A novel tri-individual memetic search algorithm for the set orienteering problem

Mu He[a], Qinghua Wu[b], Jin-Kao Hao[c], Yongliang Lu[d,*], Yuning Chen[e]

[a]*School of Public Finance and Taxation, Zhongnan University of Economics and Law, 430073 Wuhan, China*
[b]*School of Management, Huazhong University of Science and Technology, 430074 Wuhan, China*
[c]*LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*
[d]*School of Economics and Management, Fuzhou University, 350108 Fuzhou, China*
[e]*College of System Engineering, National University of Defense Technology, Changsha, China*

Computers & Operations Research 107648, 2026 https://doi.org/10.1016/j.cor.2026.107648

Abstract

The Set Orienteering Problem is a generalization of the popular Orienteering Problem, where customers are grouped into mutually exclusive clusters, and the profit of each cluster is collected by visiting at least one customer in the respective cluster. For this NP-hard problem, we develop a novel heuristic algorithm rooted in the memetic search framework. This algorithm incorporates a pre-processing phase to accelerate the search, an inherit-mutate-repair crossover operator for generating high-quality offspring solutions, and a local search that integrates strategic oscillation to adaptively explore both feasible and infeasible regions of the search space. Other innovative features of our method include a fast tri-individual population, a solution-based tabu strategy based on hashing techniques, and an adaptive mutation operator. Extensive experimental results on two sets of 496 large benchmark instances demonstrate the strong competitiveness of the proposed approach in comparison to state-of-the-art heuristics documented in the literature. Specifically, for the 192 large original benchmark instances, the proposed algorithm outperforms the current best-known solutions for 27 instances, while matching all other established results. For the 304 new larger benchmark instances, the proposed algorithm improves the best-known solutions for 138 instances. Further experiments are carried out to provide deeper insights into the performance of the algorithm by analyzing its primary elements.

Keywords: Heuristics; Set orienteering problem; Memetic search; Strategic oscillation

1. Introduction

The Orienteering Problem (OP) is a well-known classic routing optimization problem that has garnered extensive research attention in the literature. Originating from the outdoor orienteering sport (Tsiligirides, 1984), this problem is inspired by the activity where participants strive to visit various checkpoints and accumulate points within a limited time frame. The primary objective of OP is to identify an optimal subset of checkpoints to visit within the specified time limit so as to maximize the total points gathered.

*Corresponding author.
Email addresses: muhe2025@zuel.edu.cn (Mu He), qinghuawu1005@gmail.com (Qinghua Wu), jin-kao.hao@univ-angers.fr (Jin-Kao Hao), luyonglianglyl@gmail.com (Yongliang Lu), yuning_chen@nudt.edu.cn (Yuning Chen)

Due to its broad applicability in real-world applications, OP has attracted substantial interest over the past several decades, as evidenced by surveys (Vansteenwegen et al., 2011; Gunawan et al., 2016).

In this study, we concentrate on a recent variation of OP known as the Set Orienteering Problem (SOP), which was first introduced in Archetti et al. (2018). In SOP, customers are divided into disjoint clusters, and each cluster is associated with a profit that is earned only when at least one customer belonging to that cluster is visited. A single vehicle is available to visit selected customers under a predefined travel time threshold. The objective is to identify a route that visits a subset of customers such that the total accumulated profit is maximized, while adhering to the given travel time constraint.

Similar to various OP, SOP finds extensive practical utility across contemporary logistics, tourism, and transportation industries, etc. As highlighted in Archetti et al. (2018), a common application is linked to the distribution of mass products, where each supply chain (cluster) is made up of a group of retailers (customers), and a carrier selects to serve one retailer in the supply chain, after which the products are distributed internally among all retailers within that particular supply chain.

1.1. Literature review

As a relatively new problem, SOP has already attracted considerable research interest in recent years. Archetti et al. (2018) presented the first study on SOP, where they presented a mathematical model and a matheuristic algorithm (MA). This MA comprises two phases: the initial phase employs a greedy algorithm to construct an initial solution by iteratively adding the nearest vertex from the cluster with the highest profit; the subsequent phase then leverages a tabu search to enhance the solution. Furthermore, Archetti et al. (2018) provided benchmark instances adapted from the Generalized Traveling Salesman Problem, alongside random instances, to conduct a comprehensive evaluation of algorithm's performance.

Later, Pěnička et al. (2019) proposed a new Integer Linear Programming model for SOP, which outperforms the formulation of Archetti et al. (2018) through a reduction in the number of decision variables. To solve large instances, they developed a Variable Neighborhood Search (VNS) algorithm that incorporates two local search operators, as well as two perturbation operators. In another study, Golmankhaneh & Keshtkaran (2020) presented a modified Ant Colony System (ACS) algorithm. This approach updates pheromones both locally and globally, enabling efficient optimization. One significant advantage of ACS-SOP is its incorporation of a TSP-based 2-opt operator. Carrabs (2021) introduced a Biased Random-Key Genetic Algorithm (BRKGA) to tackle the SOP. The BRKGA utilizes three local search procedures to enhance solution quality. To further improve the algorithm's efficiency, the authors incorporated several specialized data structures (a hash-table and a three-dimensional matrix) to prevent redundant computations.

Recently, Dontas et al. (2023) proposed an adaptive memory matheuristic (AMMH). This algorithm utilizes a local search approach with several move operators for cluster relocation and exchanges, supported by a tabu strategy for search diversification. Additionally, it incorporates an intensification procedure involving customer insertions and deletions, optimized through an integer programming model. AMMH is further enhanced with an adaptive memory mechanism that gathers high-quality routes discovered during the search process, which are then used to recombine promising cluster sequences, thereby generating new initial solutions. Then, Archetti et al. (2024) devised an exact branch-and-cut algorithm for SOP. Empirical results demonstrate that this algorithm performs efficiently for problem instances involving up to 100 customers. Additionally, they proposed a new formulation and introduced various categories of valid inequalities to enhance performance.

Finally, Lu et al. (2024) introduced a hybrid evolutionary algorithm (HEA), which combines a cluster-based crossover operator to preserve common clusters, a randomized mutation operator to ensure diversity, and a two-stage local search process to improve offspring solutions. The HEA contributes significantly to the field of SOP optimization and shows the potential of hybridized algorithms in attaining high-quality solutions.

In addition, the literature reveals several other node routing problems that involve clustering customers, exhibiting similarities with the SOP examined in this study. These related problems offer valuable insights and potential methodologies for addressing SOP.

One such problem is the Clustered Orienteering Problem (COP) (Angelelli et al., 2014), where vertices are categorized into clusters, each of which is linked to a specific profit. However, unlike SOP, the profit of a cluster is only collected when all vertices within that cluster are visited. Another closely related problem is the Generalized Traveling Salesman Problem (GTSP) (Noon & Bean, 1993), where vertices are divided into clusters, and the objective is to identify a route that visits precisely one vertex from each cluster with minimum cost. The primary difference is that GTSP mandates visiting every cluster exactly once.

Recently, Derya et al. (2020) introduced an extension of GTSP, namely the Selective Generalized Traveling Salesman Problem (SGTSP). This problem relaxes the GTSP constraint of visiting each cluster exactly once. In contrast to SOP, SGTSP assigns a profit to an individual vertex, whereas SOP associates a profit with each cluster. Additionally, the Family Traveling Salesman Problem (FTSP) (Morán-Mirabal et al., 2014) also shares similarities with SOP, where vertices are organized into families (clusters), and the goal is to visit a predetermined number of vertices within each cluster while minimizing the overall cost. Unlike SOP, FTSP requires a specific number of vertices to be visited in each cluster.

1.2. Motivations and contributions

When each cluster contains precisely one vertex, the SOP reduces to the classic NP-hard OP. However, in comparison with the traditional OP, the SOP introduces additional constraints that increase its complexity. As a result, solving the SOP poses significant computational challenges.

From the literature review, exact algorithms have only achieved optimal solutions for small-scale SOP instances, while existing heuristics have not consistently performed well in large SOP instances. This highlights the need for enhancing both the solution quality and computational efficiency of heuristic algorithms when tackling larger instances of SOP.

Additionally, memetic search (Hao, 2012; Neri & Cotta, 2012) has shown success in solving various combinatorial optimization problems, including OP and some of its variations. Through the integration of population-driven global search and neighborhood-based local search, memetic search helps to strike a desirable balance between intensification and diversification.

Motivated by these observations, we propose a novel tri-individual memetic search (TMS) algorithm dedicated to SOP. Our computational experiments demonstrate that the presented TMS algorithm surpasses existing heuristic methods in terms of both solution quality and computational efficiency. The primary contributions can be outlined as follows.

- **Methodological contributions:** The proposed TMS algorithm presents several effective and novel methodological designs through the tailored integration and improvement of existing search concepts. First, the algorithm incorporates a local search that integrates strategic oscillation, enabling adaptive

exploration of both feasible and infeasible regions within the search space. Within the local search, a solution-based tabu strategy utilizing hashing techniques is employed, effectively preventing both short-term and long-term cycling during neighborhood search while keeping computational overhead low. Second, this study proposes an inherit-mutate-repair crossover operator for generating high-quality offspring solutions. This operator features an adaptive mutation mechanism that dynamically tracks and updates cluster selection probabilities, allowing the search to systematically focus on the most promising areas of the solution space. Third, the algorithm utilizes a small population consisting of only three individuals, which enables efficient pattern extraction while maintaining computational efficiency. Our analysis demonstrates that the collective patterns derived from these three individuals significantly contribute to the generation of promising offspring solutions.

- **Computational contributions:** We assess our algorithm using two sets of 496 large-scale instances from the literature, demonstrating its highly competitive performance. Specifically, for the 192 large original SOP instances, our algorithm outperforms the current best-known solutions in 27 instances, while matching all other established results. For the 304 new larger SOP instances, it improves upon the best-known solutions for 138 instances. These newly obtained lower bounds serve as valuable references for assessing new SOP algorithms, especially in the context of large-scale instances. Additionally, we conduct a comprehensive analysis to assess the impact of key algorithmic elements on the overall performance.

This paper is structured as follows. Section 2 outlines the problem statement. Section 3 elaborates on the proposed algorithm, followed by Section 4 which presents the computational results and compares them with state-of-the-art algorithms. Section 5 delves into critical components of the algorithm, leading to the conclusion in Section 6.

2. Problem description

The problem under consideration can be formally defined as follows. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represent a complete weighted graph, where $\mathcal{V} = \{0, 1, 2, \ldots, n\}$ denotes the vertex set and $\mathcal{E} = \{(i, j) : i, j \in \mathcal{V}, i \neq j\}$ represents the edge set. The vertex 0 functions as the depot, serving as both the origin and the terminal point of the vehicle tour, while the remaining vertices $\{1, 2, \ldots, n\}$ correspond to customers. Each edge $(i, j) \in \mathcal{E}$ $(i, j \in \mathcal{V})$ has an assigned non-negative travel cost w_{ij}, which represents the travel time between vertices i and j. Consider $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ as a cover of $\mathcal{V}\backslash\{0\}$, where $\cup_{i=1}^{k} \mathcal{C}_i = \mathcal{V}\backslash\{0\}$. Each $\mathcal{C}_i \in \mathcal{C}$ is a non-empty subset of $\mathcal{V}\backslash\{0\}$ representing a cluster, and each vertex $i \in \mathcal{V}\backslash\{0\}$ is affiliated with exactly one cluster. These clusters are mutually exclusive, that is $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for all $i \neq j, 1 \leq i, j \leq k$. Each cluster $\mathcal{C}_i \in \mathcal{C}$ is associated with an integer profit p_i, which is collected when at least one customer from $\mathcal{C}_i$ is visited. A single vehicle is positioned at depot 0 to serve these customers, with the route's duration constrained by a time limit T_{max}. The aim of SOP is to determine a vehicle route that starts from depot 0, visits a subset of customers, and finishes at depot 0, while maximizing the total profit accumulated within the specified time budget T_{max}.

Figure 1 illustrates a solution for a specific SOP instance featuring one depot and 14 customers (vertices), along with 6 clusters differentiated by various colors. Vertices sharing the same color are part of the same cluster. The clusters are labeled as $\mathcal{C}_1 = \{1, 2\}$, $\mathcal{C}_2 = \{3, 4\}$, $\mathcal{C}_3 = \{5, 6, 7\}$, $\mathcal{C}_4 = \{8, 11\}$, $\mathcal{C}_5 = \{9, 10\}$,

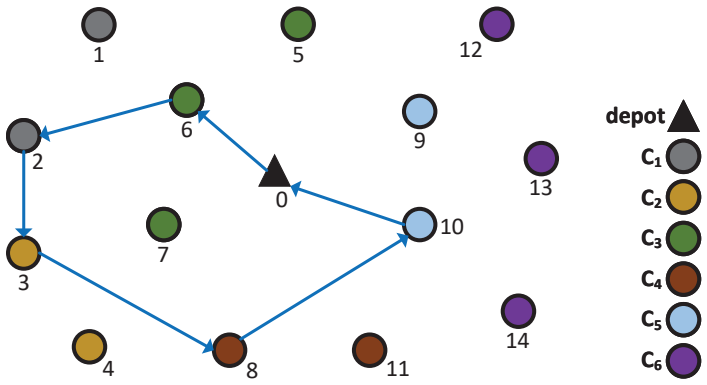


Figure 1: Illustrative example of SOP.

$\mathcal{C}_6 = \{12, 13, 14\}$. Figure 1 displays a route $r = (0, 6, 2, 3, 8, 10, 0)$ that results in the collection of profits from clusters $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$, $\mathcal{C}_4$, and $\mathcal{C}_5$.

3. Tri-individual memetic search for SOP

3.1. General procedure

Memetic search (Hao, 2012; Neri & Cotta, 2012) constitutes a robust hybrid optimization paradigm that integrates the strengths of population-driven genetic search approaches with neighborhood-oriented local optimization techniques. Taking into account the characteristics of SOP, we develop a novel tri-individual memetic search (TMS) algorithm specifically designed for this problem. The proposed TMS algorithm includes several essential components: a pre-processing phase to accelerate the search process, a construction approach for generating initial solutions (Section 3.3), a local search with a solution-based tabu strategy that dynamically navigates both feasible and infeasible spaces (Section 3.4), an inherit-mutate-repair (IMR) crossover operator that generates promising offspring solutions (Section 3.5), and a tri-individual population updating strategy (Section 3.6). Each of these components is integral to the overall performance of the TMS algorithm, providing a comprehensive and efficient approach.

Inspired by Moalic & Gondran (2018); Ding et al. (2019); Zhou et al. (2023); Zou et al. (2025), TMS adopts a small population consisting of merely three individuals, as elaborated in Section 3.3. The general structure of TMS is depicted in Algorithm 1. Before running the algorithm, a pre-processing procedure based on the principles of Carrabs (2021) is performed to eliminate unnecessary vertices, arcs, and clusters from the problem instance. After that, the algorithm begins with population initialization (Section 3.3), which is further enhanced by local search (Section 3.4). Subsequently, TMS enters an iterative loop over multiple generations. In each generation, the IMR operator generates a promising offspring solution, which is then further optimized to achieve a high-quality local optimal solution via local search (Section 3.4). Finally, a population updating mechanism (Section 3.6) is employed to decide whether each improved offspring solution should be accepted or discarded. This iterative process continues until a specified termination criterion is met, such as a predetermined time threshold or maximum number of generations. Detailed descriptions of the algorithmic elements are presented in the subsequent subsections.

Algorithm 1: Tri-individual memetic search (TMS) for SOP

Input: A SOP Instance $\mathcal{G}$
Output: Best found solution S^*

$\mathcal{G} \leftarrow$ PreProcessing($\mathcal{G}$)
$\mathcal{P} = \{S_1, S_2, S_3\} \leftarrow$ PopulationInitialization()
for $S_i \in \mathcal{P}$, $i \in \{1, 2, 3\}$ **do**
    $S_i \leftarrow$ LocalSearch(S_i)
$S^* \leftarrow arg\ max_{\forall S_i \in \mathcal{P}} f(S_i)$ /*track the best found solution*/
while *termination criterion is not met* **do**
    $S' \leftarrow$ IMR($\mathcal{P}$) /*produce an offspring via the IMR operator*/
    $S \leftarrow$ LocalSearch(S') /*improve through local optimization*/
    if $f(S) > f(S^*)$ **then**
        $S^* \leftarrow S$ /* update the best found solution */
    $\mathcal{P} \leftarrow$ PopulationUpdating($\mathcal{P}$, S)
return S^*

3.2. Solution representation and evaluation

SOP involves the determination of the visited clusters and permutation of the vertices. Thus, a solution S is represented by $S = \{b, r\}$ at both the cluster and customer levels. Specifically, at the cluster level, the solution S is encoded as a binary array with a length equal to the total number of clusters, i.e., $b = (z_1, \ldots, z_k)$, where $z_i = 1$ indicates that the i-th cluster is included in solution S while $z_i = 0$ otherwise. At the customer level, S is represented as a route r, denoted as a permutation of vertices, i.e., $r = (0, v_1, v_2, ..., v_i, 0)$.

Given these notations, the evaluation function used in our study quantifies the total profit collected by $S = \{(z_1, \ldots, z_k), (0, v_1, v_2, ..., v_i, 0)\}$, expressed as

$$f(S) = \sum_{i=1}^{k} p_i z_i \tag{1}$$

in which k denotes the total number of clusters, and p_i represents the profit associated with the i-th cluster.

3.3. Population initialization

Managing a large population can be time-consuming. Several recent studies show that memetic algorithms can achieve remarkable performance by employing a small population consisting of just two or three individuals (Moalic & Gondran, 2018; Ding et al., 2019; Zhou et al., 2023; Zou et al., 2025). Based on these results, our algorithm utilizes a population of merely three individuals to accelerate the search process.

Accordingly, TMS initializes its small population $\mathcal{P} = \{S_1, S_2, S_3\}$ by a simple greedy heuristic presented in Lu et al. (2024). Specifically, the initialization procedure (as outlined in Algorithm 2) constructs each feasible solution through the following three phases.

Phase 1: Partial Solution Construction. A partial solution S is first generated by randomly selecting 20% of the clusters (the mandatory depot cluster included) from the original SOP instance $\mathcal{G}$. This subset of clusters forms a smaller Generalized TSP (GTSP) instance, which is solved by the state-of-the-art solver GLKH[1] (Helsgaun, 2015) (adapted from the popular TSP solver LKH (Helsgaun, 2000)). The resulting

[1]The source code of GLKH is openly accessible at http://akira.ruc.dk/~keld/research/GLKH/

Algorithm 2: Solution initialization procedure

Input: A SOP Instance $\mathcal{G}$
Output: A feasible solution S

```
/* Phase 1: Partial solution construction */
Randomly select 20% of clusters (depot included) to form GTSP instance G'
S ← GLKH(G') /* Solve the GTSP using the state-of-the-art GLKH solver */
while t(S) > T_max do
    v ← arg min_{v∈S} p_{C(v)} /* C(v): cluster containing v */
    Remove v from S
/* Phase 2: Greedy completion */
Sort all unvisited clusters by Δp_i / Δcost_{v*} in descending order /* Δcost_{v*} is the best insertion cost */
for each unvisited clusters C_i do
    if t(S) + Δcost_{v*} ≤ T_max then
        Insert v* at optimal position in S
/* Phase 3: Post-Processing */
S ← LocalSearch(S) /* Section 3.4 */
return S
```

solution S visits precisely one vertex from each selected cluster. If the total travel time $t(S)$ exceeds T_{max}, we iteratively remove vertices associated with the lowest-profit clusters until the given travel time limit T_{max} constraint is satisfied.

Phase 2: Greedy Completion. The partial solution S is then completed through a profit-efficient insertion heuristic. All unvisited clusters $\mathcal{C}_i \in \mathcal{C}$ are sorted in descending order by the ratio $\frac{\Delta p_i}{\Delta cost_{v^*}}$ (the added cluster profit Δp_i relative to the added travel cost $\Delta cost_{v^*}$), where $v^* \in \mathcal{C}_i$ denotes the vertex that yields the minimal increase in tour length when inserted at the best position in the partial solution S. For each cluster $\mathcal{C}_i$ in this ordered list, we attempt to insert $v^* \in \mathcal{C}_i$ at the position that minimizes the total tour length increase, while ensuring that $t(S) \leq T_{max}$.

Phase 3: Post-Processing. The completed solution is further refined using the local search procedure outlined in Section 3.4.

To ensure diversity within the initial population $\mathcal{P}$, any newly generated solution that overlaps with an existing solution in the population by more than ρ of its clusters is regenerated using the aforementioned greedy construction algorithm.

The threshold ρ depends on the instance data. For an instance with k total clusters, the effective threshold is $\rho \cdot k$ clusters (rounded up to the nearest integer). We found that lower thresholds led to solutions that were too dissimilar, making it difficult to inherit valuable patterns from parent solutions. On the other hand, higher thresholds resulted in overly similar solutions, causing premature convergence.

Through extensive preliminary testing, we determined that $\rho = 0.8$ achieves a favorable trade-off in terms of solution quality. Additionally, if most of the existing clusters can be selected, it could happen that most solutions overlapping by more than ρ in the selected clusters. In such cases, we apply the following adaptive mechanism: if the newly generated solution shares more than ρ of its clusters with an existing solution in the population three consecutive times (indicating that the current threshold may be too strict for a particular instance), the threshold is automatically increased by 5%.

3.4. Local search

The local search process constitutes a crucial element of TMS, serving as a key driver of the intensification process. Typically, this process explores the solution space by iteratively moving from the current solution to its neighboring solution. The neighborhood encompasses solutions that are immediately adjacent or closely positioned relative to the given solution within the search space. During each iteration, it identifies the optimal neighboring solution to replace the current solution.

The framework of our local search process is outlined in Algorithm 3. Beginning from a feasible solution S, this process initially executes a feasible local search and then executes a strategic oscillation search on the improved solution S_b obtained in the first phase.

Algorithm 3: Local search

Input: A feasible solution S

Output: Improved solution S_b

1 $S_b \leftarrow \text{FeasibleLocalSearch}(S)$

2 $S_b \leftarrow \text{StrategicOscillation}(S_b)$

3 **return** S_b

3.4.1. Neighborhoods

Local search commonly utilizes move operators that facilitate the transformation of a given solution S into a neighboring solution S'. Let $\mathcal{K}(S)$ represent the set of clusters in S, while $\overline{\mathcal{K}}(S)$ denote the set of these unselected clusters. The local search jointly explores three distinct neighborhoods using the following move operators:

- **Add**: Insert the best vertex from $\mathcal{C}_i \in \overline{\mathcal{K}}(S)$ into the best position in S;
- **Swap**: Exchange a vertex u from $\mathcal{C}_i \in \mathcal{K}(S)$ with the best vertex v from $\mathcal{C}_i \in \overline{\mathcal{K}}(S)$. Then, place v in the best position within the current solution S;
- **Drop**: Remove a vertex from $\mathcal{C}_i \in \mathcal{K}(S)$ from the current solution.

The best vertex v in the cluster $\mathcal{C}_i$ corresponds to the vertex that results in the least increase in tour cost when inserted between consecutive vertices i and j in the current tour. The best insertion position is determined by minimizing the additional travel cost between each pair of consecutive vertices in S. As inserting a new cluster into the current tour is a frequently applied operation in our algorithm, it is crucial to assess the extent to which the tour cost increases in order to determine its feasibility. Through the use of the insertion cost matrix (Carrabs, 2021) generated during the pre-processing phase, we can rapidly retrieve both the minimum insertion cost for cluster $\mathcal{C}_i$ between any two vertices in the current tour, as well as the specific vertex from $\mathcal{C}_i$ that should be inserted to achieve that insertion cost. Rather than recalculating this information for each insertion, this strategy allows for its computation to be carried out only once during the pre-processing phase and stored in a three-dimensional matrix.

The utility score of each move is calculated as $\Delta = \frac{\Delta p(mv)}{\Delta c(mv)}$, where $\Delta p(mv)$ and $\Delta c(mv)$ represent the profit and cost change resulting from the application of mv, respectively. To avoid irrational ranking caused by different sign combinations of profit and cost variation, we sort candidate moves by fixed priority levels

in actual implementation. For moves within the same priority level, we further compare their Δ values to pick the best one:

1. $\Delta p(mv) > 0$, $\Delta c(mv) < 0$: The most ideal case simultaneously boosts profit and cuts travel cost, which occurs in high-quality *Swap* moves. Such moves have the highest selection priority, and moves with smaller Δ values are preferred.
2. $\Delta p(mv) > 0$, $\Delta c(mv) > 0$: Profit increases along with higher travel cost, which covers all *Add* moves and partial *Swap* moves. Larger Δ values indicate higher profit growth per unit cost rise, so moves with greater scores are favored.
3. $\Delta p(mv) < 0$, $\Delta c(mv) < 0$: Profit decreases while travel cost is reduced, commonly seen in *Swap* and *Drop* moves. This index reflects profit loss per unit saved cost; smaller values mean milder profit loss, thus being more favorable.
4. $\Delta p(mv) < 0$, $\Delta c(mv) > 0$: Moves lead to lower profit and higher travel cost, with few optimization benefits. They occur in *Swap* moves and are adopted only when all other valid moves are marked tabu, among which we select those with larger Δ values to minimize solution deterioration.

During local search, each neighborhood first selects its best candidate following the above priority and ranking rules, and the algorithm then select the best non-tabu move among all candidates.

3.4.2. Feasible local search

Algorithm 4: Feasible local search

Input: A candidate feasible solution S; maximum allowed consecutive iterations without improvement T; frequency parameter ϵ
Output: Improved solution S_b

```
1  S_b ← S
2  Noimprove ← 0, Iter ← 0 /* Iteration counters */
3  Initialize tabu_list
4  while Noimprove ≤ T do
5      Select the best non-tabu feasible neighboring solution S' from S
6      S ← S' /* Update the current solution */
7      Update tabu_list
8      if NoImprove == 1 || Iter % ϵ == 0 then
9          S ← EnhancedRouteOptimization(S)
10     if f(S) > f(S_b) then
11         S_b ← S
12         Noimprove ← 0
13     Noimprove ← Noimprove + 1, Iter ← Iter + 1
14 return S_b
```

Designed to converge on a high-quality local optimum, the feasible local search process restricts its scope exclusively to feasible solutions. As depicted in Algorithm 4, this process begins with the initialization tasks (lines 1-3), followed by entry into a "while" loop (lines 4-13) to iteratively enhance the input solution S through exploration of three aforementioned neighborhoods (i.e., *Add*, *Swap*, and *Drop*). During each

iteration, the current solution S is updated with the best non-tabu feasible neighboring solution S' (lines 5-6). The tabu list is then adjusted accordingly (line 7), and the tabu strategy is described in Section 3.4.5. Additionally, an enhanced route optimization procedure (outlined in Section 3.4.4) is implemented to reduce the tour length of S (lines 8-9), triggered when either the following conditions is met: (1) a new best solution is discovered or (2) every ϵ iterations are completed. Furthermore, upon achieving a new best objective value for the current solution, the best-found solution S_b and the no-improvement counter are updated accordingly (lines 10-13). If there is no improvement for S_b for T consecutive iterations, S_b is then returned as the final output.

3.4.3. Strategic oscillation

The strategic oscillation method (Glover, 1977; Glover & Hao, 2011) is designed to traverse the boundary between feasible and infeasible solution spaces, unlocking the potential to discover high-quality solutions that might locate beyond the reach of the feasible space alone. By offering the flexibility to explore the infeasible search space, strategic oscillation provides an expanded realm for search and enhances the effectiveness of neighborhood-based local searches. It has been validated in several closely related routing problems, including the studies by Chen et al. (2016); Lu et al. (2018).

In our approach, the oscillation boundary is defined using the maximum travel time constraint. By relaxing this constraint, our search process effectively visits intermediate solutions between the feasible and infeasible spaces. To achieve this, our procedure incorporates a self-adjusting threshold strategy that facilitates traversal between the feasible and infeasible boundaries.

To evaluate the infeasibility degree of a given solution S, we introduce the concept of a critical value $CV(S)$ defined as:

$$CV(S) = \max\{0, t(S) - T_{max}\}, \tag{2}$$

where $t(S)$ represents the duration of the current solution, and $CV(S)$ denotes the total excess duration. Therefore, for a feasible solution S (i.e., $t(S) \leq T_{max}$), $CV(S)$ equals 0.

Afterwards, when considering a candidate solution S in the expanded space, encompassing both feasible and infeasible solutions, its quality is assessed using an expanded function $Z(S)$. This function is a linear combination of the original objective value $f(S)$ and the critical value $CV(S)$, defined as:

$$Z(S) = f(S) - \varphi \times CV(S) \tag{3}$$

Here, φ is a dynamically adjusted parameter that determines the relative significance of the critical value $CV(S)$. By increasing φ, infeasible solutions become less favorable, while decreasing φ encourages the exploration of infeasible space. Specifically, if μ consecutive solutions visited are infeasible, φ is doubled. Conversely, if μ consecutive feasible solutions are visited, φ is halved. The purpose of this adaptive mechanism is to balance the exploration of both feasible and infeasible solutions during the search process. Adhering to the guiding principles, this procedure ensures smooth transitions between two regions, allowing the search to cover diverse regions and prevents excessive deviation into either the feasible or infeasible region. To reduce the sensitivity of φ to instance scales (profit and travel time), its initial value is set to $\varphi = f(S)/t(S)$, which normalizes the penalty according to the characteristics of the current solution S. This design makes the adaptive strategy robust to different problem sizes, profit scales and instance types.

Algorithm 5: Strategic oscillation

Input: A candidate feasible solution S; maximum allowed consecutive iterations without improvement T; frequency parameter ϵ

Output: Improved solution S_b

1 $S_b \leftarrow S$
2 $Noimprove \leftarrow 0$, $Iter \leftarrow 0$ /* Iteration counters */
3 Initialize $tabu_list$
4 **while** $Noimprove \leq T$ **do**
5 Select the best non-tabu neighboring solution S' from S according to Equation (3)
6 $S \leftarrow S'$ /* Update the current solution */
7 Update $tabu_list$
8 **if** $NoImprove == 1$ || $Iter \,\%\, \epsilon == 0$ **then**
9 $S \leftarrow$ EnhancedRouteOptimization(S)
10 Update parameter φ
11 **if** S *is feasible and* $f(S) > f(S_b)$ **then**
12 $S_b \leftarrow S$
13 $Noimprove \leftarrow 0$
14 $Noimprove \leftarrow Noimprove + 1$, $Iter \leftarrow Iter + 1$
15 **return** S_b

The process of the strategic oscillation is outlined in Algorithm 5. Following initial tasks (lines 1-3), the procedure enters a "while" loop to evaluate candidate solutions (lines 4-14). During each iteration, the best non-tabu neighboring solution is chosen based on Equation (3) to replace the current solution (lines 5-6). Subsequently, the tabu list is refreshed, and the enhanced route optimization procedure is executed (lines 7-9), employing the same strategies. Then, the value of φ undergoes adaptive adjustment: it is reduced by half if feasible solutions have been generated in the prior μ consecutive iterations, or doubled if infeasible solutions have been obtained (line 10). Finally, if the current solution is feasible and attains a new optimal objective value, the best-improved solution S_b and the no-improvement counter are updated accordingly (lines 11-14). The process terminates when no improvement in the best-found solution S_b is detected for T consecutive iterations.

It is worth noting that no explicit repair operator is employed during the strategic oscillation. Instead, the search automatically returns to the feasible region by adaptively adjusting the penalty parameter φ, which imposes an increasing penalty on infeasible solutions when the search persists in the infeasible space. Empirical observations show that many improved feasible solutions are obtained shortly after the search returns from the infeasible region. Specifically, for those returns that lead to an improvement, the average number of further iterations required is typically within 10. This confirms that temporarily exploring the infeasible space helps the search escape local optima and enhances solution quality.

3.4.4. Enhanced route optimization procedure

As mentioned in Sections 3.4.2 and 3.4.3, our enhanced route optimization process is employed to refine the route permutation of the current solution S, triggered when either: (1) a new optimal solution is discovered or (2) every ϵ iterations are completed. As detailed in Algorithm 6, this procedure involves the iterative application of a dynamic programming (DP) and a Lin-Kernighan TSP heuristic (Helsgaun, 2000) until no additional improvements can be found for S. The enhanced route optimization includes two

sub-problems: the Shortest Path Problem (SPP) for the fixed-order GTSP reformulation, and the Traveling Salesman Problem (TSP).

Algorithm 6: Enhanced route optimization procedure

Input: A solution S
Output: Improved solution S_b

```
improve_tag ← true
while improve_tag is true do
    improve_tag ← false
    /* Shortest Path Problem */
    S' ← DP(S) /* Solving SPP using dynamic programming */
    if t(S') < t(S) then
        S ← S'
        improve_tag ← true
    /* Traveling Salesman Problem */
    S' ← LKH(S) /* Solving TSP using LKH heuristic */
    if t(S') < t(S) then
        S ← S'
        improve_tag ← true
S_b ← S
return S_b
```

Shortest Path Problem (SPP). The Shortest Path Problem (SPP) stands as a foundational optimization task in graph theory and network analysis. Dynamic programming (DP) is a powerful technique employed to address the SPP. In the context of SPP, dynamic programming breaks down the problem into smaller subproblems and solves them recursively, gradually building up a solution by combining optimal solutions to the subproblems. When applied to SOP, the DP procedure involves substituting each vertex $v_i \in S, 1 \leq i \leq j$ with an alternative vertex from the same cluster in a given path $r = (0, v_1, v_2, ..., v_j, 0)$, where 0 represents the depot. A permutation $(\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_j)$ denotes the clusters visited by the current solution and $Len_i(v)$ represents the length of the shortest path connecting a vertex v in cluster $\mathcal{C}_i$ to the terminal vertex (depot), while traversing precisely a single vertex from each cluster $\mathcal{C}_{i+1}, \mathcal{C}_{i+2}, \ldots, \mathcal{C}_j$. The optimal cost can be calculated recursively as follows.

$$Len_j(v) = w_{v,0} \quad v \in \mathcal{C}_j \tag{4}$$

$$Len_i(u) = \min_{v \in \mathcal{C}_{i+1}} \{w_{u,v} + Len_{i+1}(v)\} \quad 0 \leq i \leq j-1, u \in \mathcal{C}_i \tag{5}$$

The DP procedure to solving the SPP offers an efficient and optimal solution, and it avoids redundant computations by storing and reusing intermediate results, significantly reducing the overall computational complexity compared to other methods.

Traveling Salesman Problem (TSP). The Lin-Kernighan heuristic (Lin & Kernighan, 1973) ranks among the top-performing approaches for TSP, renowned for its exceptional solution quality. This algorithm operates through "k-opt" moves, which replaces k edges in the current tour with k new edges to construct an improved tour. By systematically exploring progressively larger k-opt moves (typically $2 \leq k \leq 5$), the algorithm dynamically adjusts its search depth to escape local optima and achieves an optimal or

nearly optimal solution. A significant advancement was achieved by Helsgaun (2000) through the LKH[2] implementation of the Lin-Kernighan algorithm, which consistently delivers near-optimal solutions for large-scale TSP instances with remarkable efficiency. In this study, we adopt LKH to solve the TSP sub-problem.

3.4.5. Tabu strategy

A key challenge in neighborhood-based local search lies in efficiently avoiding repeated exploration of solutions, which leads to short-term and long-term cyclic behaviors. To tackle this issue effectively, this study adopts a solution-based tabu strategy (Woodruff & Zemel, 1993).

Traditional attribute-based tabu strategy restricts reverse moves within a fixed iterations called tabu tenure. Nevertheless, this approach only mitigates short-term cycles, and may fail to prevent long-term cycles caused by revisiting previously explored solutions. As the search proceeds, some early tabu decisions lose relevance, leading to repeated access to the same solutions. On the contrary, the solution-based tabu strategy precisely records every visited solution, enabling rapid determination of the tabu status for new solutions and avoiding revisiting any previously visited solution.

Rather than maintaining a large pool of solutions, we use hashing techniques within this strategy to track the visitation status of each solution throughout the search (He et al., 2024; Wu et al., 2024). Hashing techniques (Sobti & Geetha, 2012) utilize mathematical functions that map arbitrary length input data to fixed-size hash values, enabling efficient data retrieval and storage. By mapping each visited solution to a unique integer and associating it with an index in the corresponding hash vector, we effectively track the solution exploration history. Specifically, using a hash function h and a hash vector $\mathcal{H}$, each solution S is mapped to a position in $\mathcal{H}$ by the hash function h. The binary value at the $h(S)$-th position of $\mathcal{H}$ (i.e., $\mathcal{H}[h(S)]$) indicates the status of the solution S: if $\mathcal{H}[h(S)] = 1$, the solution has been previously visited; otherwise, $\mathcal{H}[h(S)] = 0$. Initially, the hash vector $\mathcal{H}$ is set to 0, indicating that no solutions have been visited.

However, hash collisions (i.e., different solutions may map to identical integer values) can occur. To resolve this limitation, we implement a triple-hash verification mechanism, which employs three independent hash vectors operating in parallel, significantly improving solution tracking accuracy. Empirical evidence (Lai et al., 2018; Lu et al., 2023) demonstrates that this multi-hash design significantly reduces collisions compared to single-hash approach, while maintaining $O(1)$ computational complexity for all operations. As Section 5.3 shows, our triple-hash mechanism yields a very low collision rate.

More precisely, for a given solution $S = \{b, r\}$, $b = (z_1, \ldots, z_k)$ where $z_i = 1$ indicates the presence of cluster i in S, $z_i = 0$ indicates its absence. Each cluster $\mathcal{C}_i$ is assigned three predetermined weights $\omega_{ij} = \lfloor i^{\eta} \rfloor$ $(j = 1, 2, 3)$, where η is a positive real number. In this work, we set $\omega_{i1} = \lfloor i^{2.7} \rfloor$, $\omega_{i2} = \lfloor i^{2.8} \rfloor$, and $\omega_{i3} = \lfloor i^{2.9} \rfloor$ to generate three distinct hash functions. Formally, the hash functions are defined as follows:

$$h_j(S) = \left(\sum_{i=1}^{k} \omega_{ij} z_i \right) \bmod \mathcal{L} \tag{6}$$

where $\mathcal{L}$ represents the hash vectors' length, set as 10^8 in our work. Each function corresponds to an independent vector.

[2]The source code of LKH is openly accessible at http://akira.ruc.dk/~keld/research/LKH/

For a given solution S with its corresponding hash value $h_j(S)$, we are able to calculate the hash value of neighboring solution S', generated by $Add(\mathcal{C}_i)$, $Swap(\mathcal{C}_{i1}, \mathcal{C}_{i2})$ and $Drop(\mathcal{C}_i)$ in constant time using Equations (7), (8) and (9), respectively.

$$h_j(S') = (h_j(S) + \omega_{ij} z_i) \bmod \mathcal{L} \tag{7}$$

$$h_j(S') = (h_j(S) - \omega_{i1j} z_{i1} + \omega_{i2j} z_{i2}) \bmod \mathcal{L} \tag{8}$$

$$h_j(S') = (h_j(S) - \omega_{ij} z_i) \bmod \mathcal{L} \tag{9}$$

The criteria for classifying a solution as tabu or non-tabu are outlined below. Initially, three hash vectors, namely $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$, are initialized to zero to signify that no solutions have been previously visited. During each iteration, for a candidate neighboring solution S', we compute the hashing values $h_1(S')$, $h_2(S')$ and $h_3(S')$, and identify the index positions $i_1 = h_1(S')$, $i_2 = h_2(S')$ and $i_3 = h_3(S')$ in the corresponding hash vectors $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$, respectively. If the values at all three indices are 1 (i.e., $\mathcal{H}_1[i_1] = 1$, $\mathcal{H}_2[i_2] = 1$, and $\mathcal{H}_3[i_3] = 1$), we assume that the candidate solution has already been visited and thus marked as tabu. Otherwise, we consider this candidate solution S' as an eligible neighboring solution that has not been visited previously. Then, we update $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$ by setting the value of indices to 1. Figure 2 illustrates the tabu status determination process for a candidate solution S' using the triple-hash verification mechanism. In the provided example, since $\mathcal{H}_1[h_1(S')] = 1$, $\mathcal{H}_2[h_2(S')] = 1$, and $\mathcal{H}_3[h_3(S')] = 1$), the solution S' is excluded from solution transitions.

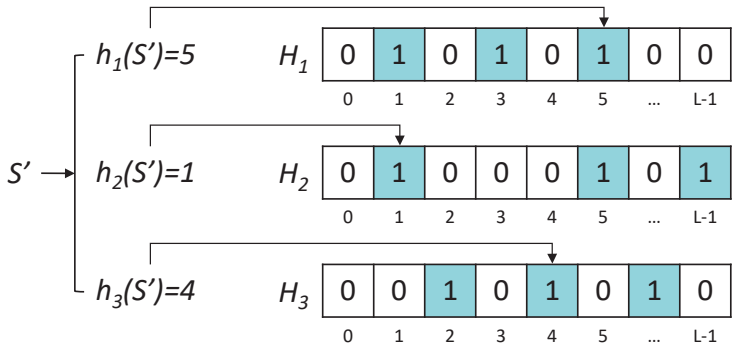


Figure 2: The tabu status determination process for a solution using triple-hash verification mechanism.

3.5. Inherit-mutate-repair crossover

Within memetic search, the crossover operator assumes a vital role for navigating the search space. A well-designed crossover should inherit beneficial traits from parent solutions while simultaneously injecting diversity into the resulting solution (Hao, 2012). In our work, we employ an inherit-mutate-repair (IMR) crossover operator at each generation to produce promising offspring solutions. Drawing inspiration from backbone-based crossover (Wu et al., 2024) and the ordered crossover operator (Oliver et al., 1987), IMR operates in three stages: (1) the inherit recombination stage, (2) the adaptive mutation stage, and (3) the greedy repair stage.

Inherit recombination stage. During the inherit recombination stage, we incorporate the concepts of backbone-based crossover and ordered crossover (OX) into SOP, to create a partial solutions S'. This process relies on three straightforward permutation-based backbone representations of the three parent solutions.

Firstly, given the three distinct solutions S_1, S_2 and S_3 in $\mathcal{P}$, we define the central backbone, denoted as $B_{ct} = S_1 \cap S_2 \cap S_3$, as the collection of clusters common to all three parent solutions (S_1, S_2 and S_3). Similarly, we define the supporting backbone, denoted as B_{sp}, as the set of clusters shared exclusively by two solutions from the population $\mathcal{P}$. It is formally defined as $B_{sp} = ((S_1 \cap S_2) \cup (S_1 \cap S_3) \cup (S_2 \cap S_3)) \backslash B_{ct}$.

Then, as illustrated in Figure 3, the central backbone B_{ct} is extracted based on the original sequence of the three parent solutions. Subsequently, OX is employed to preserve permutation characteristics from all three parents. As depicted in Figure 3, OX involves inheriting a randomly selected fragment of length $\lfloor \frac{|B_{ct}|}{3} \rfloor$ from the first parent, then advancing to the second parent to inherit a fragment of the same length after locating the last cluster (cluster 9 marks the final cluster in S', and this cluster is identified within S_2, continuing the inheritance process from this position in S_2). Finally, the missing visits are filled using sequences from the third parent. It is crucial to highlight that the solution depicted in Figure 3 operates at the cluster level. The visited vertex within each cluster in the corresponding solution will be chosen for inclusion in composing the solution at the customer level. This representation excludes visits to the depot. Consequently, in the end, the depot is integrated into the best position in S'.

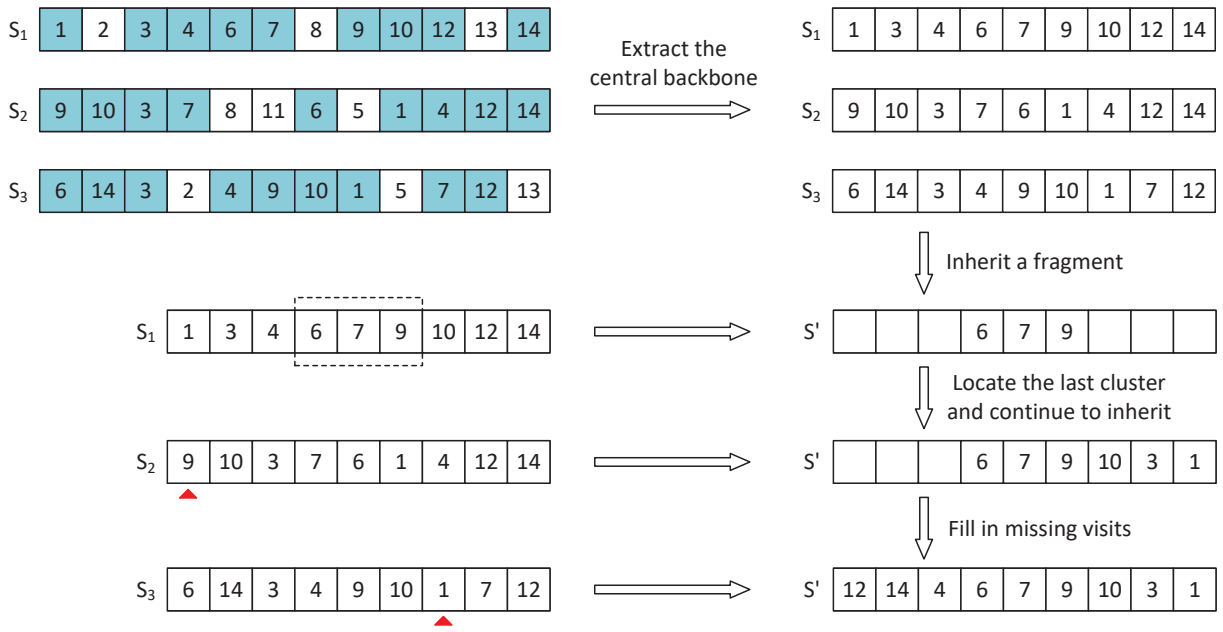


Figure 3: The OX crossover.

Adaptive mutation stage. Through the above inherit recombination stage, we initially derive a partial solution S'. If the size of S' falls below 50% of the average size of the three parent solutions, i.e., $|S'| < 50\% \times (|S_1| + |S_2| + |S_3|)/3$, a cluster is randomly chosen from the supporting backbone B_{sp} and inserted at the optimal position in S'. The procedure repeats until $|S'| \geq \rho \times (|S_1| + |S_2| + |S_3|)/3$. If the size of the solution S' generated by the inherit recombination stage exceeds ρ of the average size of the three

parent solutions, i.e., $|S'| > \rho \times (|S_1| + |S_2| + |S_3|)/3$, we apply the adaptive mutation procedure to promote search diversification. This procedure involves dropping or adding each cluster based on a probability vector, generating a new offspring solution.

In SOP, every cluster exists in one of two states: chosen or not chosen. Furthermore, each cluster allows for two potential actions: inclusion in the candidate solution or exclusion from it. Given the numerous move operations involved in local search procedure, the incorporation of the adaptive methods enhances efficiency. These techniques aid in updating the probability vector and effectively guiding the search direction.

We define a probability vector PR of length k, where PR_i denotes the probability of selecting the cluster $\mathcal{C}_i$ for inclusion in the current solution. Initially, all values of the vector are uniformly set to 0.5, indicating an equal probability for each cluster's selection. Throughout the local search process, whenever cluster $\mathcal{C}_i$ is selected for inclusion, its probability value gets updated according to the formula:

$$PR_i = \beta + (1 - \beta) \times PR_i \tag{10}$$

where β $(0 < \beta < 1)$ represents a reward factor. Conversely, if cluster $\mathcal{C}_i$ is excluded from the current solution, we adjust the probability using a penalization factor γ $(0 < \gamma < 1)$:

$$PR_i = (1 - \gamma) \times PR_i \tag{11}$$

Our probability update scheme is inspired by learning automata (Narendra & Thathachar, 2012). It focuses on enhancing the selection probability for feasible cluster selections and reducing it when clusters are removed. The probability vector undergoes updates during the local search process.

Taking the offspring S' as the starting point, this strategy first drops selected clusters in S' according to the probability vector PR. Then, unselected clusters are added to the solution based on PR. Specifically, for a randomly unselected cluster $\mathcal{C}_i$, we set $z_i = 1$ according to the i-th value in PR. The newly mutated solution serves as the input for subsequent rounds of local search. This strategy facilitates the exploration of new search spaces and maximizes the utilization of the probability vector.

Greedy repair stage. If the tour duration of the resulting offspring solution S' violates the maximum time limit T_{max}, we randomly remove a vertex from S'. This process continues iteratively until the duration of S' falls below or equals T_{max}, i.e., $t(S') \leq T_{max}$. Conversely, if the duration of the offspring solution meets the time limit, we proceed to finalize partial solutions using a greedy approach. This entails iteratively selecting unvisited clusters and applying the *Add* move operator, as detailed in Section 3.4.1, to S' until no additional vertex from an unvisited cluster can be included in S' without surpassing the time budget T_{max}.

3.6. Population updating

The population updating strategy constitutes another vital component of TMS. Given the adoption of a small tri-individual population, the updating strategy should prevent premature convergence to sustain a diverse and viable population throughout the entire search process. For this purpose, we utilize a quality-and-distance population updating strategy, which refreshes the population by taking into account both solution quality and the inter-individual distance within the population. The fundamental idea is to prioritize the inclusion of S' in the population if it is deemed "sufficiently good" based on its objective function and is dissimilar enough from any existing solutions in the population.

To facilitate this, the distance $Dist(S_a, S_b)$ between two solutions S_a and S_b is calculated as the minimum number of flip operations needed to transform S_a into S_b (Lü et al., 2010). This measure, also known as the Hamming distance, serves to quantify the similarity between two solutions, i.e., the number of binary variables that differ between S_a and S_b at the cluster level. This directly corresponds to the minimum number of single-bit flip operations needed to convert one solution into another. Thus, considering a population $\mathcal{P} = \{S_1, S_2, \ldots, S_{|\mathcal{P}|}\}$, the distance between a solution S_i $(1 \leq i \leq |\mathcal{P}|)$ and the population $\mathcal{P}$ is defined as the smallest distance from S_i to any other solution within $\mathcal{P}$, denoted by $D_{S_i,\mathcal{P}}$:

$$D_{S_i,\mathcal{P}} = \min\{Dist(S_i, S_j) | S_j \in \mathcal{P}, S_i \neq S_j\} \tag{12}$$

Then, the quality-and-distance score for the solution $S_i \in \mathcal{P}$ is defined as:

$$QD(S_i, \mathcal{P}) = \lambda \times \widetilde{A}(f(S_i)) + (1 - \lambda) \times \widetilde{A}(D_{S_i,\mathcal{P}}) \tag{13}$$

where λ is a parameter and set to 0.6 as suggested by Lü et al. (2010), and $f(S_i)$ represents the objective value of S_i. The normalized function $\widetilde{A}()$ is defined by:

$$\widetilde{A}(y) = \frac{y - y_{min}}{y_{max} - y_{min} + 1} \tag{14}$$

where y_{max} and y_{min} denote the maximum and minimum values of y within $\mathcal{P}$, respectively. The addition of 1 in the denominator ensures that there is no possibility of a zero denominator.

Algorithm 7: Population updating

Input: An improved offspring solution S'; population $\mathcal{P}$
Output: Updated population $\mathcal{P}$

1 Add S' tentatively to $\mathcal{P}$: $\mathcal{P}' = \mathcal{P} \cup \{S'\}$
2 **for** $i = 1, 2, \ldots, |\mathcal{P}|$ **do**
3 Compute the distance between S_i and $\mathcal{P}'$ using Equation (12)
4 Compute the quality-and-distance score $QD(S_i, \mathcal{P}')$ of S_i based on Equation (13)
5 Identify the solution S_w with the lowest quality-and-distance score in $\mathcal{P}'$: $S_w = \min\{QD(S_i, \mathcal{P}') | i = 1, 2, \ldots, |\mathcal{P}|\}$
6 **if** $S_w \neq S'$ **then**
7 Replace S_w with S': $\mathcal{P} = \mathcal{P} \cup \{S'\} \backslash \{S_w\}$
8 **return** $\mathcal{P}$

The population updating process is detailed in Algorithm 7. To determine whether the enhanced offspring S' should be included into $\mathcal{P}$, we first tentatively add S' to $\mathcal{P}$, resulting in $\mathcal{P}' = \mathcal{P} \cup \{S'\}$ (line 1). Subsequently, the quality-and-distance score for each solution is computed using Equation (13) (lines 2-4). Lastly, the solution S_w with the lowest score is identified and eliminated from the population (lines 5-7).

4. Computational results

This section evaluates the performance of the proposed TMS algorithm and provides a comparative analysis with the leading heuristic algorithms designed for SOP.

4.1. Benchmark and experimental settings

Original benchmark instances—Set A. Archetti et al. (2018) constructed a benchmark set containing 612 SOP instances. These instances were derived by modifying a collection of 51 GTSP benchmark instances originally described in Fischetti et al. (1997), which range from 52 to 1084 nodes. Specifically, the initial node of each instance was relocated from its original cluster to create a single depot. Subsequently, the profit p_i allocated to each cluster $\mathcal{C}_i$ and the time constraint T_{max} were determined as follows:

(1) Profits: The cluster's profit results from aggregating the profits of its customers. Each customer's profit follows one of two patterns. Under the initial pattern, every customer's profit is fixed at 1. In contrast, the second pattern generates profits using the equation $1 + (7141 \times i + 73) \mod 100$, where i denotes the index of the customer node. This scheme serves to obtain pseudo-random profits and maintain variability in instance difficulty for robust algorithm evaluation.

(2) T_{max}: $T_{max} = \omega \times BKS_G$, where BKS_G is the best-known GTSP solution from Fischetti et al. (1997), and ω takes the values of 0.4, 0.6, 0.8 and 1.0, respectively. In Archetti et al. (2018), solutions with $\omega = 1.0$ are expected to collect profits from all clusters. However, this expectation is not feasible in SOP instances since the newly introduced depot cluster inherently increases the travel cost compared to the original GTSP solution.

The naming convention follows the pattern <clusterNum>,<name>,<nodeNum>,<threshold>,<profit> for all benchmark instances, where <clusterNum> denotes the number of clusters, <name> indicates the original GTSP instance, <nodeNum> specifies the total number of nodes, <threshold> represents the time budget parameter ω (with T40, T60, T80, and T100 corresponding to ω values of 0.4, 0.6, 0.8, and 1.0 respectively), and <profit> identifies the profit pattern (p1 and p2 indicating the first and second patterns). For example, the instance "53gil262T40p1" consists of 53 clusters derived from the original "gil262" instance containing 262 nodes, with a time budget factor of 0.4 and using the first profit pattern.

These SOP instances are categorized into two groups: small size ($|\mathcal{V}| < 200$) and large size ($|\mathcal{V}| \geq 200$). Following Lu et al. (2024), we only provide comparative results for the large instance group, which consists of 192 large instances, as negligible distinctions emerged within the small instance group.

New large benchmark instances—Set B. Building on the instance generation methodology outlined by Archetti et al. (2018), Dontas et al. (2023) created a substantial dataset consisting of 304 complex SOP instances, featuring up to 3162 customers and 634 clusters. These instances were derived from 19 large-scale GTSP instances, originally introduced by Helsgaun (2015), with node counts ranging from 1000 to 3162. The construction of this dataset took into account four different values of ω (0.4, 0.6, 0.8, 1.0), two configurations for profit, and two clustering setups for each of the 19 GTSP instances, resulting in 304 SOP instances in total. The two clustering setups were: (1) retaining the original GTSP clustering, and (2) randomly assigning nodes to clusters.

The implemented TMS algorithm was coded in C++ and ran on a computer equipped with a 2.10 GHz Intel Core i7-12700 CPU and 16 GB of RAM. Details of the algorithm's parameters are outlined in Table 1. These parameters underwent tuning using the well-regarded iterated racing technique (F-race) (Birattari et al., 2010), which automatically selects necessary configurations from a finite pool. The tuning phase encompassed 15 randomly selected cases, allocated with a budget of 1000 executions. The recommended

Table 1: Parameter settings.

Parameter	Section	Description	Tested values	Final value
ρ	3.3	Similarity threshold for population initialization	{0.6, 0.7, 0.8, 0.9}	0.8
T	3.4.2	Max allowed iterations without improvement	{200, 250, 300, 350, 400}	300
ϵ	3.4.2	Frequency of using tour improvement	{10, 20, 30, 40, 50}	30
μ	3.4.3	Adaptive adjustment interval	{2, 5, 10, 15, 20}	5
φ	3.4.3	Adaptive penalty weight for strategic oscillation	$\{0.8, 1.0, 1.2\} * f(S)/t(S)$	1.0
β	3.5	Reward factor	{0.2, 0.3, 0.4, 0.5, 0.6}	0.5
γ	3.5	Penalization factor	{0.2, 0.3, 0.4, 0.5, 0.6}	0.5

parameter configuration from F-race is depicted in the "Final Value" column of Table 1, serving as the default setup for the proposed TMS algorithm. We also performed a sensitivity analysis on parameters ρ and φ across clustered and random instances (see Appendix C for full details). The results confirm that the algorithm's performance remains robust to different instance types.

4.2. Computational results and comparisons on Set A

To assess the TMS's performance, we conducted comparisons with five advanced SOP heuristics (Table 2) from the literature. Given the absence of available results for the ant colony system (ACS) (Golmankhaneh & Keshtkaran, 2020) on large SOP benchmarks, we excluded it from our comparative analysis.

Table 2: Reference algorithms for SOP.

Name	Reference	Experimental environment	Runs	Scaling factor
Matheuristic algorithm (MA)	Archetti et al. (2018)	Intel Core i7 2.80 GHz CPU with 16 GB of RAM	not specified	1.33
Variable neighborhood search (VNS)	Pěnička et al. (2019)	Intel Core i7 3.40 GHz CPU with 16 GB of RAM	20	1.62
Biased random-key genetic algorithm (BRKGA)	Carrabs (2021)	Intel Core i7 2.80 GHz CPU with 16 GB of RAM	10	1.33
Adaptive memory matheuristic (AMMH)	Dontas et al. (2023)	Intel Core i7-8700 3.20 GHz CPU with 16 GB RAM	5	1.52
Hybrid evolutionary algorithm (HEA)	Lu et al. (2024)	Intel Xeon(R) E5-2695 2.10 GHz CPU with 4 GB RAM	20	1.00
Tri-individual memetic search (TMS)	This work	Intel Core i7-12700 2.10 GHz CPU with 16 GB RAM	10	1.00

Table 2 presents details of the five SOP heuristics, encompassing their references, experimental conditions, number of executions per instance, and the resulting scaling factors. To ensure equitable comparisons in execution time, we adjusted the computational time of the five comparative algorithms documented in their respective studies by adopting the method outlined in Chen et al. (2016). This adjustment assumes a linear relationship between CPU speed and CPU frequency. It is important to note that this conversion serves purely as a guide, given that the computational time of each algorithm is influenced not only by hardware configurations but also by factors such as compilers, the programming proficiency of the developer, etc. Hence, our primary emphasis remains on solution quality.

Due to the probabilistic characteristics inherent in heuristic algorithms, we performed 10 separate runs for each problem instance, with each run terminating after 100 generations using our TMS algorithm. Tables A.1-A.4 in Appendix A depict a comparative analysis between the TMS and the five comparative algorithms across 192 large benchmark cases, varying ω from 0.4 to 1.0. The first column (f_{bk}) showcases the best objective value achieved among all compared algorithms. Subsequent columns present specific

metrics for each algorithm, including the best objective value (f_{best}), the percentage deviation (Dev.(%)) between its best objective value and the best results achieved by all compared algorithms (calculated as $100 \times (f_{bk} - f_{best})/f_{bk}$), the average objective value (f_{avg}), and the average computational time ($t(s)$) in seconds. Entries highlighted in bold indicate instances where our TMS algorithm finds a new best solution. Notably, Table A.4 excludes the average results from MA and the outcomes of BRKGA and AMMH on instances with $\omega = 1.0$, as these were not provided in Archetti et al. (2018); Carrabs (2021); Dontas et al. (2023). Furthermore, we incorporate results from the long runtime of HEA (Lu et al., 2024), known for superior performance, facilitating comprehensive performance comparisons.

Based on Tables A.1-A.4, it is evident that our TMS algorithm consistently surpasses other advanced algorithms in all scenarios. Specifically, TMS either matches or exceeds the previously best-known solutions across all 192 cases. Furthermore, it generated new lower bounds for 27 instances, demonstrating improved performance. Moreover, in terms of the average results, TMS consistently demonstrates superior performance, underscoring the stability and dependability of the proposed TMS algorithm.

We also display boxplots in Figure 4 to provide a visual comparison of the distribution and ranges of the best objective function values achieved by our TMS algorithm and the five compared algorithms. It is clear from Figure 4 that TMS exhibits a significantly smaller deviation from the best-known results, emphasizing the effectiveness of our TMS algorithm in comparison to recently developed SOP algorithms.

To offer a clearer overview of these comparative results, we present Table 3, summarizing the statistical results for the five algorithms assessed across 192 extensive instances with $\omega = 0.4$, 0.6, 0.8, and 1.0. For each of these algorithms, the table specifies three core indicators: the quantity of instances where the algorithm yields the best solution among all competing methods, the average deviation between its best objective value and the best results obtained by all comparative algorithms, and the average time spent on computation (in seconds).

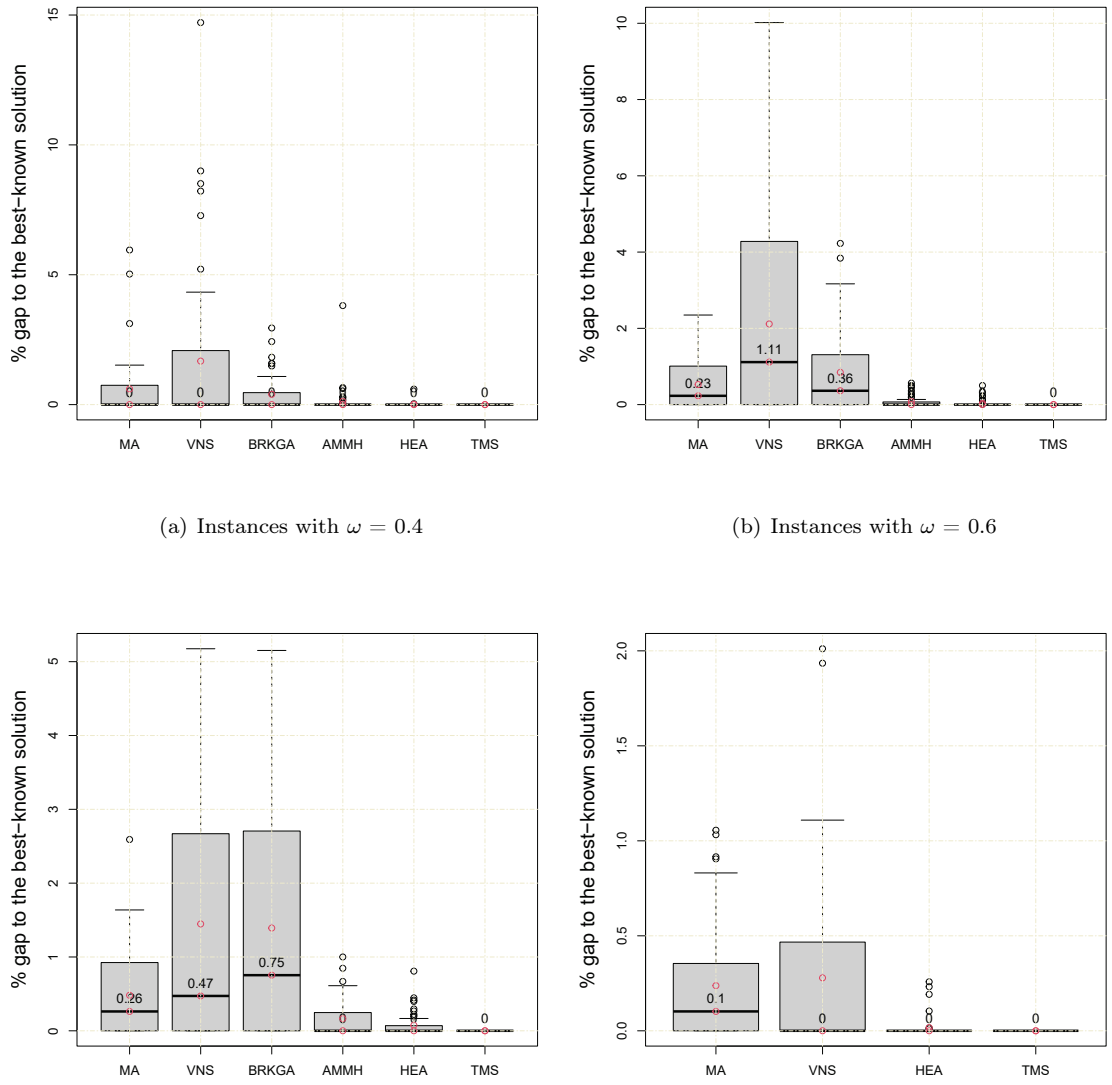
Table 3: Overview of the comparative results.

	MA			VNS			BRKGA			AMMH			HEA			TMS		
Instance	#Best	Dev.(%)	$Time(s)$	#Best	Dev.(%)	$Time(s)$	#Best	Dev.(%)	$Time(s)$	#Best	Dev.(%)	$Time(s)$	#Best	Dev.(%)	$Time(s)$	#Best	Dev.(%)	$Time(s)$
ω=0.4	25/48	0.602	220.1	26/48	1.674	56.9	28/48	0.396	32.6	41/48	0.123	116.6	46/48	0.013	157.9	48/48	0.000	89.4
ω=0.6	21/48	0.526	123.3	16/48	2.116	97.3	23/48	0.846	54.6	34/48	0.078	183.2	39/48	0.035	159.3	48/48	0.000	116.9
ω=0.8	21/48	0.474	93.0	18/48	1.448	144.7	17/48	1.385	75.6	27/48	0.165	233.8	35/48	0.075	159.3	48/48	0.000	98.4
ω=1.0	21/48	0.238	224.6	25/48	0.279	208.3	-	-	-	-	-	-	42/48	0.017	159.5	48/48	0.000	58.8

Based on Table 3, TMS shows outstanding superiority in solution quality across all cases. Specifically, TMS either achieves or exceeds the best results in every instance. Moreover, TMS displays the smallest deviation compared to the best objective value among all reference algorithms. These results clearly indicate that our proposed TMS algorithm competes exceptionally well with the reference algorithms for SOP.

4.3. Computational results and comparisons on Set B

Dontas et al. (2023) constructed a novel set of 304 challenging large instances for SOP that better reflect current research standards. To further assess the performance of TMS, we apply our proposed TMS



(a) Instances with $\omega = 0.4$

(b) Instances with $\omega = 0.6$

(c) Instances with $\omega = 0.8$

(d) Instances with $\omega = 1.0$

Figure 4: Boxplots of the best results for our TMS algorithm and the five comparative algorithm.

algorithm to these larger and more challenging SOP instances to validate its competitiveness. Similarly, each run terminates after 100 generations. Since these large-scale benchmark instances entail considerable computational effort, we further impose an additional time limit of 3600 seconds as a cutoff criterion. In practice, we observe that the algorithm generally converges to high-quality solutions long before reaching 100 generations for most instances. The time limit is adopted as an auxiliary stopping rule to restrict computational cost for large-scale instances.

Table 4: Overview of the comparative results on Set B.

	AMMH			TMS		
Instance	#Best	Dev.(%)	$Time(s)$	#Best	Dev.(%)	$Time(s)$
ω=0.4	33/76	0.446	429.1	47/76	0.567	1808.9
ω=0.6	45/76	0.304	386.4	43/76	0.338	1488.1
ω=0.8	45/76	0.264	209.9	69/76	0.063	962.3
ω=1.0	43/76	0.097	504.7	74/76	0.005	519.4

The statistical findings involving the TMS algorithm alongside the AMMH algorithm, as proposed by Dontas et al. (2023), are concisely outlined in Table 4 across the 304 new large instances with $\omega = 0.4, 0.6, 0.8$, and 1.0. This table indicates, for each algorithm, the number of instances where it delivers the best results compared to the other algorithm, the average deviation between its best objective value and the best results obtained from both the two algorithms, as well as the average runtime in seconds. Tables B.1-B.4 in Appendix B present a comparative results of the TMS and AMMH algorithms across the 304 new large-scale instances.

Based on these results, TMS demonstrates significant superiority in solution quality. Specifically, it outperforms the AMMH algorithm on 138 instances and achieves equivalent results on 95 instances. These results strongly suggest that our proposed TMS algorithm is highly competitive for SOP.

5. Analysis

To gain deeper insights into the crucial components of TMS, we conducted eight experimental analyses: (1) assessing the benefits of the small population, (2) validating the effectiveness of the tabu strategy with hashing techniques, (3) examining error rates in the hashing vectors, (4) evaluating the effectiveness of combining two local search phases, (5) analyzing the advantages of the IMR crossover operator, (6) investigating the superiority of the adaptive mutation strategy, (7) exploring the benefits of the high-quality initial population, and (8) identifying the most crucial TMS component. Our experiments were conducted on 28 benchmark instances with at least 400 vertices and $\omega = 0.8$.

5.1. Benefits of the small population

Our TMS algorithm employs a small population of merely three solutions. To explore the benefits of this limited population, we developed four variants of TMS: TMS$_{p1}$, TMS$_{p5}$, TMS$_{p10}$, and TMS$_{p20}$, consisting of 1, 5, 10, 20 solutions, respectively. In TMS$_{p1}$, which includes only one solution, the crossover operator is disabled. Instead, we employ a multi-start strategy combined with a randomized construction method (Section 3.3) until the allocated time budget is exhausted. We selected four representative instances to generate the convergence charts depicted in Figure 5.

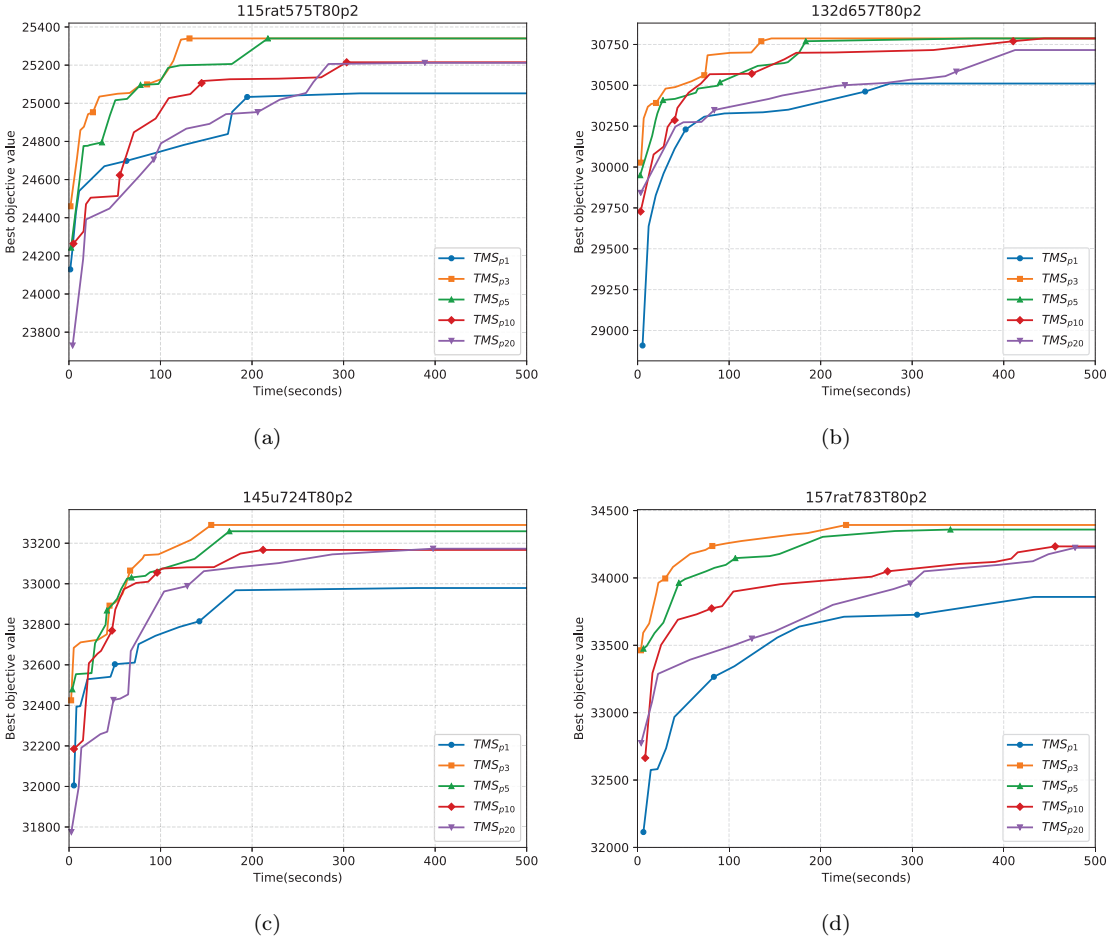
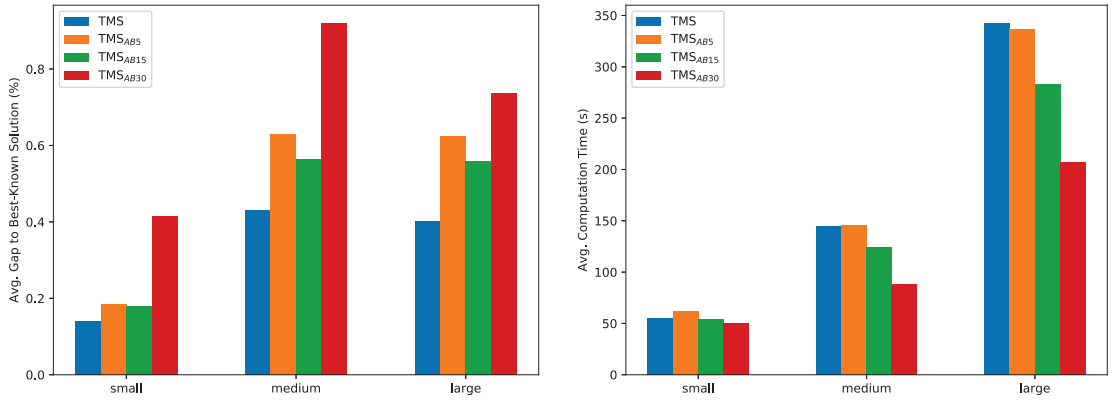


Figure 5: Convergence charts of TMS and four variants on four representative instances.

The convergence charts depicted in Figure 5 clearly illustrate the impressive effectiveness of the TMS algorithm when employing a small population. Over time, it becomes evident that TMS demonstrates significantly superior and faster convergence compared to other variants, thereby emphasizing the advantages of using a small population. Importantly, similar trends were observed across multiple experimental scenarios. The TMS_{p1} variant with a single-individual population adopts a multi-start approach, where solutions are randomly constructed at the beginning of each search round. However, due to the absence of crossover operations, it fails to inherit favorable characteristics from parent solutions, resulting in unsatisfactory solutions. In conclusion, the convergence charts provide compelling evidence for the significant contribution of a small population on enhancing TMS performance.

5.2. Effectiveness of the solution-based tabu strategy with hashing techniques

In Section 3.4.5, we employ the hashing techniques within the solution-based tabu search framework to efficiently monitor the access status of solutions. To illustrate the advantages of this method, we conduct a comparison between our TMS algorithm and three variants where the hashing techniques are replaced by the traditional attribute-based strategy (TMS_{AB}). In these variants, whenever a cluster is added to or removed from the current solution, that cluster becomes tabu for the subsequent tt iterations (tabu tenure), and we test three values ($tt = 5, 15, 30$) denoted as $\text{TMS}_{AB<tt>}$. For clear comparative analysis, we categorize the 28 benchmark instances into three groups based on cluster number k: small ($k < 100$), medium ($100 \leq k < 200$), and large ($200 \leq k$). The aggregated results are summarized in Figure 6.



(a) Average gap of different groups

(b) Average computation time on different groups

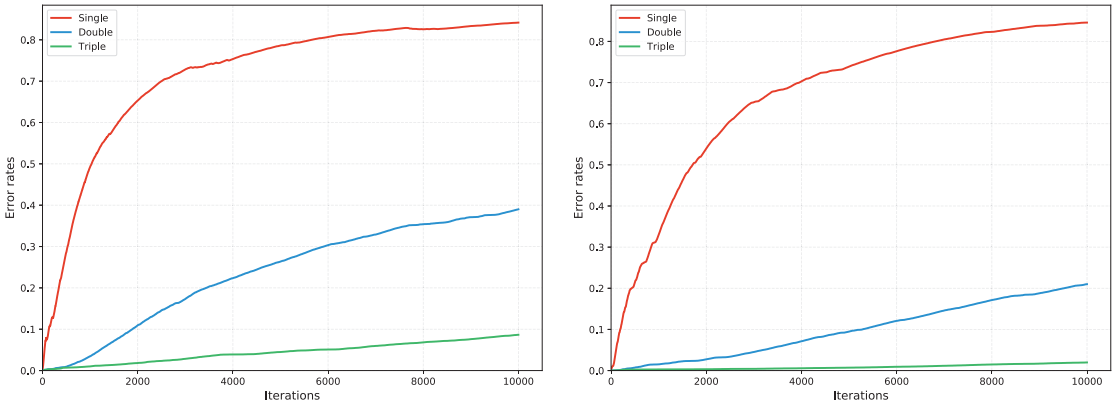
Figure 6: Comparison between TMS and ABTS with different tabu tenure.

Figure 6(a) presents the average gap to the best-known solution (calculated as $100\times(f_{bk}-f_{avg})/f_{bk}$, where f_{bk} represents the best objective value found among all algorithms), while Figure 6(b) shows the corresponding average computation time. The results clearly demonstrate that the superiority of our hashing-based strategy becomes increasingly pronounced with larger problem instances. Furthermore, TMS requires only marginally more computation time than the TMS_{AB} variants, with TMS_{AB5} exhibiting the longest runtime due to its finer-grained search characteristics. This pattern occurs because smaller tt values permit more frequent revisiting of clusters, leading to more exhaustive exploration and consequently longer computation times. The thorough findings validate that our tabu strategy is well-suited for determining the status of solutions in SOP instances, especially as problem scale increases.

5.3. Error rates of the hashing vectors

To quantitatively evaluate the effectiveness of our multi-hash approach in reducing collision errors, we conduct an experiment measuring the "error rate", defined as the probability that an unvisited solution

is incorrectly classified as tabu. Specifically, we execute the local search procedure for 10^4 iterations on two instances (80rd400T80p1 and 115rat575T80p2), and maintain a reference solution pool ($refPOP$) containing all encountered solutions. Throughout the process of searching, we monitor the evaluation of each neighborhood solution: whenever a solution is flagged as tabu by the hash mechanism, counter c_1 is incremented, and if this solution is simultaneously absent from $refPOP$ (indicating an error count), error counter c_2 is incremented. The error rate is then computed as $ER = c_2/c_1$.



(a) Error rates of instance 80rd400T80p1

(b) Error rates of instance 115rat575T80p2

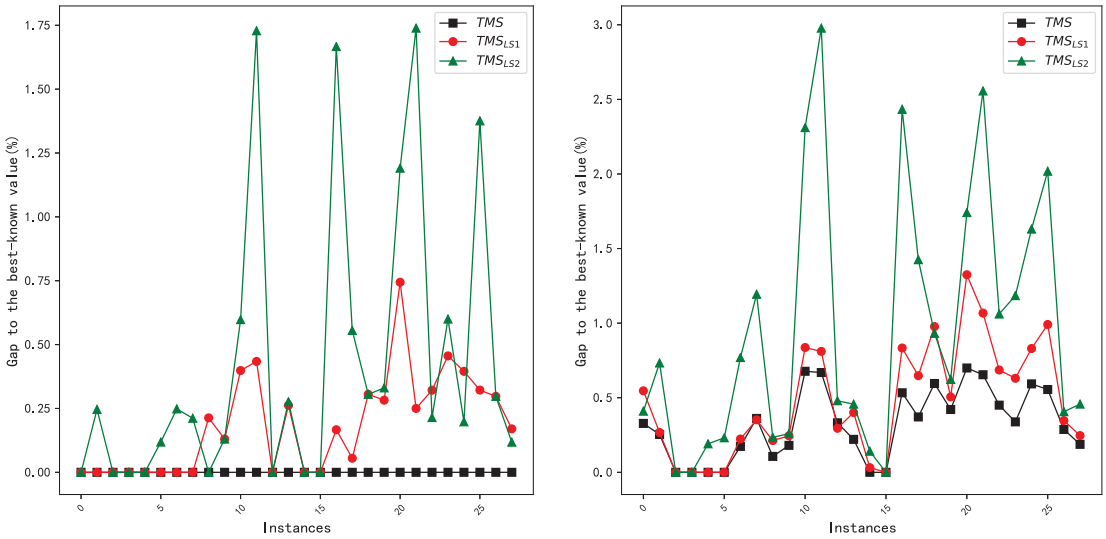
Figure 7: Effect of the number of hash vectors on error rate.

Figure 7 illustrates the evolution of error rates for different hash configurations throughout the search process. The results reveal critical performance differences: the single-hash configuration exhibits alarmingly high error rates approaching 0.9, indicating that approximately 90% of neighborhood solutions are incorrectly prohibited from visitation, which severely constrains exploration diversity. As expected, the double-hash configuration demonstrates lower error rates, and our three-hash system maintains near-zero error rates (< 0.08) throughout all iterations, demonstrating exceptional filtering accuracy.

This empirical evidence confirms that three hash vectors achieve the balance between computational efficiency and filtering precision, demonstrating the necessity of employing multiple hash functions for reliable tabu status determination in large-scale solution spaces.

5.4. Benefits of hybridizing two local search phases

The local search component of our TMS algorithm employs a two-phase local search approach. In the initial phase, a feasible local search is performed within the confines of the feasible solution space, whereas the subsequent phase employs a strategic oscillation search to investigate the infeasible solution space. To evaluate the effectiveness of this hybrid strategy, we propose two variants that substitute the complete local search process with: (1) solely the feasible local search from the first phase (TMS_{LS1}), and (2) exclusively the strategic oscillation search from the second phase (TMS_{LS2})).



(a) Gap of best results

(b) Gap of average results

Figure 8: Comparison of feasible and infeasible search space.

Figure 8 illustrates the comparative performance in terms of the best and average solution quality gap to the best-known benchmark. The results clearly show that TMS consistently outperforms both TMS$_{LS1}$ and TMS$_{LS2}$ across multiple instances, achieving superior average performance. These findings provide empirical evidence that the coordinated use of both search phases creates a synergistic effect, enhancing the overall search capability.

5.5. Advantage of the IMR crossover

The inherit-mutate-repair (IMR) crossover operator represents an innovative approach aimed at producing promising offspring solutions. To assess its impact, we introduced another variant named TMS$_{back}$, which substitutes the IMR crossover with a backbone-based crossover detailed in Lu et al. (2018). In this crossover operator, two offspring solutions o_1 and o_2 are generated from parent solutions S_1 and S_2 through a three-step procedure. First, common clusters shared by both parents are identified. Then, the vertices within these clusters are copied to each offspring while preserving their original order from the respective parent. Finally, the partial offspring are greedily completed by iteratively selecting the best feasible vertex from unvisited clusters, prioritizing those with the highest profit-to-effort ratio, and inserting it at the best position without exceeding the time budget.

Comparative outcomes between TMS and TMS$_{back}$ are depicted in Figure 9, illustrating the algorithms' best and average performances. As shown in Figure 9, solutions derived using the IMR crossover exhibit markedly superior fitness values compared to those generated by TMS$_{back}$ with the backbone-based crossover.

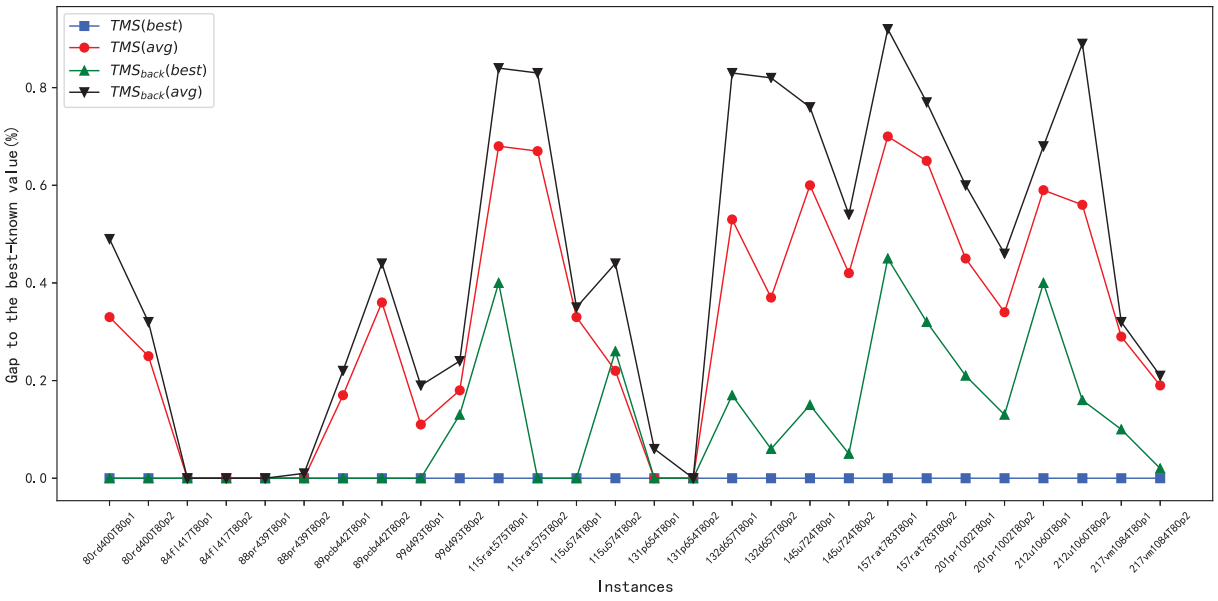


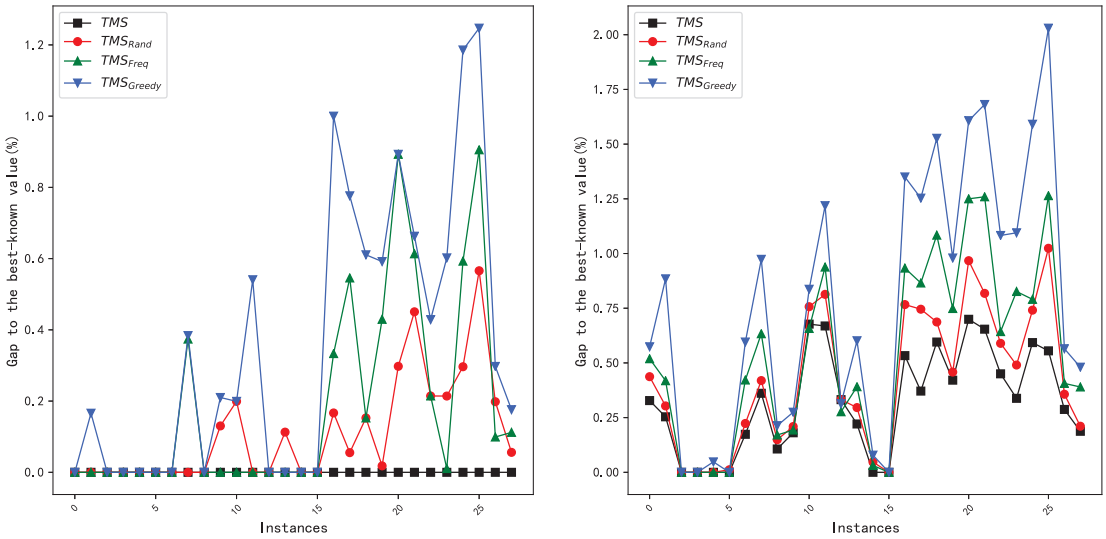
Figure 9: Comparisons of TMS with its variants TMS$_{back}$.

This underscores the IMR crossover's efficacy in effectively navigating the solution space to discover high-quality offspring solutions.

5.6. Superiority of the adaptive mutation

To assess the effectiveness of the adaptive mutation within the inherit-mutate-repair (IMR) crossover (Section 3.5), we compared TMS against three variants, each employing a distinct mutation strategy: (1) random mutation (TMS$_{Rand}$), which randomly removes half of the clusters; (2) frequency-based mutation (TMS$_{Freq}$), which eliminates the most frequently visited clusters (half of the total) to redirect the search toward unexplored regions; and (3) greedy-based mutation (TMS$_{Greedy}$), which uses profit-to-cost ratios to guide cluster selection by removing half of the clusters with the lowest profit-to-cost ratios, favoring high-value and low-cost insertions to enhance solution quality. To ensure a fair comparison, all compared algorithms employ the same greedy repair procedure detailed in Section 3.5 after the mutation step.

In Figure 10, the performance comparison between TMS and its variants is illustrated, highlighting the difference from the best-known solutions for both the best and average results in various instances. As demonstrated in the figure, while the random, frequency-based, and greedy-based approaches achieve reasonable performance, our adaptive mutation consistently yields superior solution quality for both the best and average results. The adaptive mechanism, though slightly more complex, provides an adaptive balance between intensification and diversification, which is particularly crucial for challenging instances. These results confirm that the adaptive mutation strategy using a probability vector outperforms simpler mutation strategies.



(a) Gap of best results

(b) Gap of average results

Figure 10: Comparison of different mutation strategies.

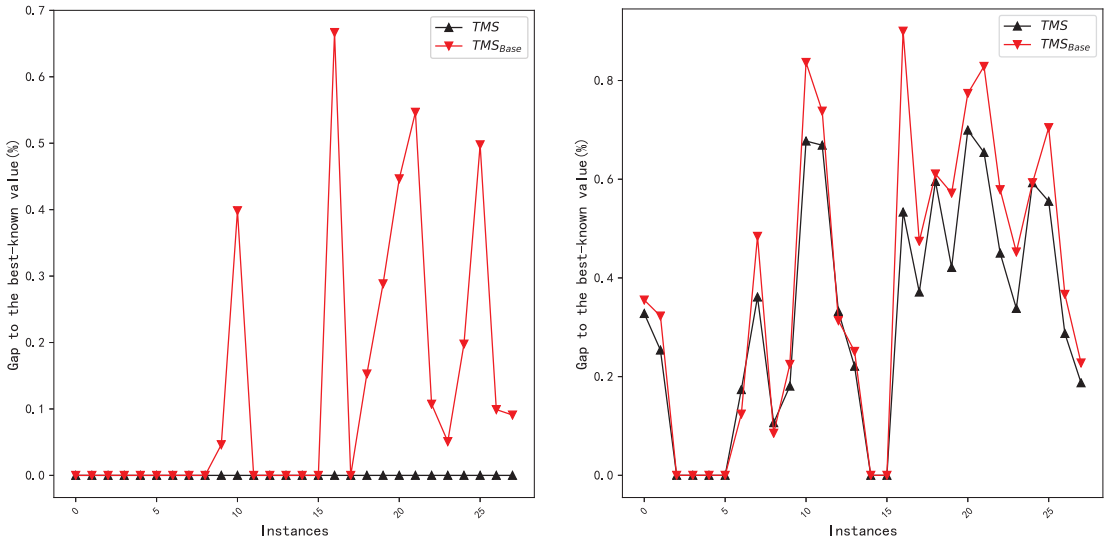
5.7. Benefit of the high-quality initial population

As described in Section 3.3, the initial population employed by the TMS algorithm is constructed using an initialization strategy supported by the GLKH solver, aiming to generate solutions with promising quality at the early stage of evolution. To quantitatively evaluate how such high-quality initial solutions affect the overall performance, we design a simplified variant named TMS_{Base}, in which the GLKH-based initialization is replaced by a basic greedy construction mechanism free of any external solvers.

The percentage gaps of the best and average results between the two algorithms are presented in Figure 11. The experimental results demonstrate that TMS_{Base} yields slightly worse best and average results than the original TMS. This observation indicates that the high-quality initial solutions generated by the GLKH solver do contribute to enhancing the algorithm's performance, as they provide a more favorable starting point for the subsequent search process. Nevertheless, even when initialized with simple and relatively weak solutions, TMS_{Base} still achieves competitive and reasonable performance. This demonstrates the effectiveness of the proposed memetic search framework of TMS, confirming that the algorithm's superiority stems primarily from its intrinsic search logic.

5.8. Identification of the most crucial TMS component

The proposed TMS algorithm integrates several key components that collectively contribute to its high performance, including: (1) GLKH-based population initialization process; (2) a local search with strategic oscillation, which adaptively alternates between exploiting feasible and infeasible regions; (3) a solution-based tabu strategy using hashing techniques; (4) an IMR crossover operator; and (5) an adaptive mutation



(a) Gap of best results

(b) Gap of average results

Figure 11: Comparison of different population initialization strategies.

procedure. To identify the most crucial component of TMS, we evaluate several variants, each excluding one specific element: "NoGLKH" (without GLKH initialization), "NoLocalSearch" (without local search), "NoTabuStrategy" (without the tabu strategy), "NoCrossover" (without the crossover operator), and "NoMutation" (without the mutation procedure).

Table 5: Comparison of the standard TMS and its components.

Set	Gap(%)	TMS (Standard)	NoGLKH	NoLocalSearch	NoTabuStrategy	NoCrossover	NoMutation
Clustered	BestGap (%)	0.000	0.128	1.175	0.342	1.172	0.660
	AvgGap (%)	0.321	0.386	1.803	0.749	1.809	1.206
Random	BestGap (%)	0.000	0.239	2.837	0.519	1.524	0.783
	AvgGap (%)	0.711	1.128	4.471	1.707	3.030	2.072

Table 5 summarizes the performance gaps of the five algorithm variants compared to the standard TMS, reporting the average gap of the best (BestGap) and average (AvgGap) results. The results show that removing the local search causes the greatest performance degradation (BestGap = 1.175, AvgGap = 1.803) on clustered instances, identifying it as the most crucial component. To further substantiate the robustness of these findings, we additionally evaluate all variants on 28 random instances. As shown in the lower panel of Table 5, the observed component importance ranking remains consistent across both instance types, with local search and crossover being the most critical. Despite their varying degrees of impact, the results collectively validate the contribution of each component, demonstrating the synergistic effect of their integration.

6. Conclusion

The SOP represents a significant extension of the classical OP, posing considerable computational challenges. In this study, we propose a novel tri-individual memetic search (TMS) algorithm, specifically designed to address SOP. Our approach integrates a feasible local search and a strategic oscillation with a solution-based tabu rule that dynamically explores both feasible and infeasible solutions. To diversify the search process, the algorithm incorporates an inherit-mutate-repair crossover operator for generating high-potential offspring solutions, alongside a tri-individual population updating strategy.

Through computational experiments and comparisons with the leading methods in the literature, we highlight the impressive performance of the proposed TMS algorithm on two comprehensive sets of SOP benchmark instances consisting of 496 large cases. In particular, among the 192 large original SOP instances, the TMS algorithm successfully identifies new lower bounds for 27 benchmark cases and achieves the best-known results for all remaining instances. Additionally, when tested on 304 new, larger, and more complex SOP instances, our TMS algorithm surpasses the previous best-known solutions for 138 cases. These newly established best-known solutions serve as important benchmarks for assessing upcoming SOP algorithms.

In future work, the proposed TMS algorithm could be further enhanced by exploring innovative strategies within the memetic framework. In addition, applying the proposed TMS algorithm to other relevant problems, such as SOP involving multiple vehicles, vehicle capacity, time-dependent profits, and dynamic real-time scenarios, would be intriguing avenues for future research.

Acknowledgment

We thank Dr. Eleftherios G. Manousakis for kindly sharing the new dataset introduced in Dontas et al. (2023). This work is partially supported by the National Natural Science Foundation of China [Grant No. 72371076, 72471100, 72122006, 72271240], the Hunan Provincial Natural Science Foundation General Project [Grant No.2022JJ30671], and "the Fundamental Research Funds for the Central Universities", Zhongnan University of Economics and Law (2722025BJ297).

Supplementary material

An Appendix detailing the computational results for all benchmark instances is available in the online supplementary material of this article.

References

Angelelli, E., Archetti, C., & Vindigni, M. (2014). The clustered orienteering problem. *European Journal of Operational Research*, *238*, 404–414.

Archetti, C., Carrabs, F., & Cerulli, R. (2018). The set orienteering problem. *European Journal of Operational Research*, *267*, 264–272.

Archetti, C., Carrabs, F., Cerulli, R., & Laureana, F. (2024). A new formulation and a branch-and-cut algorithm for the set orienteering problem. *European Journal of Operational Research*, *314*, 446–465.

Birattari, M., Yuan, Z., Balaprakash, P., & Stützle, T. (2010). F-race and iterated f-race: An overview. *Experimental methods for the analysis of optimization algorithms*, (pp. 311–336).

Carrabs, F. (2021). A biased random-key genetic algorithm for the set orienteering problem. *European Journal of Operational Research*, *292*, 830–854.

Chen, Y., Hao, J.-K., & Glover, F. (2016). A hybrid metaheuristic approach for the capacitated arc routing problem. *European Journal of Operational Research*, *253*, 25–39.

Derya, T., Dinler, E., & Keçeci, B. (2020). Selective generalized travelling salesman problem. *Mathematical and Computer Modelling of Dynamical Systems*, *26*, 80–118.

Ding, J., Lü, Z., Li, C.-M., Shen, L., Xu, L., & Glover, F. (2019). A two-individual based evolutionary algorithm for the flexible job shop scheduling problem. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 2262–2271). volume 33.

Dontas, M., Sideris, G., Manousakis, E. G., & Zachariadis, E. E. (2023). An adaptive memory matheuristic for the set orienteering problem. *European Journal of Operational Research*, *309*, 1010–1023.

Fischetti, M., Salazar González, J. J., & Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, *45*, 378–394.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision sciences*, *8*, 156–166.

Glover, F. W., & Hao, J. (2011). The case for strategic oscillation. *Annals of Operations Research*, *183*, 163–173.

Golmankhaneh, M. A., & Keshtkaran, M. (2020). Acs-sop: A new solution method to the set orienteering problem. In *2020 10th International Conference on Computer and Knowledge Engineering (ICCKE)* (pp. 644–647). IEEE.

Gunawan, A., Lau, H. C., & Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, *255*, 315–332.

Hao, J.-K. (2012). Memetic algorithms in discrete optimization. In *Handbook of memetic algorithms* (pp. 73–94). Springer.

He, M., Wu, Q., Benlic, U., Lu, Y., & Chen, Y. (2024). An effective multi-level memetic search with neighborhood reduction for the clustered team orienteering problem. *European Journal of Operational Research*, *318*, 778–801.

Helsgaun, K. (2000). An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, *126*, 106–130.

Helsgaun, K. (2015). Solving the equality generalized traveling salesman problem using the lin–kernighan–helsgaun algorithm. *Mathematical Programming Computation*, *7*, 269–287.

Lai, X., Yue, D., Hao, J.-K., & Glover, F. (2018). Solution-based tabu search for the maximum min-sum dispersion problem. *Information Sciences*, *441*, 79–94.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, *21*, 498–516.

Lu, Y., Benlic, U., & Wu, Q. (2018). A memetic algorithm for the orienteering problem with mandatory visits and exclusionary constraints. *European Journal of Operational Research*, *268*, 54–69.

Lu, Y., Benlic, U., & Wu, Q. (2024). An effective hybrid evolutionary algorithm for the set orienteering problem. *Information Sciences*, *654*, 119813.

Lü, Z., Glover, F., & Hao, J.-K. (2010). A hybrid metaheuristic approach to solving the ubqp problem. *European Journal of Operational Research*, *207*, 1254–1262.

Lu, Z., Martínez-Gavara, A., Hao, J.-K., & Lai, X. (2023). Solution-based tabu search for the capacitated dispersion problem. *Expert Systems with Applications*, *223*, 119856.

Moalic, L., & Gondran, A. (2018). Variations on memetic algorithms for graph coloring problems. *J. Heuristics*, *24*, 1–24.

Morán-Mirabal, L., González-Velarde, J., & Resende, M. G. (2014). Randomized heuristics for the family traveling salesperson problem. *International Transactions in Operational Research*, *21*, 41–57.

Narendra, K. S., & Thathachar, M. A. (2012). *Learning automata: an introduction*. Courier corporation.

Neri, F., & Cotta, C. (2012). Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, *2*, 1–14.

Noon, C. E., & Bean, J. C. (1993). An efficient transformation of the generalized traveling salesman problem. *INFOR: Information Systems and Operational Research*, *31*, 39–44.

Oliver, I., Smith, D., & Holland, J. R. (1987). Study of permutation crossover operators on the traveling salesman problem. In *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms: July 28-31, 1987, Massachusetts Institute of Technology, Cambridge, MA*.

Pěnička, R., Faigl, J., & Saska, M. (2019). Variable neighborhood search for the set orienteering problem and its application to other orienteering problem variants. *European Journal of Operational Research*, *276*, 816–825.

Sobti, R., & Geetha, G. (2012). Cryptographic hash functions: a review. *International Journal of Computer Science Issues (IJCSI)*, *9*, 461.

Tsiligirides, T. (1984). Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, *35*, 797–809.

Vansteenwegen, P., Souffriau, W., & Van Oudheusden, D. (2011). The orienteering problem: A survey.

European Journal of Operational Research, *209*, 1–10.

Woodruff, D. L., & Zemel, E. (1993). Hashing vectors for tabu search. *Annals of Operations Research*, *41*, 123–137.

Wu, Q., He, M., Hao, J.-K., & Lu, Y. (2024). An effective hybrid evolutionary algorithm for the clustered orienteering problem. *European Journal of Operational Research*, *313*, 418–434.

Zhou, Y., Wang, G., Hao, J.-K., Geng, N., & Jiang, Z. (2023). A fast tri-individual memetic search approach for the distance-based critical node problem. *European Journal of Operational Research*, *308*, 540–554.

Zou, Y., Hao, J.-K., & Wu, Q. (2025). A two-individual evolutionary algorithm for cumulative capacitated vehicle routing with single and multiple depots. *IEEE Transactions on Evolutionary Computation*, *29*, 505–518.

Online supplement

Appendix A.

Tables A.1-A.4 depict a comparative analysis between the TMS algorithm and the five state-of-the-art algorithms across 192 large benchmark cases, varying ω from 0.4 to 1.0. The first column (f_{bk}) showcases the best objective value achieved among all compared algorithms. Subsequent columns present specific metrics for each algorithm, including the best objective value (f_{best}), the percentage deviation (Dev.(%)) between its best objective value and the best results achieved by all compared algorithms (calculated as $100 \times (f_{bk} - f_{best})/f_{bk}$), the average objective value (f_{avg}), and the average computational time ($t(s)$) in seconds. Entries highlighted in bold indicate instances where our TMS algorithm finds a new best solution.

Table A.1: Computational results on instances with $\omega = 0.4$.

		MA				VNS				BRKGA				AMMH				HEA				TMS			
Instance	f_{bk}	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$
40kroa200T40p1	92	92	0.000	92.0	19.1	92	0.000	92.0	3.9	92	0.000	92.0	4.2	92	0.000	6.4	92	0.000	92.0	20.0	92	0.000	92.0	6.8
40kroa200T40p2	4550	4550	0.000	4550.0	21.0	4550	0.000	4550.0	4.1	4550	0.000	4550.0	4.1	4550	0.000	5.2	4550	0.000	4550.0	20.0	4550	0.000	4550.0	7.2
40krob200T40p1	87	87	0.000	87.0	14.4	87	0.000	87.0	4.0	87	0.000	87.0	4.1	87	0.000	8.1	87	0.000	87.0	20.0	87	0.000	87.0	8.1
40krob200T40p2	4348	4348	0.000	4348.0	16.8	4348	0.000	4348.0	4.1	4348	0.000	4348.0	4.0	4348	0.000	9.6	4348	0.000	4348.0	20.0	4348	0.000	4348.0	8.1
45ts225T40p1	101	101	0.000	101.0	15.7	101	0.000	101.0	5.6	101	0.000	101.0	4.2	101	0.000	7.3	101	0.000	101.0	25.3	101	0.000	101.0	0.0
45ts225T40p2	5037	5037	0.000	5037.0	24.2	5037	0.000	4945.0	6.2	5037	0.000	5037.0	4.7	5037	0.000	7.8	5037	0.000	5037.0	25.3	5037	0.000	5037.0	0.0
45tsp225T40p1	80	80	0.000	80.0	17.1	80	0.000	80.0	4.3	80	0.000	80.0	3.6	80	0.000	5.6	80	0.000	80.0	25.3	80	0.000	80.0	0.0
45tsp225T40p2	4297	4297	0.000	4297.0	21.5	4297	0.000	4297.0	4.2	4297	0.000	4297.0	3.6	4297	0.000	6.6	4297	0.000	4297.0	25.3	4297	0.000	4297.0	0.0
46pr226T40p1	86	86	0.000	85.4	20.1	86	0.000	86.0	3.8	86	0.000	86.0	3.8	86	0.000	6.7	86	0.000	86.0	26.0	86	0.000	86.0	0.0
46pr226T40p2	4403	4403	0.000	4394.6	19.7	4403	0.000	4403.0	3.6	4403	0.000	4403.0	3.8	4403	0.000	6.6	4403	0.000	4403.0	26.0	4403	0.000	4403.0	0.0
53gil262T40p1	105	105	0.000	104.8	25.9	105	0.000	104.2	7.4	105	0.000	104.7	6.7	105	0.000	8.1	105	0.000	105.0	34.7	105	0.000	105.0	0.0
53gil262T40p2	5330	5330	0.000	5330.0	27.9	5330	0.000	5284.9	7.7	5330	0.000	5330.0	6.7	5330	0.000	9.4	5330	0.000	5330.0	34.7	5330	0.000	5330.0	0.0
53pr264T40p1	128	128	0.000	128.0	48.7	128	0.000	128.0	6.9	127	0.781	127.0	4.7	128	0.000	5.0	128	0.000	128.0	35.0	128	0.000	128.0	0.0
53pr264T40p2	6423	6423	0.000	6423.0	51.2	6423	0.000	6423.0	7.1	6397	0.405	6383.5	4.7	6423	0.000	5.5	6423	0.000	6423.0	35.0	6423	0.000	6423.0	0.0
56a280T40p1	107	107	0.000	107.0	25.4	106	0.935	105.1	7.5	107	0.000	107.0	4.9	107	0.000	10.1	107	0.000	107.0	39.2	107	0.000	107.0	12.6
56a280T40p2	5630	5630	0.000	5611.3	30.9	5630	0.000	5605.0	8.4	5630	0.000	5606.3	5.3	5593	0.657	9.6	5630	0.000	5630.0	39.2	5630	0.000	5630.0	0.0
60pr299T40p1	131	131	0.000	131.0	38.9	131	0.000	130.6	9.0	131	0.000	131.0	5.4	131	0.000	8.7	131	0.000	131.0	44.9	131	0.000	131.0	15.8
60pr299T40p2	6600	6600	0.000	6600.0	42.5	6600	0.000	6600.0	9.2	6600	0.000	6600.0	5.7	6600	0.000	13.1	6600	0.000	6600.0	44.9	6600	0.000	6600.0	17.3
64lin318T40p1	169	169	0.000	169.0	38.5	169	0.000	168.8	12.4	169	0.000	169.0	10.0	169	0.000	15.8	169	0.000	169.0	50.9	169	0.000	169.0	19.8
64lin318T40p2	8600	8600	0.000	8600.0	47.8	8600	0.000	8593.1	9.6	8600	0.000	8600.0	9.6	8600	0.000	11.4	8600	0.000	8600.0	50.9	8600	0.000	8600.0	18.9
80rd400T40p1	209	208	0.478	206.5	86.7	209	0.000	208.2	21.9	208	0.478	207.6	15.2	209	0.000	27.7	209	0.000	209.0	80.0	209	0.000	209.0	26.3
80rd400T40p2	10971	10825	1.331	10752.3	79.5	10966	0.046	10858.8	22.5	10874	0.884	10816.6	17.0	10971	0.000	32.9	10971	0.000	10966.7	80.0	10971	0.000	10971.0	25.3
84fl417T40p1	201	201	0.000	175.5	64.1	201	0.000	201.0	19.3	201	0.000	200.6	14.8	201	0.000	18.4	201	0.000	201.0	87.6	201	0.000	201.0	37.8
84fl417T40p2	10163	10133	0.295	9375.4	62.3	10163	0.000	10119.6	19.9	10133	0.295	10123.4	16.6	10163	0.000	37.5	10163	0.000	10163.0	87.6	10163	0.000	10163.0	43.1
88pr439T40p1	335	334	0.299	333.9	96.1	333	0.597	332.4	45.3	334	0.299	329.0	27.5	335	0.000	36.7	335	0.000	334.4	96.6	335	0.000	334.9	39.0
88pr439T40p2	17082	17037	0.263	17013.5	161.5	17022	0.351	16804.6	41.5	17082	0.000	16944.6	30.5	17037	0.263	38.7	17082	0.000	17043.0	96.6	17082	0.000	17071.5	41.3
89pcb442T40p1	224	223	0.446	222.2	87.8	223	0.446	220.9	33.1	224	0.000	221.9	19.0	224	0.000	30.8	224	0.000	223.9	98.3	224	0.000	224.0	35.4
89pcb442T40p2	11627	11565	0.533	11546.3	106.7	11627	0.000	11502.1	29.7	11627	0.000	11587.6	19.9	11627	0.000	30.0	11627	0.000	11616.0	98.3	11627	0.000	11618.5	36.2
99d493T40p1	278	278	0.000	277.7	122.8	278	0.000	275.9	33.9	278	0.000	278.0	21.5	278	0.000	19.4	278	0.000	278.0	122.0	278	0.000	278.0	23.2
99d493T40p2	14131	14131	0.000	14083.3	111.0	13992	0.984	13992.0	33.5	14131	0.000	14131.0	20.3	14131	0.000	24.8	14131	0.000	14131.0	122.0	14131	0.000	14131.0	24.7
115rat575T40p1	264	262	0.758	259.5	255.3	264	0.000	258.9	48.7	264	0.000	260.9	27.4	264	0.000	103.6	264	0.000	262.7	165.3	264	0.000	262.8	77.7
115rat575T40p2	13367	13300	0.501	13203.0	395.5	13203	1.227	12950.8	54.7	13367	0.000	13293.0	29.2	13367	0.000	108.3	13367	0.000	13327.2	165.3	13367	0.000	13346.5	80.1
115u574T40p1	277	275	0.722	270.2	163.1	270	2.527	265.1	52.5	274	1.083	269.7	30.2	277	0.000	99.2	277	0.000	277.0	165.0	277	0.000	277.0	81.0
115u574T40p2	14094	13969	0.887	13739.6	165.4	13782	2.214	13407.2	49.8	14046	0.341	13823.3	31.1	14094	0.000	148.9	14094	0.000	14075.5	165.0	14094	0.000	14075.8	88.8
131p654T40p1	315	315	0.000	315.0	257.8	315	0.000	315.0	53.0	315	0.000	313.0	34.0	315	0.000	110.8	315	0.000	315.0	214.2	315	0.000	315.0	104.2
131p654T40p2	16003	16003	0.000	16003.0	347.2	15983	0.125	15958.9	47.4	16003	0.000	15902.5	34.2	16003	0.000	192.2	16003	0.000	16003.0	214.2	16003	0.000	16003.0	137.8
132d657T40p1	308	304	1.299	303.0	328.5	300	2.597	295.0	62.3	305	0.974	299.3	36.9	308	0.000	128.0	308	0.000	308.0	216.8	308	0.000	308.0	82.9
132d657T40p2	15974	15855	0.745	15790.2	415.2	15141	5.215	14633.2	62.4	15903	0.444	15738.4	36.7	15974	0.000	125.9	15974	0.000	15966.4	216.8	15974	0.000	15974.0	77.8
145u724T40p1	329	324	1.520	321.8	231.0	315	4.255	303.1	86.6	323	1.824	317.0	47.6	328	0.304	189.9	329	0.000	328.5	262.5	329	0.000	329.0	136.0
145u724T40p2	17368	17230	0.795	17127.2	288.1	16616	4.330	15998.6	92.0	17094	1.578	16974.3	48.1	17358	0.058	194.0	17368	0.000	17358.8	262.5	17368	0.000	17368.0	141.0
157rat783T40p1	350	348	0.571	344.9	300.6	346	1.143	331.2	107.9	349	0.286	342.8	57.2	350	0.000	261.3	350	0.000	347.9	307.3	350	0.000	349.2	181.7
157rat783T40p2	18244	18018	1.239	17720.8	420.3	17889	1.946	17088.8	98.6	18216	0.153	17838.8	59.0	18244	0.000	303.2	18244	0.000	18215.8	307.3	18244	0.000	18241.2	195.2
201pr1002T40p1	539	535	0.742	532.6	869.5	519	3.711	491.4	199.2	535	0.742	507.2	118.6	538	0.186	557.1	539	0.000	534.5	503.5	539	0.000	537.7	348.9
201pr1002T40p2	27628	26239	5.028	26086.2	1022.0	25357	8.220	24632.4	199.1	27189	1.589	26369.8	119.2	27628	0.000	411.0	27628	0.000	27236.7	503.5	27628	0.000	27573.6	344.8
212u1060T40p1	577	559	3.120	532.8	965.0	535	7.279	521.2	249.6	563	2.426	538.0	144.1	555	3.813	620.6	577	0.000	569.2	561.8	577	0.000	574.6	493.3
212u1060T40p2	29079	27348	5.953	26638.9	1028.9	26463	8.996	25952.5	242.6	28221	2.951	27458.3	147.5	28899	0.619	545.2	28905	0.598	28212.6	561.8	**29079**	0.000	28724.1	462.4
217vm1084T40p1	673	666	1.040	656.9	618.4	574	14.710	551.9	303.1	663	1.486	655.6	129.9	673	0.000	461.1	673	0.000	669.1	588.1	673	0.000	670.9	342.9
217vm1084T40p2	34400	34291	0.317	33818.5	947.7	31473	8.509	27875.1	289.8	34398	0.006	33799.6	126.8	34400	0.000	574.5	34388	0.035	34297.6	588.1	34400	0.000	34333.5	344.4
Average		6397.5	0.602	6331.8	220.1	6263.8	1.674	6093.3	56.9	6445.8	0.396	6370.4	32.6	6479.5	0.123	116.6	6481.8	0.013	6453.7	157.9	6485.7	0.000	6474.4	86.8

Table A.2: Computational results on instances with $\omega = 0.6$.

		MA				VNS				BRKGA				AMMH				HEA				TMS			
Instance	f_{bk}	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$
40kroa200T60p1	142	142	0.000	141.7	24.1	142	0.000	141.1	6.7	142	0.000	141.4	6.3	142	0.000	7.3	142	0.000	142.0	22.5	142	0.000	142.0	8.2
40kroa200T60p2	7105	7105	0.000	7104.6	23.1	7105	0.000	7062.4	6.5	7105	0.000	7104.6	6.4	7105	0.000	7.6	7105	0.000	7105.0	20.3	7105	0.000	7105.0	8.8
40krob200T60p1	138	138	0.000	137.3	20.2	138	0.000	137.1	8.0	138	0.000	137.4	6.7	138	0.000	7.2	138	0.000	138.0	22.3	138	0.000	138.0	9.8
40krob200T60p2	6943	6943	0.000	6943.0	19.4	6943	0.000	6884.8	8.0	6943	0.000	6940.0	6.8	6943	0.000	8.1	6943	0.000	6943.0	22.5	6943	0.000	6943.0	10.2
45ts225T60p1	152	152	0.000	151.1	8.5	152	0.000	151.0	10.4	152	0.000	151.2	6.8	152	0.000	14.8	152	0.000	152.0	27.1	152	0.000	152.0	11.6
45ts225T60p2	7948	7948	0.000	7896.7	11.9	7948	0.000	7853.5	9.7	7948	0.000	7918.2	7.0	7948	0.000	11.3	7948	0.000	7948.0	26.1	7948	0.000	7948.0	14.2
45tsp225T60p1	148	146	1.351	146.0	28.6	147	0.676	146.3	6.5	146	1.351	145.7	7.9	148	0.000	10.2	148	0.000	148.0	27.6	148	0.000	148.0	11.6
45tsp225T60p2	7603	7603	0.000	7603.0	25.4	7603	0.000	7584.0	6.6	7603	0.000	7603.0	7.3	7603	0.000	14.2	7603	0.000	7603.0	27.1	7603	0.000	7603.0	11.0
46pr226T60p1	162	162	0.000	162.0	13.6	162	0.000	161.4	7.6	162	0.000	162.0	6.9	162	0.000	6.6	162	0.000	162.0	28.2	162	0.000	162.0	10.6
46pr226T60p2	8280	8280	0.000	8280.0	16.8	8280	0.000	8233.0	6.4	8280	0.000	8280.0	7.0	8280	0.000	8.4	8280	0.000	8280.0	27.5	8280	0.000	8280.0	10.3
53gil262T60p1	177	176	0.565	176.0	23.4	176	0.565	173.3	11.5	177	0.000	175.9	10.7	177	0.000	12.5	177	0.000	177.0	36.7	177	0.000	177.0	18.7
53gil262T60p2	8864	8864	0.000	8854.6	46.4	8864	0.000	8862.4	11.5	8864	0.000	8863.4	10.9	8864	0.000	13.7	8864	0.000	8864.0	36.0	8864	0.000	8864.0	19.1
53pr264T60p1	162	162	0.000	161.5	39.3	162	0.000	160.4	9.9	162	0.000	161.4	9.4	162	0.000	10.1	162	0.000	162.0	36.5	162	0.000	162.0	18.4
53pr264T60p2	8404	8404	0.000	8404.0	40.1	8404	0.000	8397.0	9.7	8404	0.000	8404.0	8.8	8404	0.000	15.7	8404	0.000	8404.0	35.5	8404	0.000	8404.0	19.3
56a280T60p1	164	164	0.000	163.5	31.7	164	0.000	163.8	12.4	164	0.000	161.9	11.0	164	0.000	13.4	164	0.000	164.0	40.2	164	0.000	164.0	18.2
56a280T60p2	8470	8470	0.000	8468.1	34.4	8470	0.000	8330.7	12.9	8470	0.000	8468.1	10.7	8470	0.000	12.6	8470	0.000	8470.0	41.0	8470	0.000	8470.0	18.1
60pr299T60p1	187	187	0.000	187.0	34.0	177	5.348	175.0	12.5	187	0.000	187.0	11.1	187	0.000	15.1	187	0.000	187.0	45.4	187	0.000	187.0	20.4
60pr299T60p2	9544	9544	0.000	9544.0	31.6	9495	0.513	8939.9	14.1	9544	0.000	9496.3	11.0	9544	0.000	21.8	9544	0.000	9544.0	45.1	9544	0.000	9544.0	18.4
64lin318T60p1	256	256	0.000	255.1	29.3	256	0.000	249.4	21.1	256	0.000	255.1	16.9	256	0.000	30.8	256	0.000	256.0	52.4	256	0.000	256.0	19.2
64lin318T60p2	13188	13188	0.000	13188.0	28.3	13185	0.023	13038.3	20.0	13188	0.000	13188.0	16.6	13188	0.000	23.6	13188	0.000	13188.0	53.4	13188	0.000	13188.0	22.5
80rd400T60p1	298	291	2.349	288.5	54.7	294	1.342	287.5	33.1	295	1.007	286.4	24.6	298	0.000	29.9	298	0.000	295.7	81.0	298	0.000	296.0	37.6
80rd400T60p2	15241	15051	1.247	14983.9	50.9	15147	0.617	15009.8	29.3	15118	0.807	14901.9	25.1	15211	0.197	41.3	15241	0.000	15199.7	80.8	15241	0.000	15204.5	41.2
84fl417T60p1	290	288	0.690	285.9	47.1	286	1.379	254.3	23.6	288	0.690	284.9	20.8	290	0.000	21.5	290	0.000	290.0	89.8	290	0.000	290.0	44.7
84fl417T60p2	15007	14937	0.466	14937.0	57.6	14812	1.299	12890.8	22.4	14937	0.466	14826.8	21.9	15007	0.000	29.6	15007	0.000	15007.0	90.1	15007	0.000	15007.0	37.0
88pr439T60p1	385	385	0.000	384.1	63.0	385	0.000	383.5	48.4	384	0.260	382.7	38.1	385	0.000	67.7	385	0.000	384.7	97.1	385	0.000	384.8	60.5
88pr439T60p2	19642	19640	0.010	19613.2	109.5	19601	0.209	19489.7	53.6	19642	0.000	19516.9	39.8	19642	0.000	82.4	19642	0.000	19623.2	98.1	19642	0.000	19624.3	50.1
89pcb442T60p1	327	325	0.612	322.8	77.7	326	0.306	314.1	45.4	323	1.223	319.0	29.3	327	0.000	58.1	327	0.000	326.9	98.6	327	0.000	327.0	50.6
89pcb442T60p2	16644	16523	0.727	16455.4	82.5	16341	1.820	16099.6	47.8	16644	0.000	16463.7	31.3	16601	0.258	62.5	16644	0.000	16581.4	100.6	16644	0.000	16596.5	54.3
99d493T60p1	407	406	0.246	404.1	110.4	402	1.229	399.1	90.6	405	0.491	399.6	48.2	407	0.000	61.6	407	0.000	406.9	122.3	407	0.000	406.8	48.0
99d493T60p2	20618	20574	0.213	20481.5	98.9	20348	1.310	20060.5	81.9	20376	1.174	20216.1	46.9	20618	0.000	75.6	20618	0.000	20615.8	122.8	20618	0.000	20604.4	50.1
115rat575T60p1	399	396	0.752	391.6	136.9	392	1.754	377.9	79.0	394	1.253	383.2	46.4	397	0.501	149.5	397	0.501	395.8	165.6	**399**	0.000	396.6	109.7
115rat575T60p2	20175	19962	1.056	19817.6	137.9	19811	1.804	18977.7	84.7	19933	1.200	19530.3	48.4	20156	0.094	181.2	20154	0.104	20102.1	167.1	**20175**	0.000	20141.2	110.3
115u574T60p1	448	442	1.339	441.0	102.7	435	2.902	427.2	93.3	440	1.786	434.1	54.6	446	0.446	147.7	447	0.223	446.3	167.3	**448**	0.000	446.4	102.7
115u574T60p2	22812	22606	0.903	22149.3	100.7	21688	4.927	20976.8	85.0	22542	1.184	22032.8	54.1	22703	0.478	192.8	22812	0.000	22782.5	166.0	22812	0.000	22775.3	100.5
131p654T60p1	529	529	0.000	528.9	158.3	476	10.019	470.4	53.6	529	0.000	518.1	64.3	529	0.000	131.4	529	0.000	529.0	214.7	529	0.000	529.0	79.3
131p654T60p2	26905	26905	0.000	26904.4	179.4	26636	1.000	24064.8	66.1	26905	0.000	25828.1	62.6	26905	0.000	131.7	26905	0.000	26905.0	214.4	26905	0.000	26905.0	73.9
132d657T60p1	481	478	0.624	474.1	194.1	463	3.742	456.6	121.4	468	2.703	459.3	69.3	481	0.000	163.2	481	0.000	479.7	219.3	481	0.000	479.8	144.5
132d657T60p2	24520	24132	1.582	24013.7	195.7	23867	2.663	23225.9	121.8	24135	1.570	23644.3	67.1	24520	0.000	228.4	24520	0.000	24253.8	219.3	24520	0.000	24352.2	131.8
145u724T60p1	519	514	0.963	510.9	161.0	494	4.817	476.6	152.7	508	2.119	502.2	79.9	518	0.193	268.3	519	0.000	517.2	264.5	519	0.000	517.4	178.5
145u724T60p2	26905	26510	1.468	26302.3	231.8	25459	5.374	24562.6	148.1	26564	1.267	25945.0	80.2	26829	0.282	317.6	26905	0.000	26698.3	264.2	26905	0.000	26765.6	180.4
157rat783T60p1	531	524	1.318	517.4	343.3	515	3.013	495.4	174.6	521	1.883	503.5	92.9	528	0.565	310.6	531	0.000	526.4	308.6	531	0.000	528.1	258.8
157rat783T60p2	26928	26740	0.698	26516.8	280.5	25512	5.258	25076.3	184.3	26739	0.702	25794.7	94.1	26916	0.045	432.0	26928	0.000	26796.8	308.8	26928	0.000	26824.9	272.7
201pr1002T60p1	781	770	1.408	762.6	332.5	736	5.762	716.3	358.3	751	3.841	740.5	208.5	780	0.128	580.4	780	0.128	771.5	505.3	**781**	0.000	774.3	438.8
201pr1002T60p2	39590	39123	1.180	38537.7	308.9	37133	6.206	36207.1	352.2	38336	3.167	37611.0	207.3	39445	0.366	786.3	39463	0.321	39255.0	504.3	**39590**	0.000	39382.7	428.0
212u1060T60p1	836	827	1.077	821.8	380.1	790	5.502	753.6	407.5	812	2.871	790.9	246.8	836	0.000	973.9	835	0.120	831.3	563.3	836	0.000	829.8	625.7
212u1060T60p2	42090	41362	1.730	41042.7	578.9	38921	7.529	37879.5	464.4	40311	4.227	39577.9	246.3	42033	0.135	1030.2	42075	0.036	41710.1	562.1	**42090**	0.000	41744.5	620.4
217vm1084T60p1	875	871	0.457	867.3	358.7	822	6.057	779.0	547.3	860	1.714	834.8	192.3	875	0.000	982.7	873	0.229	872.5	589.8	875	0.000	872.8	484.3
217vm1084T60p2	44669	44573	0.215	44325.9	455.6	41730	6.580	40052.2	487.3	43938	1.636	42925.3	195.5	44651	0.040	988.8	44665	0.009	44543.6	588.8	**44669**	0.000	44574.8	497.1
Average		9452.5	0.526	9396.9	123.3	9204.1	2.116	8916.9	97.3	9398.6	0.846	9241.6	54.6	9507.7	0.078	183.2	9514.9	0.035	9483.0	159.3	9518.5	0.000	9492.1	116.9

Table A.3: Computational results on instances with $\omega = 0.8$.

		MA				VNS				BRKGA				AMMH			HEA				TMS			
Instance	f_{bk}	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$
40kroa200T80p1	184	184	0.000	184.0	17.7	184	0.000	183.7	6.9	184	0.000	184.0	8.2	184	0.000	6.2	184	0.000	184.0	21.8	184	0.000	184.0	8.7
40kroa200T80p2	9338	9338	0.000	9338.0	20.6	9338	0.000	9331.8	7.4	9338	0.000	9338.0	8.5	9338	0.000	7.9	9338	0.000	9338.0	21.8	9338	0.000	9338.0	9.5
40krob200T80p1	179	179	0.000	178.1	22.3	179	0.000	177.1	9.9	179	0.000	177.3	8.5	179	0.000	8.4	179	0.000	179.0	20.3	179	0.000	179.0	11.1
40krob200T80p2	9077	9077	0.000	9077.0	20.2	9077	0.000	8876.9	7.2	9077	0.000	9066.4	8.0	9077	0.000	7.5	9077	0.000	9077.0	21.5	9077	0.000	9077.0	11.2
45ts225T80p1	193	188	2.591	187.1	6.4	192	0.518	188.5	12.3	193	0.000	188.5	8.6	192	0.518	18.0	193	0.000	192.0	27.6	193	0.000	192.1	15.3
45ts225T80p2	9888	9888	0.000	9752.4	9.0	9888	0.000	9785.2	13.5	9888	0.000	9748.4	9.0	9888	0.000	16.9	9888	0.000	9888.0	27.3	9888	0.000	9888.0	17.6
45tsp225T80p1	198	196	1.010	195.9	25.4	198	0.000	197.9	10.0	196	1.010	195.8	10.9	198	0.000	17.2	198	0.000	198.0	26.3	198	0.000	198.0	16.6
45tsp225T80p2	10030	9934	0.957	9921.4	27.2	10030	0.000	10024.4	9.3	9934	0.957	9927.4	11.5	10030	0.000	9.3	10030	0.000	10030.0	26.1	10030	0.000	10030.0	15.8
46pr226T80p1	213	213	0.000	213.0	17.1	213	0.000	213.0	6.8	213	0.000	213.0	10.2	213	0.000	6.9	213	0.000	213.0	28.0	213	0.000	213.0	15.0
46pr226T80p2	10770	10770	0.000	10770.0	13.4	10770	0.000	10734.6	7.4	10770	0.000	10770.0	10.2	10770	0.000	8.5	10770	0.000	10770.0	27.7	10770	0.000	10770.0	16.9
53gil262T80p1	226	226	0.000	225.0	23.2	226	0.000	225.0	14.3	226	0.000	224.5	14.4	226	0.000	16.5	226	0.000	226.0	36.2	226	0.000	226.0	21.3
53gil262T80p2	11606	11606	0.000	11586.6	27.6	11606	0.000	11439.0	15.2	11606	0.000	11586.6	15.5	11606	0.000	22.2	11606	0.000	11606.0	35.0	11606	0.000	11606.0	21.5
53pr264T80p1	238	238	0.000	238.0	31.6	233	2.101	233.0	7.9	238	0.000	237.5	15.9	238	0.000	16.2	238	0.000	238.0	36.0	238	0.000	238.0	26.3
53pr264T80p2	12048	12048	0.000	12048.0	41.3	12048	0.000	11824.5	8.7	12048	0.000	12022.7	15.2	12048	0.000	16.3	12048	0.000	12048.0	37.5	12048	0.000	12048.0	25.8
56a280T80p1	232	232	0.000	231.8	30.7	232	0.000	228.7	16.4	232	0.000	231.3	16.6	232	0.000	16.3	232	0.000	232.0	40.0	232	0.000	232.0	19.9
56a280T80p2	11984	11984	0.000	11898.5	28.8	11984	0.000	11911.4	16.9	11952	0.267	11925.6	16.7	11952	0.267	17.4	11984	0.000	11959.8	40.0	11984	0.000	11964.8	21.2
60pr299T80p1	270	270	0.000	269.5	35.1	269	0.370	267.3	21.1	270	0.000	269.2	19.3	270	0.000	16.3	270	0.000	270.0	45.1	270	0.000	270.0	23.9
60pr299T80p2	13653	13653	0.000	13619.2	40.1	13653	0.000	13533.8	21.8	13653	0.000	13649.5	19.0	13653	0.000	18.6	13653	0.000	13653.0	47.1	13653	0.000	13653.0	21.4
64lin318T80p1	295	295	0.000	294.1	29.6	295	0.000	291.5	23.7	295	0.000	291.8	22.6	295	0.000	20.1	295	0.000	295.0	52.6	295	0.000	295.0	27.8
64lin318T80p2	15103	15103	0.000	15055.9	43.4	15103	0.000	14981.5	25.5	15103	0.000	15012.9	21.6	15103	0.000	34.4	15103	0.000	15100.3	52.6	15103	0.000	15095.1	24.0
80rd400T80p1	366	363	0.820	360.7	49.6	359	1.913	354.3	48.8	356	2.732	350.6	32.9	366	0.000	40.4	366	0.000	365.4	81.3	366	0.000	364.8	37.1
80rd400T80p2	18681	18529	0.814	18438.3	54.2	18239	2.366	18040.7	50.0	18519	0.867	18225.9	33.6	18656	0.134	60.8	18681	0.000	18629.9	80.5	18681	0.000	18633.6	39.3
84fl417T80p1	399	399	0.000	398.8	89.8	399	0.000	395.1	27.8	398	0.251	397.4	32.5	399	0.000	63.4	399	0.000	399.0	89.1	399	0.000	399.0	47.7
84fl417T80p2	20258	20248	0.049	20245.2	72.6	20220	0.188	19885.6	29.3	20220	0.188	20152.2	35.3	20258	0.000	59.3	20258	0.000	20258.0	87.8	20258	0.000	20258.0	49.6
88pr439T80p1	419	419	0.000	418.9	107.0	415	0.955	412.5	50.8	414	1.193	412.1	44.9	419	0.000	62.0	419	0.000	419.0	96.8	419	0.000	419.0	70.7
88pr439T80p2	21134	21134	0.000	21094.3	103.1	21068	0.312	21017.6	58.4	21068	0.312	20951.4	46.0	21134	0.000	80.2	21134	0.000	21131.5	98.3	21134	0.000	21134.0	66.4
89pcb442T80p1	403	401	0.496	399.2	71.7	394	2.233	385.4	59.3	399	0.993	389.5	40.1	403	0.000	48.6	403	0.000	402.4	99.6	403	0.000	402.3	50.6
89pcb442T80p2	20319	20214	0.517	20159.4	74.5	20099	1.083	19693.5	56.4	20120	0.979	19688.9	39.8	20258	0.300	72.2	20319	0.000	20242.3	99.3	20319	0.000	20245.6	53.6
99d493T80p1	469	468	0.213	467.0	58.7	467	0.426	458.6	99.5	466	0.640	462.1	61.0	469	0.000	85.3	469	0.000	468.4	124.0	469	0.000	468.5	67.8
99d493T80p2	23800	23726	0.311	23645.9	69.3	23646	0.647	23233.3	105.2	23511	1.214	23361.6	65.6	23746	0.227	107.1	23789	0.046	23750.2	124.0	**23800**	0.000	23757.0	71.7
115rat575T80p1	502	495	1.394	493.0	105.8	487	2.988	478.1	107.4	486	3.187	478.2	63.9	500	0.398	185.4	500	0.398	498.2	166.8	**502**	0.000	498.6	93.9
115rat575T80p2	25340	25072	1.058	24891.3	126.8	24875	1.835	24434.6	101.6	24779	2.214	24498.0	65.2	25340	0.000	196.3	25340	0.000	25108.2	167.1	25340	0.000	25170.5	105.4
115u574T80p1	542	539	0.554	537.4	116.3	533	1.661	527.2	137.1	532	1.845	521.5	74.2	541	0.185	159.1	542	0.000	541.1	165.5	542	0.000	540.2	79.8
115u574T80p2	27475	27389	0.313	27293.9	150.2	26828	2.355	26308.3	130.1	27077	1.449	26586.6	74.9	27420	0.200	172.5	27475	0.000	27402.8	167.0	27475	0.000	27414.3	95.5
131p654T80p1	637	636	0.157	636.0	127.1	636	0.157	632.0	60.8	636	0.157	633.0	105.3	636	0.157	200.8	637	0.000	636.9	215.4	637	0.000	637.0	117.0
131p654T80p2	32335	32335	0.000	32331.6	143.2	32216	0.368	32030.2	66.4	32227	0.334	32074.9	106.2	32335	0.000	276.7	32335	0.000	32335.0	216.4	32335	0.000	32335.0	145.1
132d657T80p1	600	592	1.333	589.3	129.8	575	4.167	561.1	163.9	575	4.167	565.1	97.3	594	1.000	269.4	599	0.167	594.8	218.8	**600**	0.000	596.8	161.4
132d657T80p2	30787	30497	0.942	30372.7	147.9	30123	2.157	29311.8	147.3	29962	2.680	29437.6	97.2	30581	0.669	322.6	30659	0.416	30563.5	219.1	**30787**	0.000	30672.7	125.1
145u724T80p1	655	645	1.527	639.4	113.3	626	4.427	603.1	227.4	634	3.206	621.1	114.7	651	0.611	307.5	654	0.153	649.5	263.0	**655**	0.000	651.1	158.8
145u724T80p2	33290	32935	1.066	32768.9	155.6	31863	4.287	31633.2	220.7	32357	2.803	31830.0	111.7	33215	0.225	409.9	33216	0.222	33095.7	263.7	**33290**	0.000	33149.8	160.9
157rat783T80p1	672	661	1.490	653.8	199.4	652	2.976	643.3	251.8	641	4.471	632.1	134.2	668	0.595	377.3	669	0.446	663.2	307.8	**672**	0.000	667.3	233.7
157rat783T80p2	34393	34081	0.907	33731.1	226.7	32613	5.175	31793.5	270.5	32972	4.132	32383.5	137.0	34250	0.416	478.2	34301	0.267	33918.1	309.3	**34393**	0.000	34168.0	262.7
201pr1002T80p1	933	928	0.536	925.1	245.5	896	3.966	884.4	537.7	909	2.572	874.5	285.3	930	0.322	1053.3	932	0.107	926.7	504.0	**933**	0.000	928.8	307.6
201pr1002T80p2	47170	46800	0.778	46719.9	195.7	45394	3.765	44801.6	561.1	45404	3.738	44720.2	288.5	47064	0.225	1026.6	47126	0.093	46898.8	504.8	**47170**	0.000	47010.5	306.2
212u1060T80p1	1012	1001	0.891	994.5	304.3	967	4.447	958.3	635.5	960	4.950	945.0	330.2	1007	0.494	1210.8	1009	0.296	1004.3	564.3	**1012**	0.000	1006.0	336.7
212u1060T80p2	51242	50704	1.050	50152.4	310.8	48879	4.611	47680.8	754.7	48602	5.152	47784.1	330.7	50808	0.847	1061.0	50828	0.808	50605.8	563.1	**51242**	0.000	50957.5	353.1
217vm1084T80p1	1009	1004	0.496	1001.8	158.9	979	2.973	968.4	918.3	979	2.973	965.3	260.8	1008	0.099	1222.4	1007	0.198	1005.3	588.8	**1009**	0.000	1006.1	394.3
217vm1084T80p2	51735	51485	0.483	51323.3	247.8	49635	4.059	48736.8	806.7	49233	4.836	48523.5	249.3	51713	0.043	1310.2	51735	0.000	51578.8	590.6	51735	0.000	51638.0	359.4
Average		11652.5	0.474	11603.5	93.0	11454.2	1.448	11281.5	144.7	11459.0	1.385	11327.6	75.6	11688.8	0.165	233.8	11698.5	0.075	11662.3	159.3	11714.6	0.000	11683.9	98.4

Table A.4: Computational results on instances with $\omega = 1.0$.

		MA			VNS				HEA				TMS			
Instance	f_{bk}	f_{best}	Dev.(%)	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$
40kroa200T100p1	198	198	0.000	30.4	198	0.000	198.0	6.3	198	0.000	198.0	21.8	198	0.000	198.0	12.5
40kroa200T100p2	10010	10010	0.000	32.7	10010	0.000	9976.0	6.2	10010	0.000	10010.0	20.8	10010	0.000	10010.0	13.9
40krob200T100p1	198	198	0.000	26.5	198	0.000	198.0	8.6	198	0.000	198.0	21.5	198	0.000	198.0	10.2
40krob200T100p2	9990	9990	0.000	38.1	9990	0.000	9982.7	9.9	9990	0.000	9990.0	20.5	9990	0.000	9990.0	7.8
45ts225T100p1	221	221	0.000	46.3	221	0.000	220.8	12.5	221	0.000	221.0	27.1	221	0.000	221.0	13.4
45ts225T100p2	11225	11187	0.339	34.9	11225	0.000	11158.4	13.4	11225	0.000	11225.0	27.8	11225	0.000	11225.0	22.2
45tsp225T100p1	221	219	0.905	22.3	220	0.452	219.1	10.5	221	0.000	220.8	27.1	221	0.000	220.6	13.4
45tsp225T100p2	11124	11103	0.189	21.9	11124	0.000	11063.9	12.2	11124	0.000	11123.9	27.1	11124	0.000	11124.0	13.4
46pr226T100p1	224	224	0.000	35.9	224	0.000	224.0	6.2	224	0.000	224.0	27.7	224	0.000	224.0	14.9
46pr226T100p2	11368	11368	0.000	35.2	11368	0.000	11358.1	7.9	11368	0.000	11368.0	27.7	11368	0.000	11368.0	15.3
53gil262T100p1	258	258	0.000	36.3	258	0.000	254.2	14.2	258	0.000	258.0	36.7	258	0.000	258.0	17.4
53gil262T100p2	13050	13050	0.000	34.5	13050	0.000	13003.6	13.4	13050	0.000	13050.0	36.5	13050	0.000	13050.0	19.6
53pr264T100p1	262	262	0.000	45.3	262	0.000	262.0	12.2	262	0.000	262.0	36.0	262	0.000	262.0	21.0
53pr264T100p2	13277	13277	0.000	49.2	13277	0.000	13277.0	12.7	13277	0.000	13277.0	35.5	13277	0.000	13277.0	19.9
56a280T100p1	273	273	0.000	44.7	273	0.000	270.8	17.8	273	0.000	272.4	40.7	273	0.000	272.7	17.5
56a280T100p2	13971	13971	0.000	49.3	13971	0.000	13834.9	17.0	13971	0.000	13960.1	40.2	13971	0.000	13964.2	18.6
60pr299T100p1	296	296	0.000	41.7	296	0.000	295.0	20.6	296	0.000	296.0	47.4	296	0.000	296.0	34.7
60pr299T100p2	15036	15005	0.206	49.1	15005	0.206	14974.4	20.7	15036	0.000	15009.7	46.6	15036	0.000	15011.2	38.5
64lin318T100p1	316	315	0.316	57.9	316	0.000	313.8	18.2	316	0.000	316.0	53.1	316	0.000	316.0	29.3
64lin318T100p2	16013	16013	0.000	103.1	16013	0.000	15948.0	19.2	16013	0.000	16013.0	51.1	16013	0.000	16013.0	24.9
80rd400T100p1	398	397	0.251	102.3	398	0.000	394.3	48.3	398	0.000	397.6	81.8	398	0.000	397.7	30.6
80rd400T100p2	20140	20055	0.422	64.1	20140	0.000	19942.2	50.4	20140	0.000	20111.2	80.3	20140	0.000	20113.6	34.3
84fl417T100p1	415	415	0.000	138.0	415	0.000	414.4	30.9	415	0.000	415.0	89.6	415	0.000	415.0	37.1
84fl417T100p2	21042	21030	0.057	152.9	21030	0.057	20956.0	31.0	21042	0.000	21042.0	89.8	21042	0.000	21042.0	44.9
88pr439T100p1	437	437	0.000	210.7	437	0.000	432.4	58.1	437	0.000	437.0	96.8	437	0.000	437.0	41.5
88pr439T100p2	22127	22110	0.077	177.1	22110	0.077	22032.2	56.7	22127	0.000	22113.4	98.6	22127	0.000	22113.4	47.1
89pcb442T100p1	440	440	0.000	172.7	440	0.000	438.2	66.2	440	0.000	440.0	99.8	440	0.000	440.0	39.6
89pcb442T100p2	22300	22300	0.000	126.7	22300	0.000	22116.3	62.1	22300	0.000	22300.0	100.6	22300	0.000	22300.0	42.0
99d493T100p1	491	490	0.204	161.1	490	0.204	487.1	115.4	491	0.000	490.7	124.0	491	0.000	490.6	59.3
99d493T100p2	24854	24827	0.109	204.1	24840	0.056	24708.3	113.7	24854	0.000	24849.9	124.3	24854	0.000	24850.4	61.4
115rat575T100p1	568	562	1.056	121.6	563	0.880	555.0	131.1	568	0.000	564.7	167.6	568	0.000	565.6	66.7
115rat575T100p2	28649	28497	0.531	87.9	28361	1.005	28043.5	129.6	28594	0.192	28491.1	167.8	**28649**	0.000	28545.7	69.7
115u574T100p1	572	571	0.175	272.7	571	0.175	569.9	137.7	572	0.000	571.5	167.3	572	0.000	571.5	59.2
115u574T100p2	28935	28888	0.162	283.6	28888	0.162	28866.5	123.8	28905	0.104	28889.7	167.0	**28935**	0.000	28904.7	58.8
131p654T100p1	652	652	0.000	474.7	652	0.000	650.6	78.7	652	0.000	652.0	216.4	652	0.000	652.0	75.2
131p654T100p2	32991	32991	0.000	480.1	32950	0.124	32894.9	75.8	32991	0.000	32991.0	214.4	32991	0.000	32991.0	141.4
132d657T100p1	655	649	0.916	168.1	649	0.916	642.1	186.7	655	0.000	653.1	217.6	655	0.000	653.6	79.3
132d657T100p2	33182	32974	0.627	207.9	33022	0.482	32901.6	171.9	33182	0.000	33115.8	219.3	33182	0.000	33121.1	81.6
145u724T100p1	722	716	0.831	132.8	717	0.693	708.6	302.7	722	0.000	719.6	263.7	722	0.000	719.7	95.7
145u724T100p2	36518	36288	0.630	155.6	36316	0.553	35964.8	293.8	36518	0.000	36422.3	263.7	36518	0.000	36424.9	126.3
157rat783T100p1	775	767	1.032	372.3	760	1.935	750.4	386.9	773	0.258	770.1	309.1	**775**	0.000	771.1	120.4
157rat783T100p2	39277	38953	0.825	194.0	38487	2.011	37999.8	399.6	39186	0.232	39029.0	309.1	**39277**	0.000	39057.7	115.4
201pr1002T100p1	1000	994	0.600	406.5	994	0.600	981.8	823.5	1000	0.000	999.0	505.5	1000	0.000	998.6	146.9
201pr1002T100p2	50570	50453	0.231	1323.6	50172	0.787	49760.9	916.6	50570	0.000	50550.0	504.5	50570	0.000	50550.3	130.0
212u1060T100p1	1058	1057	0.095	1164.7	1057	0.095	1056.3	1165.5	1058	0.000	1057.6	563.6	1058	0.000	1057.6	150.0
212u1060T100p2	53523	53450	0.136	1064.7	53437	0.161	53391.8	1100.2	53517	0.011	53517.0	564.1	**53523**	0.000	53517.6	145.6
217vm1084T100p1	1082	1078	0.370	652.7	1070	1.109	1059.7	1426.3	1082	0.000	1080.9	589.1	1082	0.000	1081.0	160.2
217vm1084T100p2	54708	54642	0.121	874.0	54363	0.631	53744.2	1257.8	54700	0.015	54691.3	588.8	**54708**	0.000	54695.2	181.7
Average		12585.9	0.238	224.6	12565.2	0.279	12489.5	208.3	12612.9	0.017	12601.2	159.5	12616.9	0.000	12603.7	58.8

Appendix B.

Tables B.1–B.4 present a comparative analysis of the TMS and AMMH algorithms across 304 new large-scale instances, with ω varying from 0.4 to 1.0.

Table B.1: Computational results on Set B instances with $\omega = 0.4$.

		AMMH				TMS						AMMH				TMS			
Instance	f_{bk}	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$	Instance	f_{bk}	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$
200E1k.0T40p1	534	530	0.749	525.0	34.4	**534**	0.000	533.8	282.6	200E1k.0RNDT40p1	875	865	1.143	849.0	59.7	**875**	0.000	858.1	332.7
200E1k.0T40p2	27539	27038	1.819	26813.0	58.5	**27539**	0.000	27367.8	306.5	200E1k.0RNDT40p2	44792	44378	0.924	42997.0	117.2	**44792**	0.000	43659.5	388.8
235pcb1173T40p1	604	599	0.828	588.0	34.0	**604**	0.000	599.4	535.0	235pcb1173RNDT40p1	991	991	0.000	971.0	110.8	974	1.715	951.2	697.4
235pcb1173T40p2	30730	30628	0.332	30237.0	110.0	**30730**	0.000	30300.4	540.8	235pcb1173RNDT40p2	50163	49845	0.634	48994.0	214.3	**50163**	0.000	49169.0	675.5
259d1291T40p1	589	576	2.207	560.0	35.0	**589**	0.000	585.5	594.0	259d1291RNDT40p1	1110	1110	0.000	1081.0	181.2	1088	1.982	1067.2	821.8
259d1291T40p2	29540	28974	1.916	28474.0	31.3	**29540**	0.000	29451.4	516.4	259d1291RNDT40p2	55179	55046	0.241	53671.0	91.4	**55179**	0.000	54001.9	831.2
261rl1304T40p1	697	679	2.582	666.0	78.9	**697**	0.000	693.1	802.9	261rl1304RNDT40p1	1254	1249	0.399	1232.0	112.0	**1254**	0.000	1246.1	695.9
261rl1304T40p2	35567	34809	2.131	33984.0	64.1	**35567**	0.000	35452.4	772.4	261rl1304RNDT40p2	63484	63484	0.000	62468.0	152.8	63349	0.213	62576.0	729.7
265rl1323T40p1	707	697	1.414	677.0	170.1	**707**	0.000	700.4	816.1	265rl1323RNDT40p1	1242	1242	0.000	1216.0	259.2	1229	1.047	1213.5	814.2
265rl1323T40p2	35400	35298	0.288	33998.0	72.7	**35400**	0.000	35314.7	812.1	265rl1323RNDT40p2	63179	63179	0.000	61818.0	247.2	63048	0.207	62167.2	830.9
276nrw1379T40p1	758	750	1.055	740.0	61.3	**758**	0.000	753.8	809.3	276nrw1379RNDT40p1	1158	1158	0.000	1139.0	136.0	1148	0.864	1124.0	995.0
276nrw1379T40p2	38511	38386	0.325	37607.0	140.1	**38511**	0.000	38260.9	793.6	276nrw1379RNDT40p2	59227	59227	0.000	58160.0	277.2	58263	1.628	56523.0	992.7
280fl1400T40p1	884	884	0.000	881.0	81.9	884	0.000	882.6	769.4	280fl1400RNDT40p1	1364	1363	0.073	1358.0	42.3	**1364**	0.000	1363.1	1063.8
280fl1400T40p2	45005	44951	0.120	43746.0	153.1	**45005**	0.000	44934.4	732.5	280fl1400RNDT40p2	68854	68854	0.000	68590.0	342.2	68854	0.000	68746.0	1068.4
287u1432T40p1	741	736	0.675	721.0	75.2	**741**	0.000	735.1	962.2	287u1432RNDT40p1	1146	1146	0.000	1129.0	155.2	1105	3.578	1073.9	1182.5
287u1432T40p2	37676	37676	0.000	37187.0	149.1	37505	0.454	37381.0	911.2	287u1432RNDT40p2	58551	58551	0.000	57677.0	338.7	56443	3.600	55163.4	1275.9
316fl1577T40p1	816	809	0.858	794.0	107.5	**816**	0.000	809.4	668.6	316fl1577RNDT40p1	1516	1516	0.000	1502.0	236.5	1502	0.923	1489.8	1253.2
316fl1577T40p2	42171	42171	0.000	40745.0	102.6	41569	1.428	41411.3	635.5	316fl1577RNDT40p2	76687	76687	0.000	76213.0	247.0	76087	0.782	74678.1	1269.0
331d1655T40p1	692	688	0.578	672.0	35.9	**692**	0.000	650.3	676.1	331d1655RNDT40p1	1266	1266	0.000	1207.0	268.3	1237	2.291	1188.0	1541.5
331d1655T40p2	35607	35539	0.191	35162.0	36.6	**35607**	0.000	33247.6	813.6	331d1655RNDT40p2	63834	63834	0.000	60171.0	212.0	62369	2.295	61243.6	1469.2
350vm1748T40p1	1165	1162	0.258	1149.0	128.3	**1165**	0.000	1162.6	1100.5	350vm1748RNDT40p1	1745	1745	0.000	1738.0	1194.6	1745	0.000	1743.6	1470.7
350vm1748T40p2	58755	58755	0.000	57322.0	122.5	58666	0.151	58616.7	1176.8	350vm1748RNDT40p2	88169	88169	0.000	87860.0	1147.1	88154	0.017	88013.8	1393.1
364u1817T40p1	925	915	1.081	896.0	35.6	**925**	0.000	916.8	2279.6	364u1817RNDT40p1	1486	1479	0.471	1449.0	428.8	**1486**	0.000	1450.6	2153.7
364u1817T40p2	46973	46273	1.490	45460.0	130.0	**46973**	0.000	46848.6	2200.4	364u1817RNDT40p2	75082	75043	0.052	72118.0	553.4	**75082**	0.000	73254.0	2242.5
378rl1889T40p1	986	985	0.101	968.0	130.1	**986**	0.000	984.0	2777.5	378rl1889RNDT40p1	1820	1806	0.769	1778.0	339.3	**1820**	0.000	1808.6	2290.8
378rl1889T40p2	50006	49869	0.274	47550.0	246.4	**50006**	0.000	49702.4	2732.0	378rl1889RNDT40p2	92699	91861	0.904	89359.0	1135.3	**92699**	0.000	91907.2	2680.6
421d2103T40p1	953	950	0.315	928.0	281.2	**953**	0.000	948.8	3600.0	421d2103RNDT40p1	1676	1666	0.597	1600.0	742.8	**1676**	0.000	1634.6	3600.0
421d2103T40p2	48536	48380	0.321	47114.0	224.7	**48536**	0.000	48255.2	3600.0	421d2103RNDT40p2	83224	82655	0.684	79855.0	821.9	**83224**	0.000	82087.2	3600.0
431u2152T40p1	1075	1058	1.581	1036.0	159.4	**1075**	0.000	1058.6	2410.0	431u2152RNDT40p1	1736	1736	0.000	1674.0	480.9	1712	1.382	1672.8	3600.0
431u2152T40p2	54859	53959	1.641	53443.0	637.0	**54859**	0.000	53881.6	2421.1	431u2152RNDT40p2	86274	85962	0.362	83734.0	1092.0	**86274**	0.000	83509.8	3600.0
464u2319T40p1	1122	1122	0.000	1102.0	487.2	1119	0.267	1112.4	2984.8	464u2319RNDT40p1	1573	1573	0.000	1546.0	416.2	1552	1.335	1533.6	3600.0
464u2319T40p2	57198	57198	0.000	55626.0	643.7	56915	0.495	56206.0	2947.3	464u2319RNDT40p2	80534	80534	0.000	79411.0	821.4	79730	0.998	78944.8	3600.0
479pr2392T40p1	1311	1301	0.763	1239.0	355.5	**1311**	0.000	1306.7	3600.0	479pr2392RNDT40p1	2091	2091	0.000	2044.0	895.0	2003	4.209	1906.8	2407.5
479pr2392T40p2	66032	65865	0.253	64184.0	267.8	**66032**	0.000	65783.4	3600.0	479pr2392RNDT40p2	105582	105582	0.000	102933.0	374.7	103477	1.994	101581.6	2323.8
608pcb3038T40p1	1556	1556	0.000	1518.0	1418.5	1556	0.000	1535.7	3600.0	608pcb3038RNDT40p1	2350	2350	0.000	2309.0	1871.4	2340	0.426	2267.6	3600.0
608pcb3038T40p2	79864	79864	0.000	78206.0	1553.0	79773	0.114	78213.4	3600.0	608pcb3038RNDT40p2	120356	120356	0.000	117274.0	2124.7	115849	3.745	113079.3	3600.0
633E3k.0T40p1	1575	1575	0.000	1536.0	991.0	1539	2.286	1523.0	3600.0	633E3k.0RNDT40p1	2548	2547	0.039	2500.0	763.8	**2548**	0.000	2485.2	3600.0
633E3k.0T40p2	81356	80978	0.465	79775.0	1823.5	**81356**	0.000	79783.2	3600.0	633E3k.0RNDT40p2	131012	131012	0.000	127453.0	2336.4	127537	2.652	124827.2	3600.0
Average		24057.4	0.700	23521.8	296.5	24151.1	0.137	23892.2	1725.8	Average		39293.6	0.192	38396.8	561.6	38927.1	0.997	38242.4	1891.9

Table B.2: Computational results on Set B instances with $\omega = 0.6$.

		AMMH				TMS			
Instance	f_{bk}	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$
200E1k.0T60p1	747	746	0.134	739.0	41.6	**747**	0.000	743.7	374.3
200E1k.0T60p2	38681	38617	0.165	38062.0	54.4	**38681**	0.000	38399.2	373.0
235pcb1173T60p1	868	865	0.346	845.0	76.5	**868**	0.000	860.5	721.2
235pcb1173T60p2	44070	43703	0.833	42502.0	61.6	**44070**	0.000	43620.2	695.0
259d1291T60p1	916	916	0.000	892.0	135.3	914	0.218	910.6	694.9
259d1291T60p2	46179	46179	0.000	44667.0	104.1	46003	0.381	44945.0	673.1
261rl1304T60p1	960	940	2.083	923.0	60.5	**960**	0.000	951.6	950.5
261rl1304T60p2	48904	48103	1.638	47410.0	200.3	**48904**	0.000	48574.3	928.6
265rl1323T60p1	1007	996	1.092	981.0	246.7	**1007**	0.000	998.4	767.6
265rl1323T60p2	51007	50504	0.986	49654.0	201.7	**51007**	0.000	50718.3	760.7
276nrw1379T60p1	1038	1028	0.963	1011.0	112.8	**1038**	0.000	1029.3	897.2
276nrw1379T60p2	52328	52043	0.545	51160.0	136.8	**52328**	0.000	51948.7	891.7
280fl1400T60p1	1329	1328	0.075	1326.0	136.6	**1329**	0.000	1328.4	752.5
280fl1400T60p2	67480	67480	0.000	67371.0	118.3	67467	0.019	67465.4	793.5
287u1432T60p1	1049	1049	0.000	1035.0	136.2	1048	0.095	1035.9	1152.3
287u1432T60p2	53648	53297	0.654	52467.0	76.0	**53648**	0.000	53119.4	1158.0
316fl1577T60p1	1142	1142	0.000	1081.0	85.0	1102	3.503	1097.8	1170.0
316fl1577T60p2	58840	58840	0.000	53906.0	47.7	55994	4.837	54507.6	1177.1
331d1655T60p1	1122	1117	0.446	1087.0	174.0	**1122**	0.000	1118.2	1369.5
331d1655T60p2	57038	57038	0.000	54420.0	357.4	56991	0.082	56800.4	1464.4
350vm1748T60p1	1473	1472	0.068	1448.0	166.9	**1473**	0.000	1469.2	2144.1
350vm1748T60p2	75419	75190	0.304	74266.0	195.8	**75419**	0.000	74854.3	2052.3
364u1817T60p1	1338	1320	1.345	1282.0	160.5	**1338**	0.000	1328.8	2880.0
364u1817T60p2	67686	66471	1.795	65133.0	257.8	**67686**	0.000	67071.2	3600.0
378rl1889T60p1	1387	1375	0.865	1355.0	638.6	**1387**	0.000	1380.8	3260.0
378rl1889T60p2	71087	71087	0.000	69565.0	298.7	71022	0.091	70625.2	3236.4
421d2103T60p1	1424	1400	1.685	1382.0	177.2	**1424**	0.000	1416.4	3600.0
421d2103T60p2	72868	71803	1.462	70743.0	309.0	**72868**	0.000	72028.8	3600.0
431u2152T60p1	1528	1523	0.327	1482.0	335.6	**1528**	0.000	1518.2	2531.2
431u2152T60p2	77530	77530	0.000	76151.0	640.8	76249	1.652	76094.3	2508.0
464u2319T60p1	1618	1611	0.433	1583.0	545.1	**1618**	0.000	1591.2	2964.9
464u2319T60p2	82101	81709	0.477	80471.0	589.8	**82101**	0.000	80997.3	2995.7
479pr2392T60p1	1808	1806	0.111	1765.0	247.2	**1808**	0.000	1804.6	3148.6
479pr2392T60p2	92444	91336	1.199	89339.0	398.5	**92444**	0.000	91528.6	3083.2
608pcb3038T60p1	2194	2179	0.684	2135.0	1358.7	**2194**	0.000	2130.8	3600.0
608pcb3038T60p2	112771	111346	1.264	108891.0	1584.8	**112771**	0.000	110561.2	3600.0
633E3k.0T60p1	2220	2220	0.000	2188.0	1104.7	2213	0.315	2162.0	3600.0
633E3k.0T60p2	114753	114753	0.000	112906.0	1759.4	113495	1.096	112365.2	3600.0
Average		34264.8	0.578	33516.4	350.9	34322.8	0.323	33976.3	1941.3

		AMMH				TMS			
Instance	f_{bk}	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$
200E1k.0RNDT60p1	998	991	0.701	982.0	25.5	**998**	0.000	993.2	122.8
200E1k.0RNDT60p2	50369	50321	0.095	50073.0	51.7	**50369**	0.000	50211.3	138.7
235pcb1173RNDT60p1	1165	1165	0.000	1157.0	138.8	1157	0.687	1151.6	281.2
235pcb1173RNDT60p2	58665	58665	0.000	58270.0	182.2	58480	0.315	58180.4	276.9
259d1291RNDT60p1	1288	1288	0.000	1283.0	149.9	1286	0.155	1281.0	372.7
259d1291RNDT60p2	65085	65031	0.083	64954.0	207.2	**65085**	0.000	64970.8	355.8
261rl1304RNDT60p1	1303	1303	0.000	1301.0	24.2	1303	0.000	1302.4	149.7
261rl1304RNDT60p2	65841	65841	0.000	65825.0	71.9	65841	0.000	65827.5	175.4
265rl1323RNDT60p1	1322	1322	0.000	1320.0	130.3	1322	0.000	1321.5	131.7
265rl1323RNDT60p2	66753	66753	0.000	66738.0	90.9	66753	0.000	66753.0	146.0
276nrw1379RNDT60p1	1365	1365	0.000	1358.0	167.4	1353	0.879	1344.1	590.3
276nrw1379RNDT60p2	68905	68905	0.000	68567.0	144.4	68639	0.386	68037.8	566.4
280fl1400RNDT60p1	1399	1399	0.000	1399.0	4.9	1399	0.000	1399.0	239.8
280fl1400RNDT60p2	70685	70685	0.000	70685.0	3.5	70685	0.000	70685.0	233.0
287u1432RNDT60p1	1386	1386	0.000	1373.0	398.2	1370	1.154	1362.4	697.4
287u1432RNDT60p2	70390	70390	0.000	69881.0	207.3	69991	0.567	69183.8	751.8
316fl1577RNDT60p1	1574	1574	0.000	1574.0	328.8	1574	0.000	1574.0	490.3
316fl1577RNDT60p2	79505	79505	0.000	79449.0	358.9	79471	0.043	79438.3	502.6
331d1655RNDT60p1	1631	1631	0.000	1614.0	237.4	1609	1.349	1597.2	676.0
331d1655RNDT60p2	82108	82108	0.000	81330.0	171.5	81537	0.695	80305.8	685.3
350vm1748RNDT60p1	1747	1747	0.000	1747.0	337.6	1747	0.000	1747.0	140.0
350vm1748RNDT60p2	88203	88203	0.000	88203.0	78.4	88203	0.000	88203.0	141.3
364u1817RNDT60p1	1794	1790	0.223	1779.0	2617.4	**1794**	0.000	1782.0	796.4
364u1817RNDT60p2	90936	90936	0.000	89909.0	1931.0	90577	0.395	90145.7	719.7
378rl1889RNDT60p1	1888	1888	0.000	1888.0	48.8	1888	0.000	1888.0	216.7
378rl1889RNDT60p2	95376	95376	0.000	95376.0	79.2	95376	0.000	95376.0	245.0
421d2103RNDT60p1	2092	2092	0.000	2078.0	645.2	2087	0.239	2078.0	1270.8
421d2103RNDT60p2	105940	105940	0.000	104893.0	955.5	105420	0.491	104757.6	1200.4
431u2152RNDT60p1	2106	2106	0.000	2091.0	422.6	2090	0.760	2084.4	1507.0
431u2152RNDT60p2	106635	106635	0.000	105740.0	185.3	106014	0.582	105277.2	1464.0
464u2319RNDT60p1	2060	2060	0.000	2041.0	731.6	2056	0.194	2043.0	2734.8
464u2319RNDT60p2	105486	105486	0.000	104361.0	759.4	105007	0.454	104219.4	3600.0
479pr2392RNDT60p1	2390	2390	0.000	2364.0	651.5	2387	0.126	2372.3	1642.4
479pr2392RNDT60p2	120549	120549	0.000	119652.0	374.4	119865	0.567	119732.0	1662.4
608pcb3038RNDT60p1	2928	2928	0.000	2891.0	465.9	2897	1.059	2888.7	3600.0
608pcb3038RNDT60p2	148365	148365	0.000	146398.0	951.8	145681	1.809	144755.7	3600.0
633E3k.0RNDT60p1	3102	3102	0.000	3096.0	797.4	3102	0.000	3094.6	3600.0
633E3k.0RNDT60p2	157436	157436	0.000	155442.0	902.4	156618	0.520	156201.1	3600.0
Average		45543.6	0.029	45239.0	421.8	45342.9	0.353	45146.5	1034.9

Table B.3: Computational results on Set B instances with $\omega = 0.8$.

		AMMH				TMS			
Instance	f_{bk}	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$
200E1k.0T80p1	911	911	0.000	904.0	54.7	911	0.000	902.6	325.0
200E1k.0T80p2	46219	46089	0.281	45696.0	114.6	**46219**	0.000	46056.4	305.4
235pcb1173T80p1	1064	1044	1.880	1034.0	56.8	**1064**	0.000	1051.6	531.6
235pcb1173T80p2	53798	53138	1.227	52441.0	97.4	**53798**	0.000	53371.0	540.8
259d1291T80p1	1168	1150	1.541	1117.0	105.5	**1168**	0.000	1165.6	707.7
259d1291T80p2	58732	57719	1.725	56423.0	87.6	**58732**	0.000	57433.6	737.6
261rl1304T80p1	1183	1178	0.423	1161.0	172.1	**1183**	0.000	1176.7	668.7
261rl1304T80p2	60042	59802	0.400	59242.0	167.8	**60042**	0.000	59909.1	686.0
265rl1323T80p1	1221	1216	0.410	1203.0	131.5	**1221**	0.000	1217.2	680.8
265rl1323T80p2	62164	61601	0.906	61060.0	114.9	**62164**	0.000	61882.6	678.9
276nrw1379T80p1	1240	1235	0.403	1224.0	183.5	**1240**	0.000	1236.6	927.6
276nrw1379T80p2	62659	62346	0.500	61547.0	98.6	**62659**	0.000	62352.0	931.5
280fl1400T80p1	1386	1386	0.000	1385.0	332.0	1386	0.000	1385.0	305.6
280fl1400T80p2	70006	70001	0.007	69996.0	145.3	**70006**	0.000	69997.3	281.0
287u1432T80p1	1280	1269	0.859	1263.0	156.7	**1280**	0.000	1275.7	748.5
287u1432T80p2	65300	65300	0.000	64515.0	330.3	65110	0.291	64803.4	694.6
316fl1577T80p1	1399	1394	0.357	1367.0	177.4	**1399**	0.000	1396.4	1369.1
316fl1577T80p2	70960	70436	0.738	68176.0	174.3	**70960**	0.000	70771.4	1318.6
331d1655T80p1	1466	1462	0.273	1437.0	212.2	**1466**	0.000	1455.4	1420.2
331d1655T80p2	74160	74140	0.027	72897.0	204.0	**74160**	0.000	73822.8	1420.5
350vm1748T80p1	1665	1662	0.180	1651.0	296.7	**1665**	0.000	1660.6	1145.7
350vm1748T80p2	84461	84266	0.231	84047.0	198.2	**84461**	0.000	84412.8	1163.8
364u1817T80p1	1635	1615	1.223	1595.0	525.6	**1635**	0.000	1629.0	1373.7
364u1817T80p2	82576	82110	0.564	81135.0	265.7	**82576**	0.000	82194.4	1398.4
378rl1889T80p1	1701	1698	0.176	1673.0	611.8	**1701**	0.000	1693.4	2116.1
378rl1889T80p2	86417	85942	0.550	84796.0	351.9	**86417**	0.000	86212.8	2156.2
421d2103T80p1	1849	1826	1.244	1778.0	499.5	**1849**	0.000	1823.3	1876.7
421d2103T80p2	94085	92994	1.160	90522.0	372.6	**94085**	0.000	92685.8	1875.9
431u2152T80p1	1898	1883	0.790	1861.0	255.1	**1898**	0.000	1886.2	2198.5
431u2152T80p2	96084	96058	0.027	94660.0	237.7	**96084**	0.000	95497.4	2170.6
464u2319T80p1	2039	2039	0.000	1999.0	1336.5	2027	0.589	2020.4	2567.1
464u2319T80p2	103130	102299	0.806	101564.0	635.2	**103130**	0.000	102598.8	2721.5
479pr2392T80p1	2161	2154	0.324	2136.0	428.9	**2161**	0.000	2156.2	2935.6
479pr2392T80p2	109663	109537	0.115	108405.0	314.8	**109663**	0.000	109196.2	2941.0
608pcb3038T80p1	2705	2705	0.000	2680.0	236.7	2673	1.183	2652.0	3600.0
608pcb3038T80p2	137872	137872	0.000	135393.0	691.6	136667	0.874	135617.0	3600.0
633E3k.0T80p1	2778	2758	0.720	2725.0	725.6	**2778**	0.000	2751.8	3600.0
633E3k.0T80p2	140711	140711	0.000	139067.0	601.3	139365	0.957	138193.7	3600.0
Average		41656.5	0.528	41099.3	308.0	41763.2	0.102	41514.3	1534.7

		AMMH				TMS			
Instance	f_{bk}	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$
200E1k.0RNDT80p1	999	999	0.000	998.0	18.1	999	0.000	999.0	111.4
200E1k.0RNDT80p2	50485	50485	0.000	50485.0	2.1	50485	0.000	50485.0	95.6
235pcb1173RNDT80p1	1172	1172	0.000	1172.0	11.9	1172	0.000	1172.0	123.2
235pcb1173RNDT80p2	59178	59178	0.000	59175.0	12.2	59178	0.000	59178.0	129.8
259d1291RNDT80p1	1290	1290	0.000	1290.0	15.0	1290	0.000	1290.0	159.9
259d1291RNDT80p2	65145	65145	0.000	65145.0	3.6	65145	0.000	65145.0	168.5
261rl1304RNDT80p1	1303	1303	0.000	1303.0	5.2	1303	0.000	1303.0	165.3
261rl1304RNDT80p2	65841	65841	0.000	65841.0	4.7	65841	0.000	65841.0	168.4
265rl1323RNDT80p1	1322	1322	0.000	1322.0	2.7	1322	0.000	1322.0	167.8
265rl1323RNDT80p2	66753	66753	0.000	66753.0	3.3	66753	0.000	66753.0	167.0
276nrw1379RNDT80p1	1378	1378	0.000	1377.0	6.8	1378	0.000	1378.0	193.0
276nrw1379RNDT80p2	69541	69541	0.000	69539.0	97.4	69541	0.000	69541.0	190.5
280fl1400RNDT80p1	1399	1399	0.000	1399.0	3.3	1399	0.000	1399.0	249.4
280fl1400RNDT80p2	70685	70685	0.000	70685.0	0.0	70685	0.000	70685.0	222.0
287u1432RNDT80p1	1431	1431	0.000	1431.0	16.3	1431	0.000	1431.0	207.6
287u1432RNDT80p2	72301	72301	0.000	72301.0	24.8	72301	0.000	72301.0	215.0
316fl1577RNDT80p1	1576	1576	0.000	1576.0	6.4	1576	0.000	1576.0	322.5
316fl1577RNDT80p2	79556	79556	0.000	79556.0	5.0	79556	0.000	79556.0	286.1
331d1655RNDT80p1	1654	1654	0.000	1654.0	24.8	1654	0.000	1654.0	255.6
331d1655RNDT80p2	83495	83495	0.000	83495.0	5.8	83495	0.000	83495.0	237.6
350vm1748RNDT80p1	1747	1747	0.000	1747.0	18.2	1747	0.000	1747.0	119.9
350vm1748RNDT80p2	88203	88203	0.000	88203.0	7.0	88203	0.000	88203.0	129.7
364u1817RNDT80p1	1816	1816	0.000	1816.0	13.7	1816	0.000	1816.0	282.1
364u1817RNDT80p2	91716	91716	0.000	91716.0	28.0	91716	0.000	91716.0	270.7
378rl1889RNDT80p1	1888	1888	0.000	1888.0	12.9	1888	0.000	1888.0	315.0
378rl1889RNDT80p2	95376	95376	0.000	95376.0	12.8	95376	0.000	95376.0	316.8
421d2103RNDT80p1	2102	2102	0.000	2102.0	49.2	2102	0.000	2102.0	638.5
421d2103RNDT80p2	106203	106203	0.000	106203.0	145.5	106203	0.000	106203.0	628.2
431u2152RNDT80p1	2151	2151	0.000	2151.0	38.6	2151	0.000	2151.0	718.7
431u2152RNDT80p2	108581	108581	0.000	108581.0	167.2	108581	0.000	108581.0	725.3
464u2319RNDT80p1	2309	2309	0.000	2303.0	337.3	2298	0.476	2296.7	1099.5
464u2319RNDT80p2	116900	116900	0.000	116720.0	637.9	116447	0.388	116178.0	1251.4
479pr2392RNDT80p1	2391	2391	0.000	2391.0	20.4	2391	0.000	2391.0	615.2
479pr2392RNDT80p2	120741	120741	0.000	120741.0	143.8	120741	0.000	120741.0	685.0
608pcb3038RNDT80p1	3037	3037	0.000	3037.0	461.2	3037	0.000	3037.0	878.5
608pcb3038RNDT80p2	153378	153378	0.000	153378.0	1170.6	153378	0.000	153378.0	835.0
633E3k.0RNDT80p1	3161	3161	0.000	3161.0	576.8	3161	0.000	3161.0	747.2
633E3k.0RNDT80p2	159596	159596	0.000	159596.0	136.6	159596	0.000	159596.0	723.3
Average		46257.9	0.000	46252.8	111.8	46245.7	0.023	46238.6	389.9

Table B.4: Computational results on Set B instances with $\omega = 1.0$.

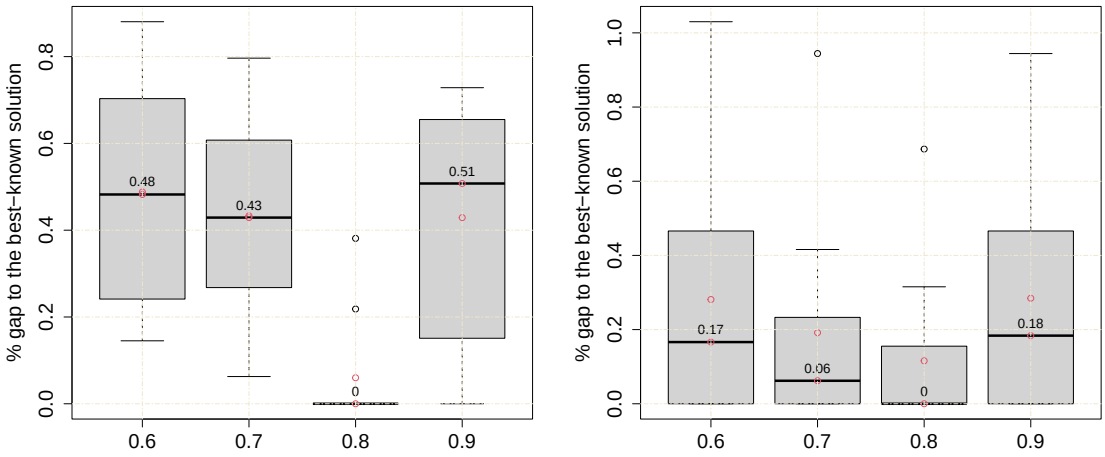
		AMMH			TMS						AMMH				TMS			
Instance	f_{bk}	f_{best}	Dev.(%)	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$	Instance	f_{bk}	f_{best}	Dev.(%)	f_{avg}	$t(s)$	f_{best}	Dev.(%)	f_{avg}	$t(s)$
200E1k.0T100p1	997	993	0.401	60.1	**997**	0.000	994.9	95.0	200E1k.0RNDT100p1	999	999	0.000	999.0	40.3	999	0.000	999.0	104.0
200E1k.0T100p2	50447	50279	0.333	51.9	**50447**	0.000	50383.6	108.2	200E1k.0RNDT100p2	50485	50485	0.000	50485.0	0.3	50485	0.000	50485.0	103.1
235pcb1173T100p1	1166	1159	0.600	49.3	**1166**	0.000	1162.6	209.2	235pcb1173RNDT100p1	1172	1172	0.000	1172.0	2.1	1172	0.000	1172.0	153.1
235pcb1173T100p2	58856	58567	0.491	52.0	**58856**	0.000	58698.7	179.1	235pcb1173RNDT100p2	59178	59178	0.000	59178.0	2.6	59178	0.000	59178.0	152.8
259d1291T100p1	1286	1285	0.078	151.2	**1286**	0.000	1285.9	261.2	259d1291RNDT100p1	1290	1290	0.000	1290.0	0.3	1290	0.000	1290.0	162.7
259d1291T100p2	65071	65023	0.074	63.9	**65071**	0.000	65024.6	223.5	259d1291RNDT100p2	65145	65145	0.000	65145.0	2.6	65145	0.000	65145.0	170.4
261rl1304T100p1	1299	1297	0.154	57.5	**1299**	0.000	1298.3	220.7	261rl1304RNDT100p1	1303	1303	0.000	1303.0	2.7	1303	0.000	1303.0	151.3
261rl1304T100p2	65715	65614	0.154	95.9	**65715**	0.000	65656.3	232.0	261rl1304RNDT100p2	65841	65841	0.000	65841.0	3.3	65841	0.000	65841.0	179.7
265rl1323T100p1	1319	1317	0.152	66.7	**1319**	0.000	1318.5	257.7	265rl1323RNDT100p1	1322	1322	0.000	1322.0	3.0	1322	0.000	1322.0	160.0
265rl1323T100p2	66681	66635	0.069	152.3	**66681**	0.000	66635.3	264.9	265rl1323RNDT100p2	66753	66753	0.000	66753.0	3.3	66753	0.000	66753.0	161.3
276nrw1379T100p1	1367	1361	0.439	85.5	**1367**	0.000	1361.1	312.0	276nrw1379RNDT100p1	1378	1378	0.000	1378.0	2.9	1378	0.000	1378.0	253.5
276nrw1379T100p2	69098	68942	0.226	116.4	**69098**	0.000	68802.4	282.9	276nrw1379RNDT100p2	69541	69541	0.000	69541.0	3.3	69541	0.000	69541.0	290.8
280fl1400T100p1	1398	1398	0.000	264.2	1398	0.000	1398.0	374.7	280fl1400RNDT100p1	1399	1399	0.000	1399.0	2.4	1399	0.000	1399.0	239.8
280fl1400T100p2	70666	70666	0.000	310.9	70666	0.000	70665.6	387.9	280fl1400RNDT100p2	70685	70685	0.000	70685.0	171.0	70685	0.000	70685.0	230.3
287u1432T100p1	1428	1424	0.280	101.3	**1428**	0.000	1425.1	241.4	287u1432RNDT100p1	1431	1431	0.000	1431.0	3.8	1431	0.000	1431.0	296.0
287u1432T100p2	72093	71932	0.223	92.7	**72093**	0.000	71934.3	233.1	287u1432RNDT100p2	72301	72301	0.000	72301.0	4.0	72301	0.000	72301.0	293.8
316fl1577T100p1	1571	1570	0.064	215.7	**1571**	0.000	1570.4	534.6	316fl1577RNDT100p1	1576	1576	0.000	1576.0	3.5	1576	0.000	1576.0	354.1
316fl1577T100p2	79478	79429	0.062	125.4	**79478**	0.000	79432.6	439.5	316fl1577RNDT100p2	79556	79556	0.000	79556.0	4.3	79556	0.000	79556.0	298.0
331d1655T100p1	1648	1648	0.000	185.9	1648	0.000	1645.2	389.9	331d1655RNDT100p1	1654	1654	0.000	1654.0	8.1	1654	0.000	1654.0	258.6
331d1655T100p2	83322	83183	0.167	86.4	**83322**	0.000	83147.4	360.5	331d1655RNDT100p2	83495	83495	0.000	83495.0	0.5	83495	0.000	83495.0	271.6
350vm1748T100p1	1744	1743	0.057	825.1	**1744**	0.000	1743.0	369.0	350vm1748RNDT100p1	1747	1747	0.000	1747.0	4.6	1747	0.000	1747.0	123.5
350vm1748T100p2	88172	88168	0.005	403.1	**88172**	0.000	88160.0	359.0	350vm1748RNDT100p2	88203	88203	0.000	88203.0	5.3	88203	0.000	88203.0	125.5
364u1817T100p1	1806	1803	0.166	691.0	**1806**	0.000	1803.0	416.4	364u1817RNDT100p1	1816	1816	0.000	1816.0	4.4	1816	0.000	1816.0	296.3
364u1817T100p2	91430	91321	0.119	930.6	**91430**	0.000	91346.4	426.1	364u1817RNDT100p2	91716	91716	0.000	91716.0	4.4	91716	0.000	91716.0	290.8
378rl1889T100p1	1883	1878	0.266	1396.2	**1883**	0.000	1880.6	424.1	378rl1889RNDT100p1	1888	1888	0.000	1888.0	10.0	1888	0.000	1888.0	274.9
378rl1889T100p2	95187	95084	0.108	942.0	**95187**	0.000	95091.8	434.3	378rl1889RNDT100p2	95376	95376	0.000	95376.0	228.8	95376	0.000	95376.0	276.9
421d2103T100p1	2092	2090	0.096	1250.4	**2092**	0.000	2091.1	845.7	421d2103RNDT100p1	2102	2102	0.000	2102.0	10.2	2102	0.000	2102.0	667.0
421d2103T100p2	105611	105385	0.214	1564.7	**105611**	0.000	105540.6	814.2	421d2103RNDT100p2	106203	106203	0.000	106203.0	2.3	106203	0.000	106203.0	652.9
431u2152T100p1	2133	2126	0.328	1703.1	**2133**	0.000	2129.1	818.0	431u2152RNDT100p1	2151	2151	0.000	2151.0	124.9	2151	0.000	2151.0	829.6
431u2152T100p2	107748	107659	0.083	1509.1	**107748**	0.000	107457.2	763.1	431u2152RNDT100p2	108581	108581	0.000	108581.0	163.7	108581	0.000	108581.0	852.7
464u2319T100p1	2298	2298	0.000	2518.5	2292	0.261	2284.0	1229.3	464u2319RNDT100p1	2318	2318	0.000	2318.0	20.8	2318	0.000	2318.0	706.3
464u2319T100p2	116014	116014	0.000	1852.7	115845	0.146	115669.7	1266.3	464u2319RNDT100p2	117031	117031	0.000	117031.0	145.6	117031	0.000	117031.0	730.5
479pr2392T100p1	2374	2365	0.379	2941.9	**2374**	0.000	2371.4	896.0	479pr2392RNDT100p1	2391	2391	0.000	2391.0	151.4	2391	0.000	2391.0	674.4
479pr2392T100p2	120080	119436	0.536	2362.3	**120080**	0.000	119956.4	845.0	479pr2392RNDT100p2	120741	120741	0.000	120741.0	148.7	120741	0.000	120741.0	630.5
608pcb3038T100p1	3016	3007	0.298	2353.2	**3016**	0.000	3012.6	1622.6	608pcb3038RNDT100p1	3037	3037	0.000	3037.0	119.9	3037	0.000	3037.0	853.5
608pcb3038T100p2	152469	152046	0.277	2180.7	**152469**	0.000	152207.8	1746.4	608pcb3038RNDT100p2	153378	153378	0.000	153378.0	417.2	153378	0.000	153378.0	825.4
633E3k.0T100p1	3111	3108	0.096	3825.3	**3111**	0.000	3102.4	2731.3	633E3k.0RNDT100p1	3161	3161	0.000	3161.0	166.3	3161	0.000	3161.0	942.5
633E3k.0T100p2	157539	156902	0.404	4645.3	**157539**	0.000	157112.0	2905.0	633E3k.0RNDT100p2	159596	159596	0.000	159596.0	87.1	159596	0.000	159596.0	917.5
Average		45951.4	0.195	954.7	46037.8	0.011	45968.4	645.3	Average		46261.6	0.000	46261.6	54.7	46261.6	0.000	46261.6	393.6

Appendix C.

Regarding the ρ similarity threshold for population initialization and the adaptive φ parameter for strategic oscillation, we have conducted rigorous sensitivity experiments across different instances (clustered vs. random). For each parameter, we test a reasonable range of candidate values while keeping all other algorithmic parameters fixed.

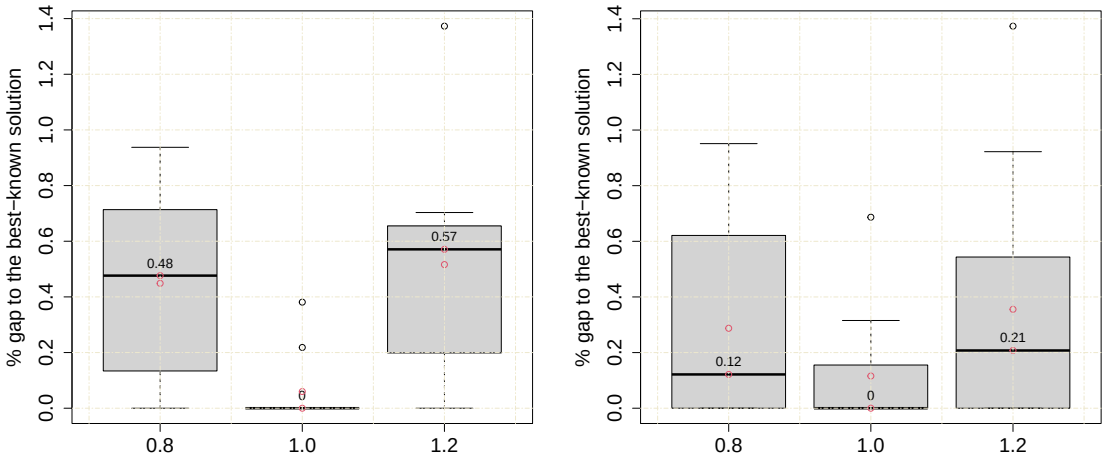
The Friedman rank sum test is employed to detect statistically significant performance differences among alternative parameter values. The test results show significant differences for the similarity threshold ρ (p-value = 0.0002538 for clustered instances, p-value = 0.001143 for random instances) and for the φ parameter (p-value = 0.0009655 for clustered instances, p-value = 0.009404 for random instances). The distribution and ranges of the best objective values are presented in the form of box-and-whisker plots (see Fig. C.1). Both parameters exhibit statistical sensitivity (p-value < 0.05) within each instance type, and they favor the same best parameter configuration consistently across clustered and random instances.

The experimental results demonstrate that our adopted parameter setting enables both mechanisms to remain stable and robust over different instance types, which verifies the rationality and robustness of the parameter settings used in our algorithm.



(a) ρ: Friedman p-value = 0.0002538 (clustered instances)

(b) ρ: Friedman p-value = 0.001143 (random instances)



(c) φ: Friedman p-value = 0.0009655 (clustered instances)

(d) φ: Friedman p-value = 0.009404 (random instances)

Figure C.1: Boxplots of the normalized best objective values for considered values of each parameter.