden-19-C73-N361	240	0.00	0.01	0.00	0.04	0.00	0.01	0.00	0.00	0.00	0.00
Golden-20-C71-N421	240	0.00	0.02	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.02
640.vrp-C129-R5	600	0.00	0.43	0.00	0.21	0.00	0.29	0.00	0.38	0.00	0.35
800.vrp-C161-R5	600	0.00	0.36	0.00	0.31	0.00	0.33	0.00	0.41	0.00	0.38
1040.vrp-C209-R5	1,800	0.00	0.44	0.00	0.36	0.00	0.36	0.00	0.47	0.00	0.51
Total		**0.00**	0.12	**0.00**	0.10	**0.00**	**0.09**	**0.00**	0.11	**0.00**	0.12

Note. Larger numbers of BKVs and smaller $\check{\phi}$ and $\overline{\phi}$ values are highlighted in bold.

Table 3. Sensitivity of BMS Performance to Parameter δ

		$\delta = 0.3$		$\delta = 0.4$		$\delta = 0.5$		$\delta = 0.6$		$\delta = 0.7$	
Instances	$\hat{t}$	$\check{\phi}$	$\overline{\phi}$	$\check{\phi}$	$\overline{\phi}$	$\check{\phi}$	$\overline{\phi}$	$\check{\phi}$	$\overline{\phi}$	$\check{\phi}$	$\overline{\phi}$
A-n45-k6-C15-V3	10	0.00	0.03	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.01
A-n63-k9-C32-V5	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
B-n66-k9-C22-V3	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
B-n68-k9-C34-V5	10	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.01
P-n50-k7-C25-V4	10	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.02
P-n101-k4-C34-V2	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
G-n262-k25-C88-V9	30	0.00	0.08	0.00	0.02	0.00	0.01	0.00	0.03	0.00	0.01
C-n151-k12-C76-V6	30	0.00	0.20	0.00	0.24	0.00	0.08	0.00	0.17	0.00	0.19
C-n200-k16-C67-V6	30	0.00	0.12	0.00	0.09	0.00	0.03	0.00	0.05	0.00	0.05
Golden-1-C22-N241	180	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Golden-3-C27-N401	180	0.00	0.03	0.00	0.02	0.00	0.02	0.00	0.03	0.00	0.04
Golden-4-C69-N481	240	0.00	0.11	0.00	0.19	0.00	0.14	0.00	0.18	0.00	0.15
Golden-6-C32-N281	180	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Golden-8-C63-N441	240	0.00	0.11	0.00	0.14	0.00	0.10	0.00	0.11	0.00	0.14
Golden-10-C65-N324	240	0.60	0.72	0.00	0.60	0.00	0.52	0.00	0.48	0.00	0.48
Golden-11-C37-N400	180	0.00	0.16	0.00	0.05	0.00	0.07	0.00	0.12	0.00	0.07
Golden-14-C54-N321	240	0.00	0.15	0.00	0.12	0.00	0.14	0.00	0.20	0.00	0.20
Golden-15-C31-N397	180	0.00	0.05	0.00	0.02	0.00	0.05	0.00	0.02	0.00	0.07
Golden-16-C35-N481	180	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.05	0.00	0.08
Golden-19-C73-N361	240	0.00	0.03	0.00	0.01	0.00	0.01	0.00	0.03	0.00	0.07
Golden-20-C71-N421	240	0.00	0.02	0.00	0.00	0.00	0.01	0.00	0.06	0.00	0.02
640.vrp-C129-R5	600	0.00	0.22	0.00	0.34	0.00	0.29	0.00	0.32	0.00	0.35
800.vrp-C161-R5	600	0.00	0.30	0.00	0.43	0.00	0.33	0.00	0.36	0.00	0.51
1040.vrp-C209-R5	1,800	0.08	0.47	0.00	0.55	0.00	0.36	0.00	0.46	0.00	0.30
Total		0.03	0.12	**0.00**	0.12	**0.00**	**0.09**	**0.00**	0.11	**0.00**	0.12

Note. Larger numbers of BKVs and smaller $\check{\phi}$ and $\overline{\phi}$ values are highlighted in bold.

its bottom, we calculate the average values of $\check{\phi}$, $\overline{\phi}$, $\overline{t}$, and count the total number of BKVs.

From Table 4, we observe that our BMS has excellent performance on 158 small and medium-scale GVRP instances. In particular, BMS can find BKV for all 158 instances, resulting in best performance both in terms of $\check{\phi}$ and $\overline{\phi}$. Specially, the average value of $\overline{\phi}$ is only 0.01 for BMS, which indicates that BMS can steadily find BKV. Compared with both TVNS and TGA, BMS outperforms them in terms of both $\check{\phi}$ and $\overline{\phi}$ significantly. We observe that BMS is slightly better than LMNS in terms of $\check{\phi}$. For the performance indicator $\overline{\phi}$, it can obtain smaller value on six out of eight sets of instances than LMNS. These results show the superiority of BMS over the state-of-the-art algorithms on 158 GVRP instances.

Similarly, Table 5 presents the comparative results of BMS and three state-of-the-art algorithms on 220 Golden instances. From it, we observe that BMS also demonstrates excellent performance. Over all 220 instances, it finds new upper bounds for eight instances, and best known values for 208 instances. Compared with TVNS, LMNS, and TGA, BMS wins 216, 55, and 170 instances in terms of BKV, which is remarkably

Table 4. Detailed Results Between BMS and State-of-the-Art Algorithms on 158 GVRP Instances

		TVNS (Defryn and Sörensen 2017)				LMNS (Hintsch 2021)				TGA (Cosma, Pop, and Sitar 2022)				BMS			
Set (# instances)	$\hat{t}$	$\check{\phi}$	$\overline{\phi}$	$\overline{t}$	#BKV	$\check{\phi}$	$\overline{\phi}$	$\overline{t}$	#BKV	$\check{\phi}$	$\overline{\phi}$	$\overline{t}$	#BKV	$\check{\phi}$	$\overline{\phi}$	$\overline{t}$	#BKV
A-2 (27)	10	0.01	0.06	0.05	26	**0.00**	0.06	0.38	**27**	1.21	4.35	6.08	15	**0.00**	**0.00**	0.29	**27**
B-2 (23)	10	0.05	0.17	0.07	22	**0.00**	**0.00**	0.16	**23**	1.63	6.31	6.91	16	**0.00**	0.01	0.24	**23**
P-2 (24)	10	0.15	0.50	0.23	17	**0.00**	0.11	0.65	**24**	0.96	3.77	5.39	14	**0.00**	**0.00**	0.62	**24**
GC-2 (5)	30	5.81	8.83	4.01	2	0.01	**0.30**	8.93	4	33.90	44.62	27.50	0	**0.00**	0.40	11.29	**5**
A-3 (27)	10	**0.00**	0.16	0.09	**27**	**0.00**	0.06	0.14	**27**	0.02	1.67	3.44	26	**0.00**	**0.00**	0.31	**27**
B-3 (23)	10	**0.00**	<0.01	0.03	**23**	**0.00**	0.07	0.14	**23**	**0.00**	1.74	4.01	**23**	**0.00**	**0.00**	0.16	**23**
P-3 (24)	10	0.10	0.19	0.18	20	**0.00**	0.07	0.54	**24**	0.25	1.51	3.63	21	**0.00**	**0.00**	0.43	**24**
GC-3 (5)	30	0.46	1.05	3.18	1	**0.00**	0.31	6.62	**5**	20.16	26.18	25.72	0	**0.00**	**0.01**	6.33	**5**
Total		0.24	0.48	0.33	138	<0.01	0.08	0.80	157	2.34	5.24	6.27	115	**0.00**	**0.01**	0.88	**158**

Note. Larger numbers of BKVs and smaller $\check{\phi}$ and $\overline{\phi}$ values are highlighted in bold.

Table 5. Detailed Results Between BMS and State-of-the-Art Algorithms on 220 Golden Instances

		TVNS (Defryn and Sörensen 2017)				LMNS (Hintsch 2021)				TGA (Cosma, Pop, and Sitar 2022)				BMS			
Set (# instances)	$\hat{t}$	$\check{\phi}$	$\overline{\phi}$	$\bar{t}$	#BKV	$\check{\phi}$	$\overline{\phi}$	$\bar{t}$	#BKV	$\check{\phi}$	$\overline{\phi}$	$\bar{t}$	#BKV	$\check{\phi}$	$\overline{\phi}$	$\bar{t}$	#BKV
5 (20)	240	3.30	4.09	25.21	0	0.04	0.23	96.27	15	5.10	8.23	190.45	0	**0.00**	**0.12**	80.27	**18(2)**
6 (20)	240	2.83	3.77	23.34	0	0.05	0.25	95.57	15	3.44	6.04	182.54	0	**0.00**	**0.09**	64.90	**17(3)**
7 (20)	240	2.71	3.43	26.91	0	0.04	0.22	92.67	15	4.43	7.53	190.45	0	**0.00**	**0.06**	53.95	**19(1)**
8 (20)	240	2.40	3.23	29.65	0	0.08	0.22	81.42	15	3.07	5.66	182.54	3	**0.00**	**0.09**	54.59	**20**
9 (20)	180	2.21	2.98	29.65	0	0.04	0.20	67.28	17	3.29	6.22	142.01	4	**0.00**	**0.04**	40.88	**19 (1)**
10 (20)	180	2.15	2.86	25.14	0	0.04	0.21	68.42	15	2.13	4.38	132.51	4	**0.00**	**0.02**	33.23	**20**
11 (20)	180	2.13	2.73	28.32	0	0.05	0.23	61.25	13	1.59	3.90	129.65	5	**0.01**	**0.03**	29.58	**19**
12 (20)	180	1.90	2.52	37.46	0	0.06	0.19	63.75	12	1.10	2.87	124.49	5	**0.01**	**0.03**	25.54	**19**
13 (20)	180	1.88	2.58	40.37	0	0.03	0.20	63.26	14	1.07	2.85	124.89	6	**0.01**	**0.05**	42.48	**19**
14 (20)	180	1.80	2.44	39.09	0	0.02	0.15	59.92	15	0.68	2.10	118.06	9	**0.00**	**0.01**	40.51	**20**
15 (20)	180	1.86	2.48	36.45	0	0.03	0.14	65.75	15	0.45	1.62	107.80	10	**<0.01**	**0.03**	39.72	**18(1)**
Total		2.29	3.01	31.05	0	0.04	0.20	74.14	161	2.40	4.67	147.76	46	**<0.01**	**0.05**	45.97	**208(8)**

Notes. The number in parentheses () indicates the number of new upper bounds found by BMS. Larger numbers of BKVs and smaller $\check{\phi}$ and $\overline{\phi}$ values are highlighted in bold.

significant. In addition, we find that BMS achieves the smallest average values of both $\check{\phi}$ and $\overline{\phi}$, and it outperforms its peers on all 11 sets of instances in terms of both $\check{\phi}$ and $\overline{\phi}$.

To further evaluate the performance of BMS, we compare it with its peers on 12 large-scale Li instances. Their comparative results are presented in Table 6. It is observed that BMS demonstrates excellent performance. In particular, it can find new upper bounds for all 12 instances, which represents a great progress in this field. Compared with TVNS and TGA, BMS significantly outperforms them on all 12 instances in terms of both $\check{\phi}$ and $\overline{\phi}$. We observe that BMS achieves better $\check{\phi}$ values than LMNS on all 12 instances. For the performance indicator $\overline{\phi}$, BMS also outperforms LMNS by finding 10 better results and one same result in terms of $\overline{\phi}$. More importantly, BMS achieves these results in shorter computation time than LMNS. These results show that BMS performs far better than the state-of-the-art algorithms.

Furthermore, we give a summary of performance comparisons between BMS and its peers on GVRP, Golden, and Li instances in Table 7. Over all 390 instances, we observe that TVNS and TGA can find the BKV on 138 and 161 instances, respectively. LMNS demonstrates better performance by reaching the BKV on 318 out of 390 instances. BMS achieves better performance by finding better or the same performance on 386 out of 390 instances. In particular, it finds new upper bounds on 20 instances, and matches BKV on 366 instances, which is never seen in the existing work.

6. Ablation Studies

In this section, we perform additional experiments to gain a deeper understanding of BMS. In particular, we conduct three groups of experiments: (1) to evaluate the

Table 6. Detailed Results Between BMS and State-of-the-Art Algorithms on 12 Li Instances

		TVNS (Defryn and Sörensen 2017)				LMNS (Hintsch 2021)				TGA (Cosma, Pop, and Sitar 2022)				BMS			
Instances	$\hat{t}$	$\check{\phi}$	$\overline{\phi}$	$\bar{t}$	#BKV	$\check{\phi}$	$\overline{\phi}$	$\bar{t}$	#BKV	$\check{\phi}$	$\overline{\phi}$	$\bar{t}$	#BKV	$\check{\phi}$	$\overline{\phi}$	$\bar{t}$	#BKV
560.vrp-C113-R5	600	0.90	1.36	29.96	0	0.21	0.60	414.59	0	40.46	45.49	513.38	0	**0.00**	**0.09**	331.34	**(1)**
600.vrp-C121-R5	600	0.64	0.77	22.16	0	0.43	0.63	380.33	0	31.07	36.84	511.15	0	**0.00**	**0.08**	440.97	**(1)**
640.vrp-C129-R5	600	3.77	5.25	52.45	0	0.09	**0.29**	383.97	0	43.88	48.52	563.28	0	**0.00**	**0.29**	282.23	**(1)**
720.vrp-C145-R5	600	3.97	5.18	55.67	0	0.02	0.37	416.99	0	54.23	60.90	554.47	0	**0.00**	**0.30**	312.80	**(1)**
760.vrp-C153-R5	600	0.51	0.77	26.87	0	0.24	0.54	356.35	0	42.55	45.16	446.65	0	**0.00**	**0.13**	445.58	**(1)**
800.vrp-C161-R5	600	4.43	5.12	116.95	0	0.06	0.35	395.08	0	63.97	68.74	551.70	0	**0.00**	**0.33**	300.78	**(1)**
840.vrp-C169-R5	1,800	0.55	0.64	38.00	0	0.29	0.47	1,082.01	0	43.83	45.55	1,352.60	0	**0.00**	**0.16**	1,073.65	**(1)**
880.vrp-C177-R5	1,800	5.36	6.58	95.61	0	0.03	**0.38**	1,066.60	0	60.21	65.93	1,701.27	0	**0.00**	0.69	493.01	**(1)**
960.vrp-C193-R5	1,800	5.21	6.59	134.86	0	0.08	0.78	1,304.11	0	63.63	70.37	1,685.77	0	**0.00**	**0.57**	509.24	**(1)**
1040.vrp-C209-R5	1,800	4.80	6.68	226.05	0	0.07	0.66	1,192.13	0	69.55	77.21	1,685.97	0	**0.00**	**0.36**	622.17	**(1)**
1120.vrp-C225-R5	1,800	4.80	6.72	257.06	0	0.05	0.83	1,161.86	0	79.24	85.95	1,679.89	0	**0.00**	**0.46**	524.45	**(1)**
1200.vrp-C241-R5	1,800	5.41	7.16	305.28	0	0.10	1.24	1,228.19	0	87.91	92.31	1,704.74	0	**0.00**	**0.94**	459.71	**(1)**
Total		3.36	4.40	113.41	0	0.14	0.59	781.85	0	56.71	61.91	1,079.24	0	**0.00**	**0.37**	482.99	**(12)**

Notes. The number in parentheses () indicates the number of new upper bounds found by BMS. Larger numbers of BKVs and smaller $\check{\phi}$ and $\overline{\phi}$ values are highlighted in bold.

Table 7. Summary of BMS and State-of-the-Art Algorithms on 390 Benchmark Instances

Instances		TVNS (Defryn and Sörensen 2017)	LMNS (Hintsch 2021)	TGA (Cosma, Pop, and Sitar 2022)	BMS
GVRP	158	138	157	115	158
Golden	220	0	161	46	208(8)
Li	12	0	0	0	(12)
Total	390	138	318	161	366 (20)

benefit of GMC, (2) to investigate the superiority of BHNS, and (3) to verify the effectiveness of TPR. Following a general practice (Absi et al. 2015, Su et al. 2020), our following experiments are conducted on the 24 aforementioned representative instances.

6.1. Benefit of Group Matching-Based Crossover

To demonstrate the benefit of the group matching-based crossover (GMC) operator, we experimentally compare BMS with its an alternative version BMS$_A$, where the GMC operator is replaced by the add-repair crossover originally used in two-level genetic algorithm (TGA) (Cosma, Pop, and Sitar 2022). To the best of our knowledge, TGA is the only existing population-based algorithm for SoftCluVRP.

Comparative results of BMS and BMS$_A$ on 24 representative instances in terms of $\hat{f}$ and $\overline{f}$ are presented in Figure 3. In Figure 3, the x-axis presents instances, and y-axis denotes the performance gaps. By treating BMS$_A$ as a baseline algorithm, we calculate their performance gap as $(f - f')/f' \times 100\%$, where f is its result and f' is the result of BMS$_A$. A performance gap smaller than zero indicates that BMS obtains a better result on the corresponding instance.

From Figure 3, we can observe that BMS performs a better performance than BMS$_A$ in terms of $\hat{f}$. In particular, it finds better results (i.e., the values below 0 in the left part of Figure 3) on four instances, and same results on the remaining 20 instances. BMS demonstrates better performance than BMS$_A$ in terms of $\overline{f}$, including improved results on 13 instances, and the same results on the remaining 11 instances. Moreover, we observe that there is no significant difference between BMS and BMS$_A$ on nine small and medium instances. For the remaining 15 large instances, BMS shows better performance than BMS$_A$. These results confirm the superiority of GMC over the add-repair crossover of TGA (Cosma, Pop, and Sitar 2022).

6.2. Superiority of Bilevel Hybrid Neighborhood Search

Given a candidate solution $S = \{g, r\}$, BMS employs BHNS to improve it to a local optimum. To show its superiority, we experimentally compare BMS with its a variant BMS$_B$. BMS$_B$ is obtained from BMS by replacing BHNS with two-level variable neighborhood search (TVNS) (Defryn and Sörensen 2017). Figure 4 describes the comparative performance of BMS and BMS$_B$.

Figure 3. (Color online) Comparison Between BMS (with Group-Matching-Based Crossover) and BMS$_A$ (with Add-Repair Crossover)

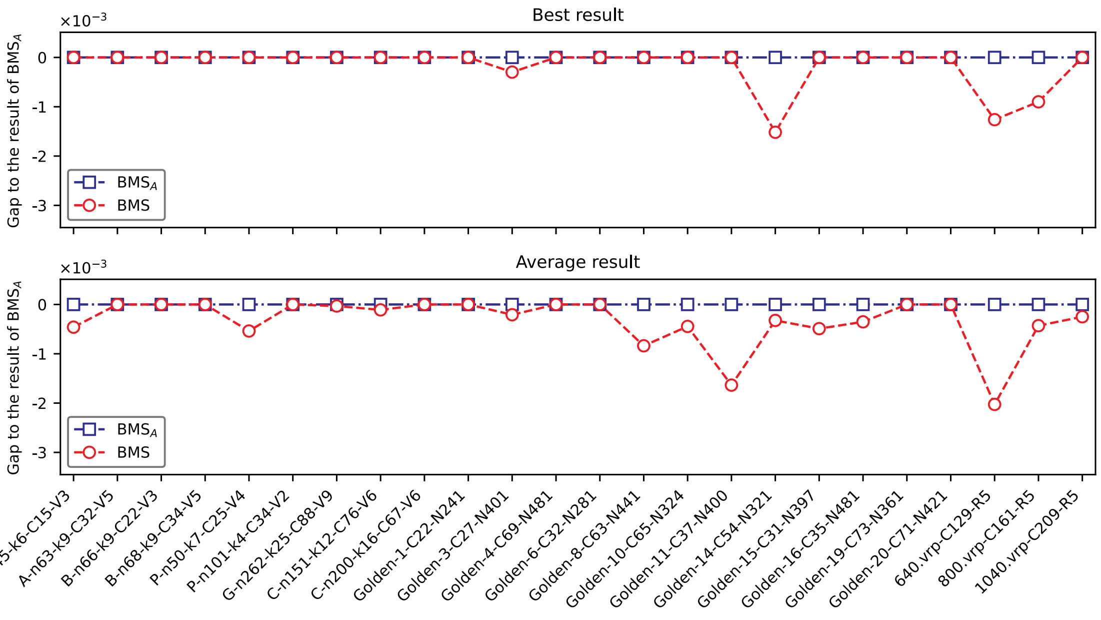
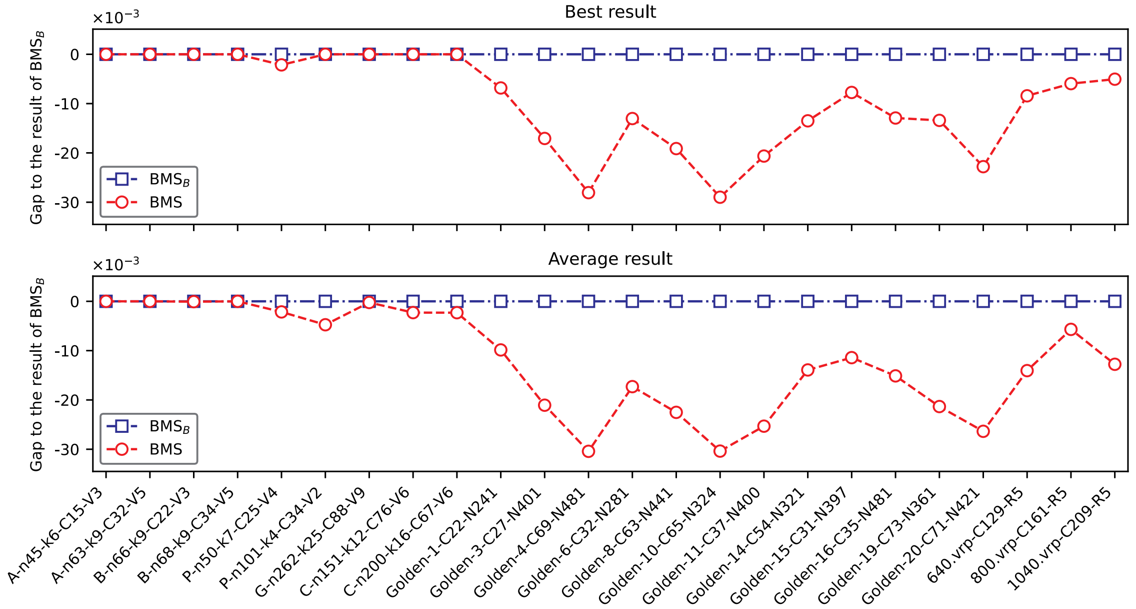


Figure 4. (Color online) Comparison Between BMS (with Bilevel Hybrid Neighborhood Search) and BMS$_B$ (with Two-Level Variable Neighborhood Search)



From Figure 4, in terms of the best result, BMS obtains better performances on 16 out of 24 instances than BMS$_B$, and they enjoy same performance on the remaining eight instances. In terms of average result, BMS also demonstrates better performance by finding improved results on 19 instances than BMS$_B$ and the same results on the remaining five instances as BMS$_B$ does. We also observe that BMS performs better on large instances. In particular, it finds improved results in terms of both best and average results on all 15 large instances. These results confirm the superiority of BHNS over TVNS.

6.3. Effectiveness of Tabu-Driven Population Reconstruction Strategy

BMS integrates a TPR strategy to escape from local optima. It utilizes the basic idea of tabu search and neighborhood decomposition strategy. To evaluate its effectiveness, we experimentally compare BMS with its two variants: BMS$_C$ and BMS$_D$. The former is obtained from BMS by disabling the TPR strategy, whereas the latter can be obtained from BMS by replacing the TPR strategy with the simple population initialization strategy. We resort to the running profile to perform performance comparison (Dolan and Moré 2002). It is a natural way to observe the evolution of the best solution value during a search. It is defined as the function $t \longmapsto f(t)$, where t is the computation time and $f(t)$ is the best solution value by t. Figure 5 presents the running profiles of BMS, BMS$_C$, and BMS$_D$ on 24 test instances.

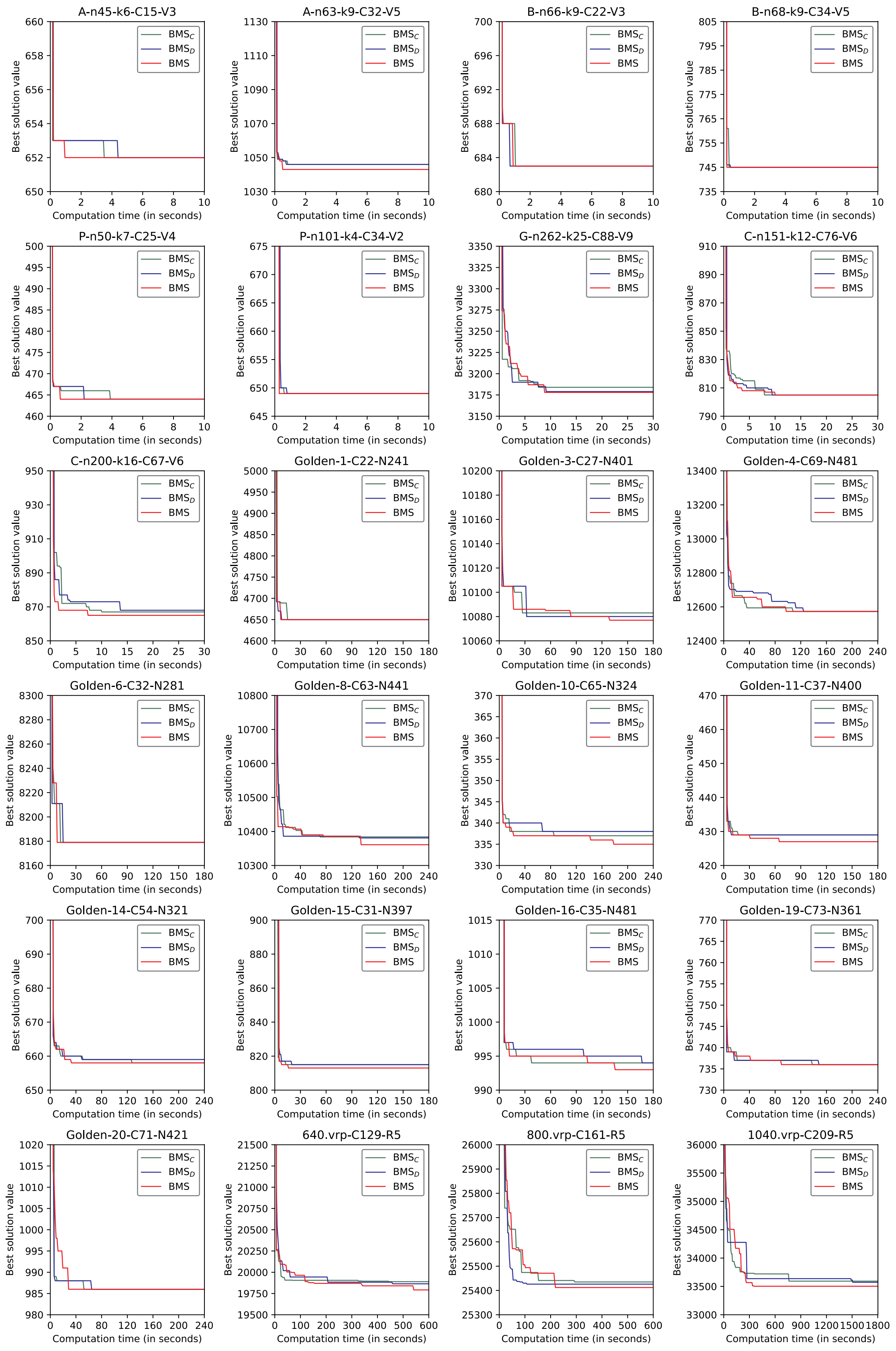
In Figure 5, the x-axis indicates the computation time in seconds, and the y-axis denotes the best solution value obtained so far. From it, we can observe that BMS is more likely to find a better solution than both BMS$_C$ and BMS$_D$. Concerning A-n63-k9-C32-V5, we see that BMS's best solution value decreases quickly at the beginning of the run and then much more slowly, reaching a steady value of about 1,043 after 0.5 second. Both BMS$_C$ and BMS$_D$ share a similar running profile, but they converge to a worse value than BMS. The same observations hold on 23 other instances. These results show a clear role of the tabu-driven population reconstruction strategy in BMS.

7. Conclusion and Future Work

The soft-clustered vehicle routing problem (SoftCluVRP) is an important variant of classical capacitated vehicle routing problems, where customers are divided into clusters and all customers of the same cluster must be served by the same vehicle. It has emerged as an important NP-hard problem with various real-world applications, such as freight transportation and parcel delivery in courier companies. In this paper, we propose an efficient bilevel memetic search (BMS) to solve it. In addition to the general population initialization and population updating modules, BMS integrates three novel and distinct modules: a group matching-based crossover for solution recombination, a bilevel hybrid neighborhood search for local optimization, and a tabu-driven population reconstruction strategy to rebuild population for escaping from local optima.

To demonstrate its effectiveness, we conduct extensive experimental evaluations on three sets of 390 widely used benchmark instances. The results demonstrate that BMS significantly outperforms the state-of-the-art algorithms. It impressively finds new upper bounds on 20 large-scale

Figure 5. (Color online) Running Profiles of BMS (with Tabu-Driven Population Reconstruction), BMS$_C$ (Without Tabu-Driven Population Reconstruction), and BMS$_D$ (with Simple Population Initialization)



instances, and matches 366 best-known upper bounds, which can be viewed as breakthrough progress in this field. We also perform three groups of ablation studies to verify the benefit of its key algorithmic modules. Two research directions should be pursued. The first is to apply BMS to solve soft-clustered capacitated arc-routing problems (Hintsch, Irnich, and Kiilerich 2021) and colored traveling salesman problems (Li et al. 2015, Zhou et al. 2022). The second is to combine metaheuristics with exact solution algorithms, for example, mathematical programming, for solving SoftCluVRP.

Acknowledgments

The authors thank the anonymous referees for their helpful comments and suggestions. They are also grateful to Stefan Ropke, Thibaut Vidal, and Maria Battarra for kindly sharing the benchmark instances.

References

Absi N, Archetti C, Dauzère-Pérès S, Feillet D (2015) A two-phase iterative heuristic approach for the production routing problem. *Transportation Sci.* 49(4):784–795.

Accorsi L, Vigo D (2021) A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems. *Transportation Sci.* 55(4):832–856.

Balas E, Simonetti N (2001) Linear time dynamic-programming algorithms for new classes of restricted TSPs: A computational study. *INFORMS J. Comput.* 13(1):56–75.

Battarra M, Erdoğan G, Vigo D (2014) Exact algorithms for the clustered vehicle routing problem. *Oper. Res.* 62(1):58–71.

Bektaş T, Erdoğan G, Røpke S (2011) Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Sci.* 45(3):299–316.

Cordeau J, Gaudioso M, Laporte G, Moccia L (2006) A memetic heuristic for the generalized quadratic assignment problem. *INFORMS J. Comput.* 18(4):433–443.

Cosma O, Pop PC, Sitar CP (2022) A two-level based genetic algorithm for solving the soft-clustered vehicle routing problem. *Carpathian J. Math.* 38(1):117–128.

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

Dolan ED, Moré JJ (2002) Benchmarking optimization software with performance profiles. *Math. Programming* 91(2):201–213.

Flood MM (1956) The traveling-salesman problem. *Oper. Res.* 4(1):61–75.

Fu ZH, Hao JK (2015) Dynamic programming driven memetic search for the Steiner tree problem with revenues, budget, and hop constraints. *INFORMS J. Comput.* 27(2):221–237.

Glover FW (1989) Tabu search, Part I. *INFORMS J. Comput.* 1(3):190–206.

Golden BL, Wasil EA, Kelly JP, Chao IM (1998) The impact of metaheuristics on solving the vehicle routing problem: Algorithms, problem sets, and computational results. Crainic TG, Laporte G, eds. *Fleet Management and Logistics* (Springer, Boston), 33–56.

Gusfield D (2002) Partition-distance: A problem and class of perfect graphs arising in clustering. *Inform. Processing Lett.* 82(3): 159–164.

Helsgaun K (2000) An effective implementation of the Lin-Kernighan traveling salesman heuristic. *Eur. J. Oper. Res.* 126(1):106–130.

Helsgaun K (2017) An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University, Department of Computer Science, Roskilde, Denmark.

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

Hintsch T (2021) Large multiple neighborhood search for the soft-clustered vehicle-routing problem. *Comput. Oper. Res.* 129:105132.

Hintsch T, Irnich S (2018) Large multiple neighborhood search for the clustered vehicle-routing problem. *Eur. J. Oper. Res.* 270(1):118–131.

Hintsch T, Irnich S (2020) Exact solution of the soft-clustered vehicle-routing problem. *Eur. J. Oper. Res.* 280(1):164–178.

Hintsch T, Irnich S, Kiilerich L (2021) Branch-price-and-cut for the soft-clustered capacitated arc-routing problem. *Transportation Sci.* 55(3):687–705.

Hoogeboom M, Battarra M, Erdogan G, Vigo D (2016) Erratum: Exact algorithms for the clustered vehicle routing problem. *Oper. Res.* 64(2):456–457.

Kuhn HW (2005) The Hungarian method for the assignment problem. *Naval Res. Logist.* 52(1):7–21.

Lai X, Hao J, Fu Z, Yue D (2021) Neighborhood decomposition based variable neighborhood search and tabu search for maximally diverse grouping. *Eur. J. Oper. Res.* 289(3):1067–1086.

Li F, Golden B, Wasil E (2005) Very large-scale vehicle routing: New test problems, algorithms, and results. *Comput. Oper. Res.* 32(5):1165–1179.

Li B, Wu G, He Y, Fan M, Pedrycz W (2022) An overview and experimental study of learning-based optimization algorithms for the vehicle routing problem. *IEEE/CAA J. Automatica Sinica* 9(7):1115–1138.

Li J, Zhou M, Sun Q, Dai X, Yu X (2015) Colored traveling salesman problem. *IEEE Trans. on Cybernetics* 45(11):2390–2401.

Lü Z, Hao JK (2010) A memetic algorithm for graph coloring. *Eur. J. Oper. Res.* 203(1):241–250.

Lysgaard J, Letchford AN, Eglese RW (2004) A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Math. Programming* 100(2):423–445.

Miller DL (1995) A matching based exact algorithm for capacitated vehicle routing problems. *INFORMS J. Comput.* 7(1):1–9.

Porumbel D, Hao JK, Kuntz P (2010) An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring. *Comput. Oper. Res.* 37:1822–1832.

Ramos-Figueroa O, Quiroz-Castellanos M, Mezura-Montes E, Schütze O (2020) Metaheuristics to solve grouping problems: A review and a case study. *Swarm Evolutionary Comput.* 53:100643.

Setubal JC (1996) Sequential and parallel experimental results with bipartite matching algorithms. Technical report, University of Campinas, Institute of Computing, Campinas, Brazil.

Sevaux M, Sörensen K (2008) Hamiltonian paths in large clustered routing problems. *Proc. EU/Meeting 2008 Workshop Metaheuristics Logist. Vehicle Routing*, vol. 8, 411–417.

Su Z, Lü Z, Wang Z, Qi Y, Benlic U (2020) A matheuristic algorithm for the inventory routing problem. *Transportation Sci.* 54(2):330–354.

Vidal T, Battarra M, Subramanian A, Erdogan G (2015) Hybrid metaheuristics for the clustered vehicle routing problem. *Comput. Oper. Res.* 58:87–99.

Wang J, Sun Y, Zhang Z, Gao S (2020) Solving multitrip pickup and delivery problem with time windows and manpower planning using multiobjective algorithms. *IEEE/CAA J. Automatica Sinica* 7(4):1134–1153.

Whitley D (1994) A genetic algorithm tutorial. *Statist. Comput.* 4(2):65–85.

Zhou Y, Hao JK, Duval B (2017) Opposition-based memetic search for the maximum diversity problem. *IEEE Trans. Evolutionary Comput.* 21(5):731–745.

Zhou Y, Hao JK, Glover F (2019) Memetic search for identifying critical nodes in sparse graphs. *IEEE Trans. Cybernetics* 49(10):3699–3712.

Zhou Y, Xu W, Fu ZH, Zhou M (2022) Multi-neighborhood simulated annealing-based iterated local search for colored traveling salesman problems. *IEEE Trans. Intelligent Transportation Systems* 23(9):16072–16082.

Zhou Y, Hao JK, Fu ZH, Wang Z, Lai X (2021) Variable population memetic search: A case study on the critical node problem. *IEEE Trans. Evolutionary Comput.* 25(1):187–200.