Multilevel iterated tabu search for the multi-constraint graph partitioning problem

Zhi Lu[a], Linlin Chen[a], Jian Gao[b], Jin-Kao Hao[c,*]

[a]*Business School, University of Shanghai for Science & Technology, 516 Jungong Rd., Shanghai 200093, China*
[b]*School of Information Science and Technology, Northeast Normal University, 5268 Renmin Street, Changchun 130024, China*
[c]*LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

Computers & Operations Research 189: 107389, May 2026

Abstract

The multi-constraint graph partitioning (MCGP) problem involves partitioning a set of vertices into nonempty, pairwise-disjoint subsets such that each subset must satisfy certain bound constraints while minimizing the total cost of edges with both endpoints in the same subset. Arising from an integrated vehicle and pollster problem in a real-world application, MCGP generalizes a number of other well-known graph partitioning problems. Due to its NP-hard nature, solving MCGP is computationally challenging. This work presents the first multilevel iterated tabu search (MITS) algorithm to tackle MCGP. Specifically, the algorithm uses a problem-specific coarsening method to reduce progressively the input graph and relies on a dedicated feasible-and-infeasible iterated tabu search procedure to refine the solution to each reduced graph. Extensive experiments on two sets of 665 benchmark instances demonstrate that MITS significantly outperforms state-of-the-art algorithms by finding 573 new upper bounds, while matching 83 previous best-known upper bounds. We also apply the algorithm to another related graph partitioning problem to demonstrate its broader applicability. Additionally, we conduct analysis studies on key algorithmic components to verify the effectiveness of the proposed ideas and strategies.

Keywords: Graph partitioning, heuristics, multilevel, tabu search, feasible and infeasible search.

*Corresponding author. jin-kao.hao@univ-angers.fr

1. Introduction

Graph partitioning problems are popular and general models that are useful for formulating numerous practical applications in various domains. Given an undirected graph with a vertex set and an edge set, graph partitioning problems are to divide the vertex set into two or more disjoint subsets while optimizing a defined objective and in some cases satisfying some given constraints. The study of graph partitioning problems began in the sixties, and related problems have emerged over time [4, 5]. Among them are graph partitioning problems with multi-constraints [24, 35, 2, 36], which arise in parallel preconditioned iterative methods [41], task assignment [38], and circuit partitioning [46], among others. The multi-constraint graph partitioning (MCGP) problem studied in this work is an extension of previous constrained graph partitioning problems.

Let $G = (V, E)$ be a complete, undirected graph with a vertex set $V = \{1, ..., n\}$ and an edge set $E = \{\{i, j\} : i, j \in V, i \neq j\}$, where $n = |V|$ and $m = |E|$ are the number of vertices and edges, respectively. Let $d : E \to \mathbb{R}^+$ be a function that assigns a cost (also called distance) $d(i, j)$ to each edge $\{i, j\}$ of the graph. For an integer $\kappa \geq 2$ and the set $K = \{1, ..., \kappa\}$, we define $\mathcal{P} = \{S_1, ..., S_\kappa\}$ as a κ-partition of the vertex set V, i.e., $\bigcup_{k=1}^{\kappa} S_k = V$, $S_i \cap S_j = \emptyset$, $S_i, S_j \neq \emptyset$, $i \neq j$, $\forall i, j \in K$. Furthermore, let $T = \{1, ..., \tau\}$ be a set of attributes and $W_v = (w_v^1, ..., w_v^\tau) > 0$ be a τ-dimensional vector of weights associated with each vertex $v \in V$. Given a partition $\mathcal{P} = \{S_1, ..., S_\kappa\}$, for each subset $S_k, k \in K$ and each attribute $t \in T$, the weight of S_k with respect to $t \in T$ is given by $w(\mathcal{P}, S_k, t) = \sum_{v \in S_k} w_v^t$. The partition is feasible only if for each subset S_k, its weight on each attribute t, $w(\mathcal{P}, S_k, t)$, is between a given lower bound W_{kt}^L and a given upper bound W_{kt}^U.

The MCGP problem consists in finding a κ-partition $\mathcal{P} = \{S_1, ..., S_\kappa\}$ such that the objective function $F(\mathcal{P})$ defined by Eq. (1) is minimized and the weight constraints defined by Eq. (2) are satisfied (see Fig. 2 for an illustrative example).

$$\min \; F(\mathcal{P}) = \sum_{k=1}^{\kappa} \sum_{i,j \in S_k, i \neq j} d(i, j) \tag{1}$$

$$\text{s.t.} \;\; W_{kt}^L \leq w(\mathcal{P}, S_k, t) \leq W_{kt}^U, \;\; \forall k \in K \wedge \forall t \in T \tag{2}$$

The MCGP problem arises from an integrated vehicle and pollster problem faced by the National Statistics Bureau of Ecuador (INEC) [33], which

conducts monthly polls at a fixed set of stores to monitor consumer price behavior for basic commodities. The purpose is to divide the set of stores into subsets corresponding to stores to be visited on the same day, while minimizing the distance traveled between stores within each subset to increase operational efficiency. These subsets must satisfy various constraints, such as the number of stores available per day, working time, and waiting time for pollsters, all of which have lower and upper limits. The problem instances studied in [33] are samples of real-world data from Guayaquil, Ecuador's second most populous city, with vertices $n \in [20, 55]$, $\kappa = \{3, 4, 5, 6\}$, and two attributes ($\tau = 2$) that correspond to working times and waiting times. Since the number of stores in the resulting subsets is small (typically < 20), the routing problem within each store partition is not critical. Consequently, the study of [33] focused on the more challenging store partition problem by introducing the MCGP model, which has broader applications.

MCGP defined in Eqs. (1) and (2) generalizes a number of other graph partitioning problems. If the weight constraints (2) are ignored, MCGP becomes 1) the κ-partitioning problem [9], 2) the κ-way equipartition problem [31] when $\tau = 1$ (i.e., $W_{kt}^L = W_k^L$, $W_{kt}^U = W_k^U$), $n \bmod \kappa \equiv 0$, $W_v = (1, ..., 1)$, and $W_k^L = W_k^U = n/\kappa, \forall v \in V, \forall k \in K$, 3) the size-constrained graph partitioning problem [27] when the lower and upper bounds are posed on the size of each subset of the partition, and 4) the general graph partitioning problem [13] and the equipartition problem [10, 26] with $\kappa = 2$ and $|S_1| = |S_2|$, and the bisection problem [10, 26] with $\kappa = 2$ and $|S_1| \neq |S_2|$ when the cardinalities of each subset of the partition are given a prior, $\tau = 1$, and $W_k^L = W_k^U = |S_k|, \forall k \in K$. If the weight constraint (2) is kept with $\tau = 2$, MCGP becomes the balanced κ-way graph partitioning problem with weight constraints BKPWC) [32]. All of these problems as well as MCGP are NP-hard [33], and it is unlikely to find a polynomial-time algorithm to optimally solve them. Therefore, it's critical to design effective heuristic solution methods that can achieve good results in a reasonable computation time.

While graph partitioning problems have attracted considerable attention from researchers and practitioners, few efforts have been devoted to solving MCGP. The existing exact method is based on Integer Programming (IP) formulations of the problem [33]. Tested on 25 small instances with $n \in [20, 55]$, $\kappa = \{3, 4, 5, 6\}$, and $\tau = 2$, the method was able to find feasible solutions for these instances and to prove the optimality of solutions for the 10 smallest instances. Recently, a heuristic algorithm [20] has been developed, which is based on the general GRASP metaheuristic. To enrich the set

of solution methods for solving the computationally challenging MCGP, we propose the first effective multilevel iterated tabu search (MITS) algorithm. The contributions of this work are listed as follows.

- Following the general framework of the multilevel optimization approach, MITS uses a problem-specific coarsening method to reduce progressively the input graph and relies on a dedicated feasible-and-infeasible Iterated Tabu Search (ITS) procedure to refine the solution to each reduced graph. Specifically, ITS consists of three novel and effective search components, 1) it explores both feasible and infeasible solutions with four complementary move operators (including two new ones *Push* and *Relocate*) to attain high-quality solutions, 2) it benefits from a fast incremental evaluation technique to ensure high computation efficiency, and 3) it applies an informed greedy randomized perturbation to escape deep local optima and diversify the search.

- We perform extensive experiments to evaluate the performance of MITS on two sets of 665 benchmark instances. Computational results demonstrate that MITS is highly effective compared to state-of-the-art algorithms in terms of both solution quality and computation time. In particular, MITS can find new upper bounds on 573 instances and match the previous best-known upper bounds on 83 instances. To demonstrate its broader applicability, we apply MITS to the related κ-way graph partitioning problem with weight constraints and report competitive results compared to those of dedicated, state-of-the-art approaches. Additionally, we investigate key components of MITS to confirm the importance of the proposed ideas and techniques.

The rest of the paper is organized as follows. Section 2 discusses related work on MCGP. Section 3 describes the MITS algorithm in detail. Section 4 conducts extensive performance comparisons between MITS and state-of-the-art algorithms. Section 5 presents the application of MITS to the related BKPWC problem. Section 6 analyzes the key algorithmic components to verify their impacts on the performance of the algorithm. The last section presents conclusions and perspectives.

2. Related work

We review existing MCGP studies, and the powerful multilevel optimization approach to graph partitioning problems and other hard combinatorial

optimization problems.

2.1. The MCGP problem

MCGP comes from an integrated vehicle and pollster problem in a real-world application. By now, two studies have been performed on MCGP. Recalde et al. [33] first defined MCGP and discussed its NP-hardness. They provided two integer programming formulations of the problem, and proved several families of valid inequalities associated with the respective polyhedra. They also proposed a preprocessing technique based on Branch & Bound and cutting planes to reduce the number of variables and strengthen the MCGP formulations. They generated 25 small instances from the real-world vehicle and pollster routing problem [32]. Their results showed that the exact methods and valid inequalities are effective and can find optimal solutions for the 10 smallest instances.

Herrán et al. [20] proposed the first heuristic algorithm called two-phase reactive GRASP (R-GRASP) to tackle MCGP. R-GRASP uses three constructive methods with a reactive parameter selection rule to generate an initial solution. Then, a two-stage local search based on two move operators (*insert* and *exchange*) is used to explore the search space. The first stage aims to find a feasible solution, while the second stage tries to improve the quality of the feasible solution with respect to the objective function F. They also designed efficient incremental neighborhood evaluation techniques for the two move operators, achieving an average execution time savings of 87%. They conducted experimental studies comparing R-GRASP with the previous work [33] and a general-purpose local search solver `Hexaly` [21]. Their results showed that R-GRASP achieved the best solutions for all 25 small instances quickly. They also generated a new set of 640 larger instances from the TSP Library and provided detailed results for future comparisons.

Our literature review indicates that the existing exact approaches are limited to small instances. Meanwhile, only one heuristic algorithm was dedicated to MCGP. To enrich the solution approaches for solving this challenging problem, we are devoted to designing an effective heuristic algorithm for MCGP.

2.2. Multilevel optimization

Multilevel optimization is a powerful and practical approach for solving large-scale optimization problems in graphs [23, 43]. This approach is particularly popular for dealing with partitioning and clustering tasks in large

complex graphs. The approach consists of three phases, as illustrated in Fig. 1: graph coarsening, solution refinement, and graph uncoarsening. To make the approach successful, each of these three fundamental phases must be carefully designed and tailored to specific features of the problem under investigation [22, 39, 44, 45, 8]. A detailed survey of recent advances in multilevel optimization can be found in [43].

Since graph partitioning problems motivated the earliest research efforts on the multilevel approach, we review the most recent and relevant studies. Sanders & Schulz [34] presented a multilevel network partitioning algorithm based on multigrid linear solvers for the graph κ-partitioning problem. Benlic & Hao [3] introduced a multilevel approach that combines memetic and tabu search algorithms to improve graph κ-partitions. LaSalle & Karypis [28] introduced and compared several approaches to parallelizing the multilevel phases in the bounded capacity graph partitioning problem. Lu et al. [30] introduced the first iterated multilevel simulated annealing algorithm for large-scale graph partitioning with the criterion of conductance minimization. Hausberger et al. [18] proposed and developed a memetic algorithm based on a novel multilevel approach for signed graph clustering. The multilevel approach has also been applied to other hard combinatorial optimization problems, such as the graph coloring problem [37], the traveling salesman problem [45], the clustered orienteering problem [19], etc.

To sum up, the multilevel approach is a general problem-solving strategy for solving large-scale optimization problems in graphs. Recent literature primarily addresses specific problems or proposes algorithms designed to improve particular phases, such as the matching algorithm required during the coarsening phase. The potential applicability of this powerful approach is huge, since it has been highly successful when applied to graph partitioning and many other challenging graph problems. However, it has not yet been investigated for solving MCGP. This work aims to fill the gap by presenting the first effective multilevel algorithm for MCGP that produces high-quality solutions in a reasonable amount of time for large graphs arising from real-world applications.

3. Multilevel iterated tabu search for the MCGP problem

3.1. The general framework

Following the general framework of multilevel optimization, as shown in Fig. 1, the proposed multilevel iterated tabu search (MITS) for the MCGP

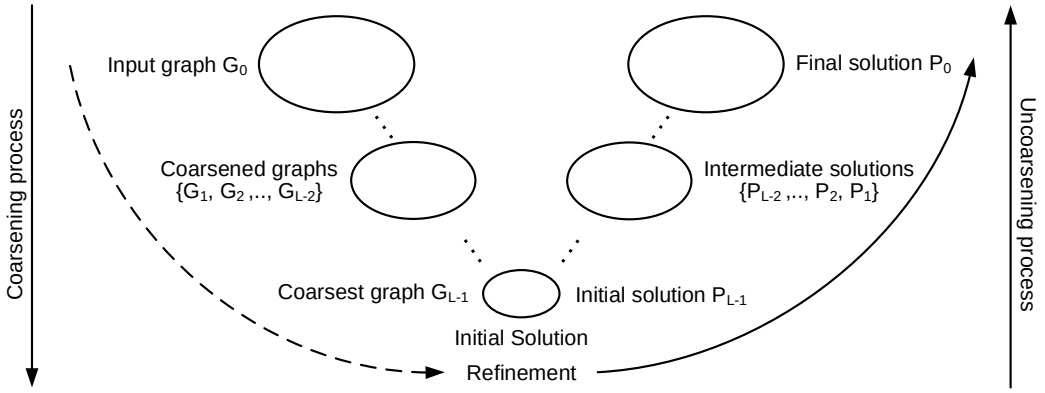

Figure 1: The general framework of the multilevel optimization approach.

problem consists of three main phases: 1) graph coarsening (including matching and contraction), 2) solution finding (including initial solution and refinement), and 3) graph uncoarsening (including projection and refinement). The main idea is to iteratively coarsen the input graph into a hierarchy of gradually smaller graphs until the coarsest graph is reached, solve the problem to obtain an initial solution in the coarsest graph, and then successively project this solution back to the intermediate graphs (the reverse sequence of coarsened graphs) and perform solution refinement at each hierarchy level until the original graph is recovered.

Specifically, the MITS algorithm combines a problem-specific coarsening method and a dedicated Iterated Tabu Search (ITS) refinement procedure. Algorithm 1 shows the general scheme of MITS. At first, the input graph is progressively coarsened level by level using the problem-specific coarsening method (lines 3-5, Section 3.3) to select and collapse vertices into 'larger' vertices, until the graph is reduced by the desired amount $\rho \times n$ (ρ is a parameter called the reduction factor). The coarsening process is repeated until the maximum number of coarsening levels, L, is reached. At this point, the coarsest graph, G_{L-1}, is produced. Then, an initial solution $\mathcal{P}_{L-1}$ on the graph G_{L-1} is created by using an initialization procedure followed by the ITS refinement procedure (lines 6-7, Sections 3.4 and 3.6). The uncoarsening phase then begins in reverse order, and the initial solution is projected along the sequence of intermediate graphs to create the intermediate solutions, denoted by $\{\mathcal{P}_{L-2}, ..., \mathcal{P}_2, \mathcal{P}_1\}$. The solution at each level is refined by ITS before moving to the next level. The uncoarsening phase stops when the

Algorithm 1: Multilevel iterated tabu search (MITS) for the multi-constraint graph partitioning (MCGP) problem

Input: Input graph $G_0 = (V_0, E_0)$, maximum number of coarsening levels L, reduction factor ρ, and the other inputs are related to the Iterated Tabu Search (shake strength β, search depth of tabu search D, tabu tenure tt, penalty coefficient α, and update frequency of the penalty factor λ).
Output: Best solution found $\mathcal{P}^*$

1 $l \leftarrow 0$
2 $\mathcal{P}^* \leftarrow \emptyset$
3 **while** $l < L - 1$ **do**
4 $\quad G_{l+1} \leftarrow$ Problem-specific_Coarsening(G_l, ρ) `// Section 3.3`
5 $\quad l \leftarrow l + 1$
6 $\mathcal{P}_l \leftarrow$ Initial_Solution$(G_l, \mathcal{P}_l)$ `// Section 3.4`
7 $\mathcal{P}_l \leftarrow$ Iterated_Tabu_Search$(G_l, \mathcal{P}_l, \beta, D, tt, \alpha, \lambda)$ `// Section 3.6`
8 **while** $l > 0$ **do**
9 $\quad (G_{l-1}, \mathcal{P}_{l-1}) \leftarrow$ Uncoarsening$(G_l, \mathcal{P}_l)$ `// Section 3.5`
10 $\quad \mathcal{P}_{l-1} \leftarrow$ Iterated_Tabu_Search$(G_{l-1}, \mathcal{P}_{l-1}, \beta, D, tt, \alpha, \lambda)$
11 $\quad l \leftarrow l - 1$
12 **if** $time() \leq t_{max}$ **then**
13 $\quad \mathcal{P}^* \leftarrow$ Iterated_Tabu_Search$(G_0, \mathcal{P}_0, \beta, D, tt, \alpha, \lambda)$
14 **return** $\mathcal{P}^*$

original graph G_0 is obtained with the associated final solution $\mathcal{P}_0$ (lines 8-11, Sections 3.5 and 3.6). The final solution $\mathcal{P}_0$ is further improved by ITS until the maximum time limit t_{max} is reached (lines 12-13).

This work shares a similarity with [3], which is dedicated to the κ-balanced graph partitioning problem, as both rely on the general multilevel optimization framework. However, the current work distinguishes itself by three notable features. First, we use a single-trajectory iterated local search for solution refinement, whereas in [3], a population-based memetic search is adopted. Second, our solution refinement explores both feasible and infeasible solution, whereas the search strategy in [3] is limited to feasible solutions only. Third, we adopt a problem-specific coarsening method that fully preserves the critical topological features of MCGP, which differs from the classic coarsening method used in [3]. In the following, we describe the MITS algorithm step by step, including the solution representation and evaluation, the problem-specific coarsening phase, the initial solution procedure, the uncoarsening phase, the ITS refinement procedure, and an analysis of its overall time complexity.

3.2. Solution representation and evaluation

For a given MCGP instance that is composed of a complete, undirected graph $G = (V, E)$, the number of partitions κ, the number of attributes τ, and the lower and upper bounds W_{kt}^L and W_{kt}^U, the MITS algorithm explores the search space Ω composed of all partitions of the vertex set V into κ nonempty, pairwise-disjoint subsets defined as follows.

$$\begin{aligned}\Omega = \{\mathcal{P} = \{S_1, ..., S_\kappa\} : \\ \cup_{k=1}^{\kappa} S_k = V, S_i \cap S_j = \emptyset, S_i, S_j \neq \emptyset, i \neq j, \forall i, j \in \{1, ..., \kappa\}\}\end{aligned} \tag{3}$$

Following [20], we first define, for a given solution $\mathcal{P} = \{S_1, ..., S_\kappa\}$, $\sigma(\mathcal{P}, S_k)$ $(k = 1, ..., \kappa)$ as the degree of constraint violations in terms of lower and upper bounds,

$$\sigma(\mathcal{P}, S_k) = \sum_{t=1}^{\tau} (\max\{W_{kt}^L - w(\mathcal{P}, S_k, t), 0\} + \max\{w(\mathcal{P}, S_k, t) - W_{kt}^U, 0\}) \tag{4}$$

which aggregates the deficit or excess relative to the lower and upper bounds of each attribute for all vertices belonging to the subset S_k in $\mathcal{P}$[1].

Then, the penalty function C is defined by,

$$C(\mathcal{P}) = \sum_{k=1}^{\kappa} \sigma(\mathcal{P}, S_k) \tag{5}$$

which summarizes the degrees of constraint violations over all subsets of the solution $\mathcal{P}$. Therefore, if $C(\mathcal{P}) = 0$, the solution is feasible and satisfies the multi-dimensional weight constraints. If $C(\mathcal{P}) > 0$, the solution includes at least one subset that violates the weight constraints, making it infeasible.

Following the general idea of the penalty-based evaluation function for constrained optimization, we introduce an extended evaluation function F^* to assess both feasible and infeasible solutions of Ω, which enriches the objective function F (Eq. (1)) with the penalty function C (Eq. (5)) as follows,

$$F^*(\mathcal{P}) = F(\mathcal{P}) + \phi \times C(\mathcal{P}) \tag{6}$$

[1]More generally, if the upper and lower bounds of the attributes are not defined on the same scale, then it will be more appropriate to use, in the definition of the constraint violation $\sigma(\mathcal{P}, S_k)$ of Eq. (4), a scale-invariant measure instead of the absolute differences.

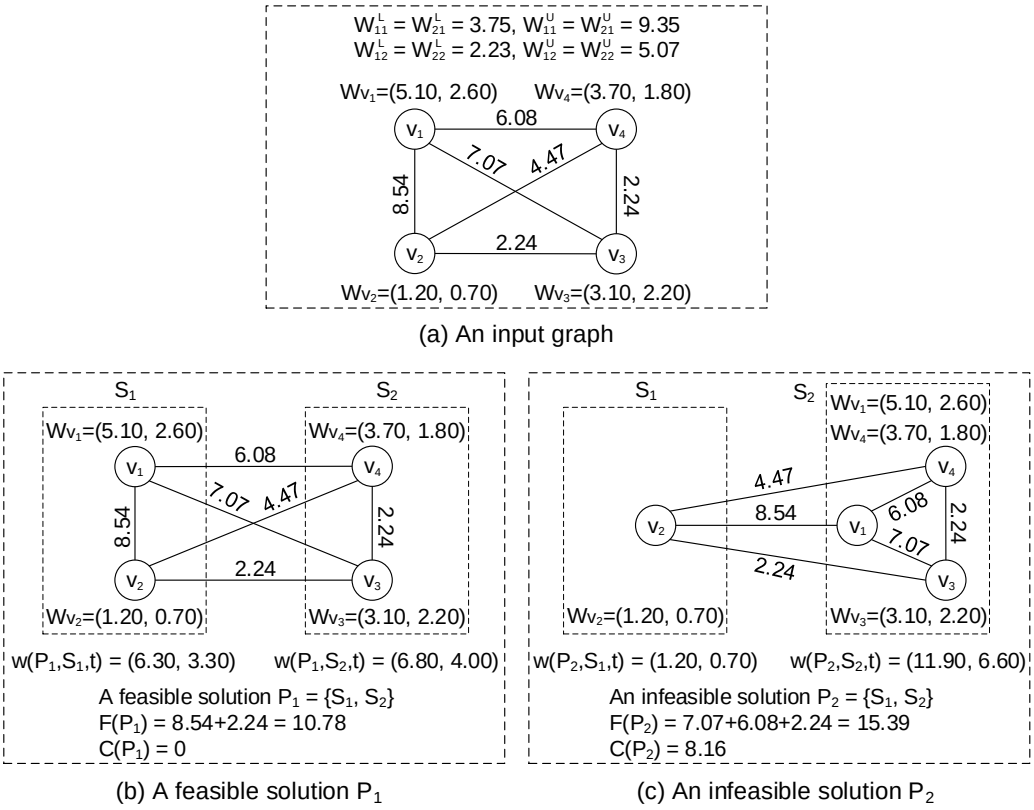

Figure 2: Illustration of MCGP with a feasible solution and an infeasible solution.

In other words, F^* is a linear combination of the objective function F and the penalty function C that evaluates both feasible and infeasible solutions, and $\phi \geq 1$ is the penalty factor that controls the relative importance given to C.

Since F^* is to be minimized, increasing ϕ augments the value of F^*, which makes the infeasible solution under consideration less attractive. On the other hand, decreasing ϕ lowers the value of F^*, which makes the infeasible solution more attractive. By varying ϕ, we can control the transition between the feasible and infeasible regions. Section 3.6.2 explains the adaptive technique that dynamically adjusts ϕ according to the search situation. We can assess the relative quality of two candidate solutions $\mathcal{P}$ and $\mathcal{P}'$ by F^* as follows: if $F^*(\mathcal{P}) < F^*(\mathcal{P}')$, then we say $\mathcal{P}$ is better than $\mathcal{P}'$.

A MCGP problem example with 4 vertices, 6 edges, 2 partitions, and 2

attributes is depicted in Fig. 2. Fig. 2 (a) shows an input graph with its costs (distances), attribute values, and attribute bounds. Figs. 2 (b) and (c) present a feasible solution $\mathcal{P}_1$ and an infeasible solution $\mathcal{P}_2$, respectively.

- For $\mathcal{P}_1 = \{S_1 = \{v_1, v_2\}, S_2 = \{v_3, v_4\}\}$, the weight of subset $S_1 \in \mathcal{P}_1$ on attribute 1 is $w(\mathcal{P}_1, S_1, 1) = 6.30$ satisfying the bounds constraints $W_{11}^L = 3.75$ and $W_{11}^U = 9.35$, and the weight of S_1 on attribute 2 is $w(\mathcal{P}_1, S_1, 2) = 3.30$ also within the bounds $W_{12}^L = 2.23$ and $W_{12}^U = 5.07$. The same observations hold for $S_2 \in \mathcal{P}_1$. Thus, $\mathcal{P}_1$ is a feasible solution and its objective function value $F(\mathcal{P}_1) = d(v_1, v_2) + d(v_3, v_4) = 8.54 + 2.24 = 10.78$.

- For $\mathcal{P}_2 = \{S_1 = \{v_2\}, S_2 = \{v_1, v_3, v_4\}\}$, the weight of subset $S_1 \in \mathcal{P}_2$ on attribute 1 is $w(\mathcal{P}_2, S_1, 1) = 1.20$ that violates the lower limit $W_{11}^L = 3.75$ but satisfies the upper limit $W_{11}^U = 9.35$, and the weight on attribute 2 is $w(\mathcal{P}_2, S_1, 2) = 0.70$ violating the lower limit $W_{12}^L = 2.23$. The degree of constraint violation for $S_1 \in \mathcal{P}_2$ is $\sigma(\mathcal{P}_2, S_1) = \max\{W_{11}^L - w(\mathcal{P}_2, S_1, 1), 0\} + \max\{w(\mathcal{P}_2, S_1, 1) - W_{11}^U, 0\} + \max\{W_{12}^L - w(\mathcal{P}_2, S_1, 2), 0\} + \max\{w(\mathcal{P}_2, S_1, 2) - W_{12}^U, 0\} = (3.75 - 1.20) + 0 + (2.23 - 0.70) + 0 = 4.08$. The same observations hold for $S_2 \in \mathcal{P}_2$ and the degree of violation for $S_2 \in \mathcal{P}_2$ is $\sigma(\mathcal{P}_2, S_2) = \max\{W_{21}^L - w(\mathcal{P}_2, S_2, 1), 0\} + \max\{w(\mathcal{P}_2, S_2, 1) - W_{21}^U, 0\} + \max\{W_{22}^L - w(\mathcal{P}_2, S_2, 2), 0\} + \max\{w(\mathcal{P}_2, S_2, 2) - W_{22}^U, 0\} = 0 + (11.90 - 9.35) + 0 + (6.60 - 5.07) = 4.08$. Thus, $\mathcal{P}_2$ is an infeasible solution with its penalty function value $C(\mathcal{P}_2) = \sigma(\mathcal{P}_2, S_1) + \sigma(\mathcal{P}_2, S_2) = 4.08 + 4.08 = 8.16$ and its objective function value $F(\mathcal{P}_2) = d(v_1, v_3) + d(v_1, v_4) + d(v_3, v_4) = 7.07 + 6.08 + 2.24 = 15.39$.

3.3. Problem-specific coarsening

The coarsening phase begins with an input graph G_0 and creates a sequence of progressively coarsened graphs G_l, which are intermediate approximations of G_0 with gradually decreasing levels of detail. The size of the graph decreases at each level such that $|V_l| > |V_{l+1}|$, until the maximum number of coarsening levels L is reached ($l = 0, 1, ..., L - 2$). The coarsening process typically consists of two main steps: a matching step to identify vertices for merging and a contraction step to combine them. Ideally, each coarsened graph should preserve the critical topological (problem-specific) properties of its predecessor. Several methods have been proposed for this purpose, which are mainly divided into edge-selection matching [23, 25, 14, 12], vertex-selection matching [44, 45, 42, 6], and cluster-selection matching [1, 7].

In this work, we propose a problem-specific coarsening method that fully preserves the critical topological features of MCGP. The matching step aims to find an independent edge set $M \in E_l$ such that no two edges in M are adjacent. Following that, we sort the edge set E_l in non-descending order based on the cost of each edge d_l, denoted as E'_l. We examine the edges $e \in E'_l$ to find matching edges that satify 1) the two vertices v and u connected by edge e are not already matched, and 2) the total weight $W_{v+u} = W_v + W_u$ of the merged vertices $v+u$ does not exceed the upper bound W^U_{kt} ($\forall k \in K, \forall t \in T$). Otherwise, we move on to the next edge in E'_l. The process repeats until the size of the coarsened graph G_l reaches $\rho \times n$ (ρ is a parameter called the reduction factor). Then, the contraction step collapses the endpoints of each edge $\{v, u\} \in M$ into a new vertex $v + u$ in the coarsened graph G_{l+1}, and vertices not contained in M are directly inherited by G_{l+1}. The weight of the new vertex $v + u \in V_{l+1}$ is set to the sum of the weights of v and u. The edge between v and u is removed, and the edges incident on both v and u are merged to form a new edge in E_{l+1} with a weight set to the sum of the weights of the merged edges.

Fig. 3 shows an example of the coarsening process from the input graph G_0 with 6 vertices, 15 edges, 2 partitions, and 2 attributes (Fig. 3 (a)) to the coarsened graph G_1 (Fig. 3 (d)). First, we sort the edge set E_0 in non-descending order and identify the matching set $M = \{\{v_3, v_6\}, \{v_2, v_5\}, \{v_1, v_4\}\}$. Then, as shown in Fig. 3 (b), we merge v_3 and v_6 to form v'_3 with vertex weight $W_{v'_3} = W_{v_3} + W_{v_6} = (6.10, 6.40)$ that does not exceed the upper bounds $W^U_{k1} = 11.37$ and $W^U_{k2} = 11.74$, $\forall k \in K$. The edge $\{v_3, v_6\}$ is removed, while the edges $\{v_3, v_1\}$ and $\{v_6, v_1\}$, $\{v_3, v_2\}$ and $\{v_6, v_2\}$, $\{v_3, v_4\}$ and $\{v_6, v_4\}$, and $\{v_3, v_5\}$ and $\{v_6, v_5\}$ which are incident to both v_3 and v_6 are merged to form the new edges $\{v'_3, v_1\}$, $\{v'_3, v_2\}$, $\{v'_3, v_4\}$, and $\{v'_3, v_5\}$ in G_1 with edge weights $d(v'_3, v_1) = d(v_3, v_1) + d(v_6, v_1) = 12.17$, $d(v'_3, v_2) = d(v_3, v_2) + d(v_6, v_2) = 5.85$, $d(v'_3, v_4) = d(v_3, v_4) + d(v_6, v_4) = 4.48$, and $d(v'_3, v_5) = d(v_3, v_5) + d(v_6, v_5) = 8.26$. The same operations are performed to merge the vertices v_2 and v_5, and v_1 and v_4. Finally, we obtain the coarsened graph G_1 shown in Fig. 3 (d).

This method is fast, taking only $\mathcal{O}(|E_l|)$ time in the worst case, and is also problem-specific for the following reasons. First, the coarsening process sorts the edges by weight, meaning the lighter ones are folded more frequently than the heavier ones. Therefore, good solutions are easier to obtain during the uncoarsening process since unfolding lighter edges does not significantly deteriorate the quality of the solution. Second, the matching step guarantees

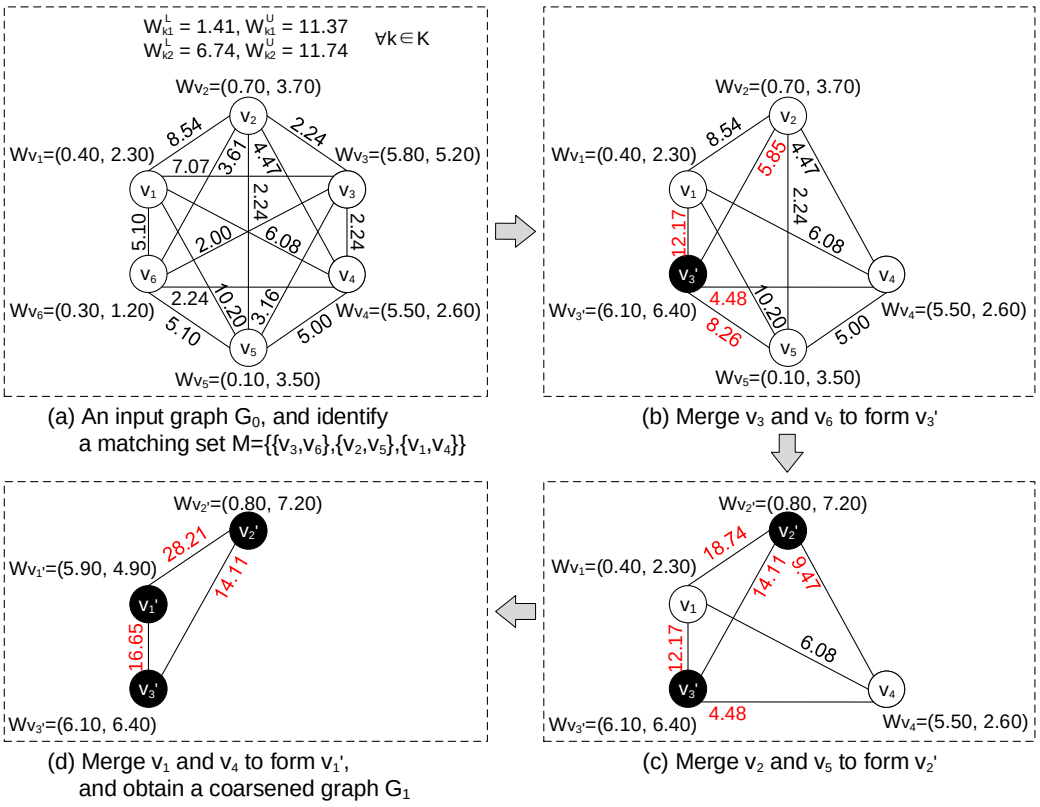

Figure 3: Illustration of the problem-specific coarsening process.

that the merged vertices meet their specified upper bounds. Third, MCGP comes from a real-world application involving monthly polls conducted at a fixed set of stores to monitor consumer price behavior for basic commodities. The stores are connected in pairs, which allows for aggressive reduction rates by identifying many non-adjacent edges and leaving few unmatched vertices.

3.4. Initial solution

MITS uses a simple and fast random construction method to obtain an initial solution $\mathcal{P}_{L-1}$ in G_{L-1} in two steps, 1) we randomly assign κ vertices to κ empty subsets, and 2) we assign the remaining vertices one by one to a random subset $k \in K$ until all vertices are assigned to exactly one subset. This solution $\mathcal{P}_{L-1}$ is then used as the initial solution of MITS. As shown in Sections 4.4 and 4.5, even when starting from a random initial solution, MITS

can find solutions that are much better than the state-of-the-art algorithms. It is important to note that the generated initial solution is not guaranteed to be feasible. Additionally, the subsequent uncoarsening process progressively projects the solution level by level, which may result in constraint violations. This situation motivated us to design a refinement procedure that alternates between feasible and infeasible solutions (Section 3.6).

3.5. Uncoarsening

In the uncoarsening phase, we start with the initial solution $\mathcal{P}_{L-1}$ found in G_{L-1}, and improve it using the Iterated Tabu Search (ITS) refinement procedure (Section 3.6). Then, we progressively transfer this solution to the inverse sequence of coarsened graphs. Specifically, the starting solution is gradually projected and improved by ITS on the intermediate graphs $G_{l-1}, G_{l-2}, \ldots, G_1$ until it reaches the original graph G_0. An intermediate solution $\mathcal{P}_{l-1}$ is created from $\mathcal{P}_l$ by assigning the vertices in V_{l-1} to the same subset as their successor in V_l. During each level of the uncoarsening phase, the current solution $\mathcal{P}_l$, derived from graph G_l, is projected onto graph G_{l-1}, $l = 1, 2, ..., L-1$. In general, the quality of the solution progressively improves during the uncoarsening process because more degrees of freedom are available to ITS in an uncoarsened graph.

3.6. Iterated tabu search refinement

The proposed Iterated Tabu Search (ITS) is the key refinement procedure in our MITS algorithm, which is applied at each level during the uncoarsening phase to improve the projected solution from the precedent level. As shown in Algorithm 2, ITS iterates a process of three steps (lines 3-14). First, it performs an adaptive feasible and infeasible tabu search (AFITS) that switches the search back and forth between feasible and infeasible areas (line 5, Section 3.6.2), followed by a feasible tabu search (FTS) to further improve the quality of the solution (line 7). To explore candidate solutions effectively, four move operators are used including two basic operators (*Insert* and *Swap*) proposed in [20] and two new operators (*Push* and *Relocate*) along with a fast incremental evaluation technique. Third, ITS applies an informed greedy randomized perturbation to help the search escape local optima (line 14, Section 3.6.3). During the search process, the recorded best found solution $\mathcal{P}^*$ is updated each time an improved best solution is found (lines 8-12). ITS stops and returns $\mathcal{P}^*$ when a feasible solution cannot be found or the maximum time limit t_{max} is reached (line 15).

Algorithm 2: Iterated Tabu Search (ITS) refinement

Input: Graph $G = (V, E)$, input solution $\mathcal{P}$, shake strength β, search depth of tabu search D, tabu tenure tt, and penalty coefficient α, update frequency of the penalty factor λ.
Output: Best solution found $\mathcal{P}^*$

```
P* ← P                                   // P* records the best solution found so far
flag ← true
while flag = true ∧ time() < t_max do
    flag ← false
    P ← Adaptive_Feasible_Infeasible_TS(F*, P, D, tt, α, λ)        // Section 3.6.2
    if C(P) = 0 then
        P ← Feasible_TS(F, P, D, tt)                               // Section 3.6.2
        if F(P) < F(P*) then
            flag ← true
            P* ← P
    else if F(P) < F(P*) ∧ C(P) < C(P*) then
        P* ← P
    if flag = true then
        P ← Informed_Greedy_Randomized_Perturb(P, β)               // Section 3.6.3
return P*
```

Next, we explain the neighborhood structures with the fast incremental evaluation technique, the tabu search procedures, and the informed greedy randomized perturbation.

3.6.1. Neighborhood structures

ITS iteratively transforms the incumbent solution into a neighboring solution by applying move operations. Typically, a move operation (or simply a move) changes the solution by moving one or more vertices to one or more new subsets. Formally, let $\mathcal{P}$ be the incumbent solution and let mv be a move. We use $\mathcal{P}' \leftarrow \mathcal{P} \oplus mv$ to denote the neighboring solution $\mathcal{P}'$ obtained by applying mv to $\mathcal{P}$. For each move operation mv, we define the move value $\triangle_f$ as the change in function value f between the incumbent solution $\mathcal{P}$ and the neighboring solution $\mathcal{P}'$ obtained by applying the move,

$$\triangle_f(mv) = f(\mathcal{P}') - f(\mathcal{P}) \tag{7}$$

where f is either the objective function F or the extended evaluation function F^* according to whether the considered solution is feasible or infeasible.

Common move operations for partitioning problems involve jointly considering the relocation of one vertex and the exchange of two vertices. These correspond to the *Insert* and *Swap* moves used in the previous study [20]. From these two basic move operators, we define two new search operators *Push* and *Relocate* to further improve exploration efficiency. Thus, ITS thoroughly explores the search space with these four distinct move operators. From a current solution $\mathcal{P} = \{S_1, ..., S_\kappa\}$, the four move operators are illustrated in Fig. 4 and described below.

- ***Insert* operator**: Given a vertex $v \in S_i$ and a target subset S_j ($i \neq j$, $i, j \in K$), the *Insert* move displaces v from S_i to S_j (denoted as $Insert(v, S_i, S_j)$). The set of solutions that form the neighborhood for this move is defined as,

$$N_{Insert}(\mathcal{P}) = \{\mathcal{P}' \leftarrow \mathcal{P} \oplus Insert(v, S_i, S_j) : v \in S_i, i \neq j, i, j \in K\} \quad (8)$$

- ***Swap* operator**: Given two vertices $v \in S_i$, $u \in S_j$ ($v \neq u$, $i \neq j$, $i, j \in K$), the *Swap* move displaces v from S_i to S_j, and u from S_j to S_i, i.e., exchanging two vertices with each other (denoted as $Swap(v, u, S_i, S_j)$). The set of solutions that form the neighborhood for this move is defined as,

$$\begin{aligned} N_{Swap}(\mathcal{P}) = \{\mathcal{P}' \leftarrow \mathcal{P} \oplus Swap(v, u, S_i, S_j) : \\ v \in S_i, u \in S_j, v \neq u, i \neq j, i, j \in K\} \end{aligned} \quad (9)$$

- ***Push* operator**: Given two vertices $v \in S_i$, $u \in S_j$ ($v \neq u$, $i \neq j$, $i, j \in K$). The second vertex u is inserted into a new subset S_l, $l \in K$, $l \neq i \neq j$, and the first vertex v is inserted into the original subset of the second vertex S_j (denoted as $Push(v, u, S_i, S_j, S_l)$). It can be said that "a vertex v has pushed a vertex u into the new subset S_l". The set of solutions that form the neighborhood for this move is defined as,

$$\begin{aligned} N_{Push}(\mathcal{P}) = \{\mathcal{P}' \leftarrow \mathcal{P} \oplus Push(v, u, S_i, S_j, S_l) : \\ v \in S_i, u \in S_j, v \neq u, i \neq j \neq l, i, j, l \in K\} \end{aligned} \quad (10)$$

- ***Relocate* operator**: Given three vertices $v \in S_i$, $u \in S_j$, and $g \in S_l$ ($v \neq u \neq g$, $i \neq j \neq l$, $i, j, l \in K$). The third vertex g is inserted

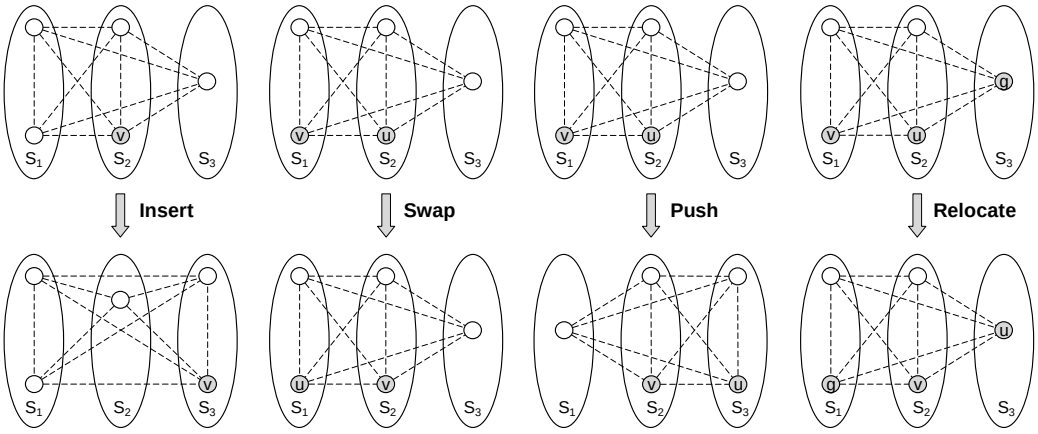

Figure 4: Illustration of the *Insert*, *Swap*, *Push*, and *Relocate* move operators.

into the original subset of the first vertex S_i, the second vertex u is inserted into the original subset of the third vertex S_l, and the first vertex v is inserted into the original subset of the second vertex S_j (denoted as $Relocate(v, u, g, S_i, S_j, S_l)$). The set of solutions that form the neighborhood for this move is defined as,

$$N_{Relocate}(\mathcal{P}) = \{\mathcal{P}' \leftarrow \mathcal{P} \oplus Relocate(v, u, g, S_i, S_j, S_l) : v \in S_i, \\ u \in S_j, g \in S_l, v \neq u \neq g, i \neq j \neq l, i, j, l \in K\} \quad (11)$$

To ensure a high computation efficiency of exploring these neighborhoods, we use a fast incremental evaluation technique for these search operators, which is described in detail in Appendix A.

ITS explores these neighborhoods in combination. The idea is to evaluate improvements (move values) by exploring all possible moves resulting from the basic *Insert* and *Swap* operations, and then to quickly evaluate improvements by exploring some moves from the new *Push* and *Relocate* operations that might be 'missed' by the basic *Insert* and *Swap* operations. First, we evaluate all possible *Insert* moves for a vertex $v \in V$ from its current subset S_i to another subset S_j, and then all possible *Swap* moves that involve exchanging $v \in S_i$ and $u \in S_j$ ($S_i \neq S_j$) with the best improvement strategy. Next, we shuffle the vertices in V to generate the shuffle set V_{shuffle}. We sequentially select two vertices $v, u \in V_{\text{shuffle}}$ where $v \in S_i$ and $u \in S_j$ ($i \neq j$)

Table 1: Summary of the complexity of the neighborhood related to the four move operators *Insert*, *Swap*, *Push*, and *Relocate*.

Neighborhood	Size	Function	Move evaluation	Update
N_{Insert}	$\mathcal{O}(n \times \kappa)$	F^* F	$\mathcal{O}(n \times \kappa \times \tau)$	$\mathcal{O}(n+\tau)$ $\mathcal{O}(n)$
N_{Swap}	$\mathcal{O}(n^2)$	F^* F	$\mathcal{O}(n^2 \times \tau)$	$\mathcal{O}(n+\tau)$ $\mathcal{O}(n)$
N_{Push}	$\mathcal{O}(n \times \kappa)$	F^* F	$\mathcal{O}(n \times \kappa \times \tau)$	$\mathcal{O}(n+\tau)$ $\mathcal{O}(n)$
$N_{Relocate}$	$\mathcal{O}(n)$	F^* F	$\mathcal{O}(n \times \tau)$	$\mathcal{O}(n+\tau)$ $\mathcal{O}(n)$

and a new subset $S_l \in \mathcal{P}$ for the *Push* evaluation and then sequentially select three vertices $v, u, g \in V_{\text{shuffle}}$ where $v \in S_i$, $u \in S_j$, and $g \in S_l$ $(i \neq j \neq l)$ for the *Relocate* evaluation, all of them with the best improvement strategy. The move with the minimum move value from these evaluations is our best neighboring move (ties are randomly broken). Thus, we formalize the maximum improvement with respect to the function value f as follows,

$$\begin{aligned} BestMove(\mathcal{P}, f) = \min\{&\Delta_f(Insert(v, S_i, S_j)), \Delta_f(Swap(v, u, S_i, S_j)), \\ &\Delta_f(Push(v, u, S_i, S_j, S_l)), \Delta_f(Relocate(v, u, g, S_i, S_j, S_l))\} \end{aligned} \tag{12}$$

where f is either F or F^* according to whether the evaluated solution $\mathcal{P}$ is feasible or infeasible. The notation $BestMove(\mathcal{P}, f)$ is used as the function that returns the best admissible neighboring solution $\mathcal{P}'$ with the maximum improvement with respect to f.

Table 1 provides a summary of the complexity of each neighborhood, which includes its size and the complexity of evaluating the move and updating the associated data structures δ (Eq. (A.1)) and w (Section 1) after the move is performed. As shown in Table 1, the time complexity of evaluating the solution quality and updating the data structures after each move is bounded by $\mathcal{O}(n \times \tau \times (n + \kappa))$ with respect to F and F^*.

3.6.2. Adaptive feasible and infeasible tabu search

The algorithm of [20] uses the penalty function C separately to find a feasible solution, and then applies the objective function F to explore the feasible solutions. Unlike this approach, we adopt the strategic oscillation

Algorithm 3: Adaptive feasible and infeasible tabu search (AFITS)

Input: Graph $G = (V, E)$, input solution $\mathcal{P}$, search depth of tabu search D, tabu tenure tt, penalty coefficient α, update frequency of the penalty factor λ.
Output: Best solution found $\mathcal{P}^*$

$\mathcal{P}^* \leftarrow \mathcal{P}$ // $\mathcal{P}^*$ records the best solution found so far
$iter \leftarrow 0$ // $iter$ counts the number of non-improving iterations
$X \leftarrow 0, Y \leftarrow 0$ // X (Y) records the consecutive number of feasible (infeasible) solutions found
$\phi \leftarrow 1$ // ϕ is the penalty factor
while $iter < D$ **do**
    $\mathcal{P}' \leftarrow BestMove(\mathcal{P}, F^*)$ // Choose a best admissible neighboring solution in terms of F^* (Eq. (6))
    $\mathcal{P} \leftarrow \mathcal{P}'$ // Update the current solution
    Update the tabu list with tt
    if $C(\mathcal{P}) = 0$ **then**
        if $F(\mathcal{P}) < F(\mathcal{P}^*)$ **then**
            $\mathcal{P}^* \leftarrow \mathcal{P}$
            $iter \leftarrow 0$
        $Y \leftarrow 0$
        $X \leftarrow X + 1$
    else
        if $F(\mathcal{P}) < F(\mathcal{P}^*) \wedge C(\mathcal{P}) < C(\mathcal{P}^*)$ **then**
            $\mathcal{P}^* \leftarrow \mathcal{P}$
        $iter \leftarrow iter + 1$
        $X \leftarrow 0$
        $Y \leftarrow Y + 1$
    // Adaptive adjustment mechanism for penalty factor ϕ
    if $X \ \% \ \lambda = 0 \wedge X \neq 0$ **then**
        $\phi \leftarrow \phi / \alpha$
    else if $Y \ \% \ \lambda = 0 \wedge Y \neq 0$ **then**
        $\phi \leftarrow \phi \times \alpha$
    if $\phi < 1$ **then**
        $\phi \leftarrow 1$
return $\mathcal{P}^*$

approach [15] and investigate for the first time the benefits of dynamically examining both feasible and infeasible solutions to explore the search space. The main idea is to enable the search to move back and forth between feasible and infeasible areas by relaxing the weight constraints on each subset in a controlled manner. Specifically, we use the extended evaluation function F^* (Eq. (6)) that combines the objective function F and the penalty function C to evaluate the quality of both feasible and infeasible solutions.

The adaptive feasible and infeasible tabu search (AFITS) explores the four neighborhoods N_{Insert}, N_{Swap}, N_{Push}, and $N_{Relocate}$ in a union way with respect to F^*. To quickly compute the move value of a candidate move, AFITS uses the fast incremental evaluation technique described in Appendix A and defined by,

$$\triangle_{F^*}(mv) = \triangle_F(mv) + \phi \times \triangle_C(mv) \tag{13}$$

The general scheme of AFITS is presented in Algorithm 3. AFITS iteratively replaces the current solution $\mathcal{P}$ with the best admissible neighboring solution $\mathcal{P}'$ (ties are randomly broken), in terms of F^* from the previously defined neighborhood (denoted by $BestMove(\mathcal{P}, F^*)$) until a stopping condition is met, i.e., the best solution found so far is not updated during D consecutive iterations (D is a parameter called the search depth of tabu search) (lines 5-26). To prevent the search from short-term cycling, each time a vertex $v \in V$ is removed from its original subset S_i ($i \in K$), it cannot be put back into S_i for the next tt iterations (tt is a parameter called the tabu tenure) (line 8). However, we always perform a move if it leads to a solution that is better than all previously visited solutions (the aspiration criterion). The best feasible solution found $\mathcal{P}^*$ is updated by $\mathcal{P}$ if it has a better quality than $\mathcal{P}^*$ in terms of F^* (lines 10-11). During the search process, the search trajectory is dynamically adjusted by varying the penalty factor ϕ of F^*. Specifically, starting with the value of 1 (line 4), ϕ is divided (or multiplied) by α if the last λ solutions are feasible (or infeasible) (α and λ are two parameters called the penalty coefficient and the update frequency of the penalty factor, respectively) (lines 21-26). A large value of ϕ strongly penalizes infeasible solutions and gives more importance to feasible solutions. A small value of ϕ leads to the opposite effect. The adaptive adjustment mechanism with dynamic ϕ values allows the search process to switch between feasible and infeasible areas to achieve an appropriate balance between intensification and diversification. Once a feasible solution is found during AFITS, we trigger

the feasible tabu search (FTS) over the objective function F, which aims to further improve the quality of the feasible solution (line 7, Algorithm 2).

3.6.3. Informed greedy randomized perturbation

Each time a new feasible solution that is better than the best recorded solution is found during the ITS procedure (lines 8-10, Algorithm 2), a new deep local optimum is reached. To help ITS escape the trap, a perturbation method is applied to diversify the search. Inspired by the long-term memory mechanism adopted in [11], we propose an informed greedy randomized perturbation method to modify the last local optimum, which will serve as the starting solution for the next round of ITS (lines 13 and 14, Algorithm 2). To make an informed perturbation, we use information on vertex move frequency collected during the last AFITS and FTS runs. Specifically, we identify the $\beta \times |V_l|$ ($l = 0, 1, ..., L-1$) least frequently moved vertices (β is a parameter called the shake strength), and force them to move to a randomly selected new subset. In this way, the perturbation focuses on "hard to move" vertices and creates opportunities to overcome deep local optimum traps that would be difficult to escape otherwise.

3.7. Complexity analysis

To analyze the computation complexity of MITS, we first consider the ITS refinement procedure (Algorithm 2) of MITS. For each round of ITS, we sequentially run AFITS, FTS, and the informed greedy randomized perturbation procedure. AFITS and FTS each take $\mathcal{O}(D \times n \times \tau \times (n+\kappa))$ time to run over the extended evaluation function F^* and the objective function F, respectively. The perturbation runs with a time complexity of $\mathcal{O}(n \times \log n)$. Thus, the total time complexity of ITS is bounded by $\mathcal{O}(D \times n \times \tau \times (n+\kappa))$.

Then, we consider three other main components of MITS (Algorithm 1). MITS starts the search with a problem-specific coarsening method with a time complexity of $\mathcal{O}(L \times n)$. The initial solution can be obtained in $\mathcal{O}(n)$ time, and the uncoarsening procedure takes a total of $\mathcal{O}(1)$ time, since it only restores each corresponding intermediate graph recorded during the coarsening process. Thus, the total time complexity of MITS is bounded by $\mathcal{O}(D \times n \times \tau \times (n+\kappa))$.

4. Experimental results

We conduct extensive computational experiments of the proposed MITS algorithm on two sets of benchmark instances, and provide comparisons with

Table 2: INEC benchmark instances.

Name	n	κ	τ	Name	n	κ	τ
muestra1_20_3	20	3	2	muestra4_40_5	40	5	2
muestra2_20_3	20	3	2	muestra5_40_5	40	5	2
muestra3_20_3	20	3	2	muestra1_50_6	50	6	2
muestra4_20_3	20	3	2	muestra2_50_6	50	6	2
muestra5_20_3	20	3	2	muestra3_50_6	50	6	2
muestra1_30_4	30	4	2	muestra4_50_6	50	6	2
muestra2_30_4	30	4	2	muestra5_50_6	50	6	2
muestra3_30_4	30	4	2	muestra1_55_6	55	6	2
muestra4_30_4	30	4	2	muestra2_55_6	55	6	2
muestra5_30_4	30	4	2	muestra3_55_6	55	6	2
muestra1_40_5	40	5	2	muestra4_55_6	55	6	2
muestra2_40_5	40	5	2	muestra5_55_6	55	6	2
muestra3_40_5	40	5	2				

Table 3: TSPLib benchmark instances with $\kappa \in \{5, 10, 15, 20\}$ and $\tau \in \{2, 3, 4, 5\}$ for each of the 40 TSP graphs.

Name	n	Name	n	Name	n
kroA100	100	pr144	144	gr229	229
kroB100	100	ch150	150	gil262	262
kroC100	100	kroA150	150	pr264	264
kroD100	100	kroB150	150	a280	280
rd100	100	pr152	152	pr299	299
eil101	101	u159	159	lin318	318
lin105	105	rat195	195	rd400	400
pr107	107	d198	198	fl417	417
gr120	120	kroA200	200	gr431	431
pr124	124	kroB200	200	pr439	439
bier127	127	gr202	202	pcb442	442
ch130	130	ts225	225	d493	493
pr136	136	tsp225	225		
gr137	137	pr226	226		

the best-performing algorithms in the literature.

4.1. Benchmark instances

We use the following two sets of 665 benchmark instances introduced in [33, 20].

- **INEC set** contains 25 small instances with $n = \{20, 30, 40, 50, 55\}$, $\kappa = \{3, 4, 5, 6\}$, and $\tau = 2$. These instances are examples of real-world applications that arise in the National Statistics Bureau of Ecuador (INEC) [17, 33]. Each instance is a random sample of n points from a set of 820 stores. The vertices correspond to the locations of the stores, and the distance function corresponds to the walking travel

time between them. Two attributes are considered: the working time for data collection and the waiting time spent by pollsters in queues at each selected store.

- **TSPLib set** contains 640 large instances derived from 40 selected graphs with $n \in [100, 500)$ for the Symmetric Traveling Salesman Problem in the TSP Library [40]. Each graph has 4 different values of $\kappa = \{5, 10, 15, 20\}$ and $\tau = \{2, 3, 4, 5\}$, respectively, resulting in a total of 640 instances. These instances are generated using the same procedure as the INEC set and correspond to complex scenarios of the real-world problem described above.

The main features of these two sets of instances are summarized in Tables 2 and 3, which show the name (*Name*), the number of vertices (n), the number of partitions (κ), and the number of attributes (τ) of each instance.

4.2. Experimental settings

The MITS algorithm was implemented in C++[2], and compiled by g++ 12.2.0 using the optimization flag '-O3' option. All computational experiments were conducted on a computer with an Intel E5-2670 processor (2.80GHz) and 4GB of RAM, running the Linux operating system. To evaluate the performance of MITS, we compared them with two state-of-the-art methods from the literature: 1) IP, the integer programming based approach proposed in [33] and recently solved using `Gurobi 9.1.2` with a cutoff time of 2,000 seconds in [20], and 2) R-GRASP, the two-phase reactive GRASP heuristic algorithm [20] whose code was kindly made available to us by the authors. The numerical results of IP with `Gurobi 9.1.2` were directly taken from [20]. Due to the stochastic nature of heuristic algorithms, we independently ran R-GRASP and MITS 10 times to solve each instance on the same computing platform mentioned above, using their default parameter settings and a time limit as the unique stopping criterion. The time limit t_{max} is set to 10 seconds for instances with $n \in [0, 100)$, 60 seconds for instances with $n \in [100, 200)$, 300 seconds for instances with $n \in [200, 400)$, and 1,500 seconds for instances with $n \in [400, 500)$.

[2]The code of our MITS algorithm will be made publicly available at: `https://github.com/hellozhilu/MITS`, upon publication of the paper.

Table 4: Settings of important parameters.

Parameter	Description	Component	Calibrated values	Value
L	Maximum number of coarsening levels	Multilevel	$\{5, 6, ..., 9\}$	7
ρ	Reduction factor	Multilevel	$\{1\%, 2\%, ..., 5\%\}$	4%
β	Shake strength	ITS	$\{0.1, 0.2, ..., 0.5\}$	0.1
D	Search depth of tabu search	ITS	$\{0.5, 0.6, ..., 0.9\} \times n$	$0.8 \times n$
tt	Tabu tenure	ITS	$\{0.5, 0.6, ..., 0.9\} \times n$	$0.9 \times n$
α	Penalty coefficient	ITS	$\{1.0, 1.5, ..., 3.0\}$	2.0
λ	Update frequency of the penalty factor	ITS	$\{5, 10, ..., 25\}$	5

4.3. Parameter tuning

The MITS algorithm requires 7 parameters: L, ρ, β, D, tt, α, and λ. Specifically, L and ρ are related to multilevel, where L is the maximum number of coarsening levels and ρ is the reduction factor. The other parameters are related to ITS: shake strength β, search depth of tabu search D, tabu tenure tt, penalty coefficient α, and update frequency of the penalty factor λ. These parameters were tuned using the automatic tuning package `irace` [29], which is based on an iterative racing procedure designed for offline parameter configuration of parameterized algorithms. The 68 randomly selected instances used for the tuning experiment are listed in Table B.1 of Appendix B. The range of the candidate parameter values and the final values obtained by `irace` are shown in Table 4. The final values are used in all experiments reported in the paper and form the default parameter setting of the MITS algorithm.

4.4. Computational results and comparisons on small INEC set

Table 5 presents the computational results on the 25 small instances from the INEC set obtained by the MITS algorithm, the IP-based approach [33], and the R-GRASP algorithm [20]. Columns 1-4 show the characteristics of the instances, including their name ($Name$), the number of vertices (n), the number of partitions (κ), and the number of attributes (τ). For the single-run approach IP, we report the best result ($Best$) and the runtime in seconds ($t(s)$). For the multi-run algorithms R-GRASP and MITS, we report the best ($Best$) and average (Avg) results over 10 independent runs, as well as the average computation time in seconds to reach the best solution value for each run ($t(s)$). The best of the $Best$ (Avg) values among the compared algorithms for each instance is highlighted in boldface. Additionally, the 'Average' row shows the average results across the 25 instances. '#Wins'

Table 5: Detailed comparison results of the proposed MITS algorithm with the IP-based approach [33] and the state-of-the-art R-GRASP algorithm [20] on the 25 small instances from the INEC set.

Instance				IP [33]		R-GRASP [20]			**MITS**		
Name	n	κ	τ	*Best*	$t(s)$	*Best*	*Avg*	$t(s)$	*Best*	*Avg*	$t(s)$
muestra1_20_3	20	3	2	**1934.69***	1.68	**1934.69***	**1934.69**	0	**1934.69***	**1934.69**	0
muestra2_20_3	20	3	2	**1756.26***	2.99	**1756.26***	**1756.26**	0	**1756.26***	**1756.26**	0.14
muestra3_20_3	20	3	2	**1364.55***	1.38	**1364.55***	**1364.55**	0	**1364.55***	**1364.55**	0.08
muestra4_20_3	20	3	2	**2051.81***	1.59	**2051.81***	**2051.81**	0	**2051.81***	**2051.81**	0.36
muestra5_20_3	20	3	2	**1019.87***	0.72	**1019.87***	**1019.87**	0	**1019.87***	**1019.87**	0
muestra1_30_4	30	4	2	**3324.67***	56.14	**3324.67***	**3324.67**	0.01	**3324.67***	**3324.67**	0.04
muestra2_30_4	30	4	2	**2242.02***	124.84	**2242.02***	**2242.02**	0.01	**2242.02***	**2242.02**	0.11
muestra3_30_4	30	4	2	**3081.71***	414.53	**3081.71***	**3081.71**	0.01	**3081.71***	**3081.71**	0.69
muestra4_30_4	30	4	2	**2440.49***	69.23	**2440.49***	**2440.49**	0	**2440.49***	**2440.49**	0.25
muestra5_30_4	30	4	2	**1909.96***	193.11	**1909.96***	**1909.96**	0.01	**1909.96***	**1909.96**	0
muestra1_40_5	40	5	2	**3752.90**	-	**3752.90**	**3752.90**	0.17	**3752.90**	**3752.90**	0.03
muestra2_40_5	40	5	2	**3017.60**	-	**3017.60**	**3017.60**	0.01	**3017.60**	**3017.60**	0.01
muestra3_40_5	40	5	2	**3342.41**	-	**3342.41**	**3342.41**	0.01	**3342.41**	**3342.41**	0.13
muestra4_40_5	40	5	2	**4311.32**	-	**4311.32**	**4311.32**	0.03	**4311.32**	**4311.32**	0.16
muestra5_40_5	40	5	2	**2661.67**	-	**2661.67**	**2661.67**	0.01	**2661.67**	**2661.67**	0
muestra1_50_6	50	6	2	**4000.54**	-	**4000.54**	**4000.54**	0.15	**4000.54**	**4000.54**	0.02
muestra2_50_6	50	6	2	4479.51	-	**4446.16**	**4446.16**	0.57	**4446.16**	**4446.16**	0.01
muestra3_50_6	50	6	2	**3730.50**	-	**3730.50**	**3730.50**	0.02	**3730.50**	**3730.50**	0.20
muestra4_50_6	50	6	2	5031.38	-	**5021.54**	**5021.54**	0.65	**5021.54**	**5021.54**	2.31
muestra5_50_6	50	6	2	**4046.14**	-	**4046.14**	**4046.14**	0.02	**4046.14**	**4046.14**	0
muestra1_55_6	55	6	2	4706.92	-	**4693.65**	**4693.65**	1.41	**4693.65**	**4693.65**	0
muestra2_55_6	55	6	2	3689.44	-	**3671.28**	**3671.28**	0.21	**3671.28**	**3671.28**	0
muestra3_55_6	55	6	2	4193.40	-	**4167.54**	**4167.54**	0.67	**4167.54**	**4167.54**	0.13
muestra4_55_6	55	6	2	4622.87	-	**4610.31**	**4610.31**	0.60	**4610.31**	**4610.31**	0.19
muestra5_55_6	55	6	2	**4645.89**	-	**4645.89**	**4645.89**	0.04	**4645.89**	**4645.89**	0.20
Average				3254.34	-	**3249.82**	**3249.82**	0.18	**3249.82**	**3249.82**	0.20
#Wins				0		0	0		0	0	
#Ties				19		25	25		25	25	

0 means less than 0.01 seconds. '$*$' symbol indicates an optimal value.
'-' symbol indicates that the corresponding result is not available, i.e., the time limit was reached.

(or '#Ties') indicates the number of instances in which the corresponding method yields the best (or equal) results among all the compared methods in terms of each corresponding performance indicator.

As shown in Table 5, MITS and R-GRASP perform equally well on the 25 small instances and find the same best-known solutions. First, both MITS and R-GRASP consistently find the 10 optimal solutions proven by the IP-based approach in a fraction of a second. Second, the direct application of the IP-based method to larger instances is generally not effective. A comparison on the remaining 15 instances without proven optimal solutions shows that MITS and R-GRASP are more robust, consistently finding the same best

solutions in all 10 runs for each instance, which are better than (6 instances) or equal to (9 instances) the best IP bounds. These results also show that the advantages of MITS and R-GRASP increase with instance size in terms of both solution quality and computation time. Subsequent subsections will show that both heuristic algorithms can be applied to much larger instances, with MITS performing significantly better than R-GRASP.

4.5. Computational results and comparisons on large TSPLib set

To further evaluate the performance of MITS on large and more challenging instances, we conduct an experiment comparing MITS with the state-of-the-art R-GRASP algorithm [20] using the 640 large instances from the TSPLib set. The comparison results of MITS and R-GRASP are presented in Table 6 grouped by graph name, and the detailed results of each instance are listed in Table C.1 of Appendix C. In Table 6, columns 1 and 2 show the characteristics of each instance, including the graph name ($Name$) and the number of vertices (n). The remaining columns are similar to those in Table 5, and include the average deviation of each algorithm from the best solution found among them ($Dev(\%)$), calculated as $(Avg - Prev)/Prev \times 100\%$, where $Prev$ is the best solution found among all compared algorithms. The best of the $Best$ (Avg) values among the compared algorithms for each graph is highlighted in boldface. To verify whether there is a statistically significant difference between the compared results, we use the Wilcoxon signed-rank test at a 0.01 confidence level, where a p-value less than 0.01 indicates a statistically significant difference.

Table 6 shows that MITS significantly outperforms R-GRASP in all performance indicators. In terms of $Best$, MITS achieves improved best values (new upper bounds) in 573 instances, while R-GRASP achieves such a result only in 9 instances. The two algorithms perform equally well in the remaining 58 instances. In terms of Avg, the superiority of MITS over R-GRASP is even more pronounced. MITS achieves the best results in 585 instances against 45 instances for R-GRASP. The average deviation Dev of MITS is also much smaller than that of R-GRASP, indicating a much better performance stability. Additionally, the average running times for MITS and R-GRASP to find their best solutions over the 640 instances are comparable. The very small p-values ($\ll 0.01$) further confirm that the differences between MITS and R-GRASP are statistically significant for both $Best$ and Avg.

To show the evolution of the best solution found during the search process, we provide the running profiles (convergence graphs) of MITS and R-

Table 6: Comparison results of the proposed MITS algorithm with the state-of-the-art R-GRASP algorithm [20] on the 640 large instances grouped by their name (16 instances per group) from the TSPLib set.

Instance		R-GRASP [20]				**MITS**			
Name	*n*	*Best*	*Avg*	*t*(*s*)	*Dev* (%)	*Best*	*Avg*	*t*(*s*)	*Dev* (%)
kroA100	100	303102.71	306656.66	28.94	5.04	**298345.72**	**300951.52**	15.07	1.24
kroB100	100	294319.80	297413.22	28.25	4.76	**289301.83**	**291335.89**	12.96	1.09
kroC100	100	284201.37	287318.22	28.32	4.86	**279901.78**	**281907.35**	13.08	1.03
kroD100	100	285718.78	288435.20	27.41	4.96	**281398.06**	**283808.27**	13.38	1.18
rd100	100	99319.80	100046.75	24.87	4.70	**97712.78**	**98577.03**	12.29	1.20
eil101	101	6988.57	7032.48	28.74	3.79	**6901.97**	**6939.93**	15.06	1.00
lin105	105	219323.12	221708.76	31.70	5.64	**214410.87**	**216355.36**	17.41	1.25
pr107	107	747851.98	758928.69	27.88	9.92	**727679.35**	**738895.91**	13.26	2.75
gr120	120	23953.52	24111.53	29.26	4.40	**23574.19**	**23695.17**	16.82	0.89
pr124	124	1312935.57	1331776.57	26.90	7.09	**1287137.60**	**1296966.33**	15.27	1.55
bier127	127	1885188.27	1897948.84	27.86	3.61	**1863339.39**	**1871838.62**	18.98	0.76
ch130	130	119565.80	120445.22	27.91	5.01	**117413.38**	**118296.93**	17.31	1.06
pr136	136	1921387.30	1936894.03	28.21	6.44	**1875150.66**	**1888021.03**	17.56	1.19
gr137	137	14624.58	14729.92	25.16	5.13	**14370.71**	**14444.29**	18.33	0.87
pr144	144	1975775.27	1993542.31	29.08	7.80	**1923570.90**	**1942959.95**	20.36	1.70
ch150	150	158519.66	160032.98	27.02	5.91	**154844.90**	**156049.28**	19.33	1.18
kroA150	150	676572.61	683110.71	30.02	6.02	**662172.89**	**667317.86**	20.40	1.31
kroB150	150	643381.68	649107.06	28.38	6.11	**626944.73**	**631643.17**	20.97	1.13
pr152	152	2521791.75	2545475.25	28.77	7.19	**2470334.68**	**2504376.91**	21.91	1.78
u159	159	1209981.56	1219220.79	29.15	6.49	**1177631.38**	**1183952.73**	20.20	0.96
rat195	195	82400.05	83070.70	30.08	4.77	**80708.15**	**81588.58**	36.03	1.60
d198	198	599020.61	606901.07	28.94	6.61	**583819.70**	**590463.72**	41.02	1.83
kroA200	200	1154156.01	1160595.25	136.91	5.02	**1132640.46**	**1138656.86**	93.34	0.97
kroB200	200	1167799.39	1175280.11	141.30	4.87	**1146691.41**	**1154135.12**	90.77	1.03
gr202	202	14623.47	14696.72	140.61	3.99	**14385.01**	**14455.58**	106.19	0.71
ts225	225	7122872.03	7159332.29	155.80	5.10	**6996094.35**	**7037167.22**	87.09	1.08
tsp225	225	173486.78	174598.30	156.67	5.07	**170353.92**	**171324.51**	99.79	1.04
pr226	226	5785706.66	5830301.76	141.49	8.16	**5623658.20**	**5676890.28**	106.62	1.62
gr229	229	73710.57	74036.66	150.23	3.94	**72386.71**	**72680.38**	127.24	0.68
gil262	262	136448.97	137101.14	157.86	4.51	**134003.08**	**134853.22**	128.37	0.83
pr264	264	3609209.57	3648218.42	152.45	8.14	**3485405.88**	**3577076.80**	150.96	2.43
a280	280	170758.07	171693.75	152.07	5.24	**167167.59**	**168370.30**	162.69	1.17
pr299	299	3693376.34	3711731.81	157.19	5.12	**3617268.34**	**3637902.91**	177.46	1.10
lin318	318	3385192.54	3405172.69	149.88	4.62	**3321113.97**	**3344450.19**	208.30	0.93
rd400	400	1621619.41	1627516.66	733.48	3.33	**1600874.60**	**1607842.90**	656.58	0.73
fl417	417	2013672.70	2028142.80	776.02	7.42	**1957441.37**	**1969716.92**	745.75	1.17
gr431	431	216597.54	217575.11	781.03	3.69	**212994.35**	**214048.87**	993.78	0.83
pr439	439	15172900.10	15251052.20	783.64	3.79	**14938852.62**	**15035076.67**	810.04	1.03
pcb442	442	6573628.85	6595033.10	765.27	3.95	**6465380.24**	**6501982.89**	860.84	0.94
d493	493	4515462.15	4531117.64	789.97	3.01	**4456124.08**	**4484176.28**	1114.06	0.89
Average		1799678.64	1811177.58	176.12	5.38	**1764237.55**	**1778279.84**	178.42	1.19
#Wins		9	45			**573**	**585**		
#Ties		58	10			58	10		
p-value		2.91e-95	4.44e-85						

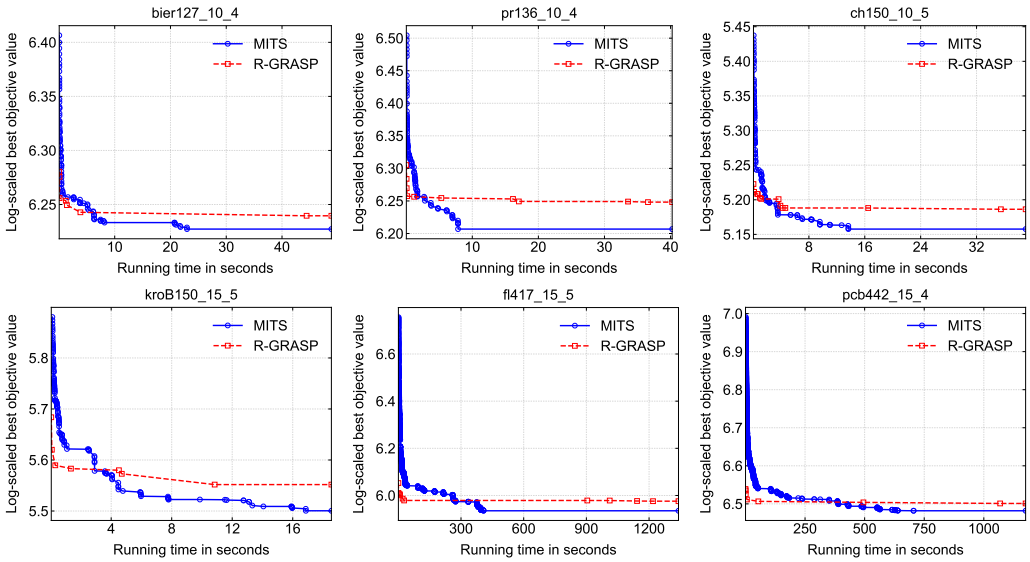

Figure 5: Running profiles (convergence graphs) of the proposed MITS algorithm compared with the state-of-the-art R-GRASP algorithm [20] on 6 representative TSPLib instances.

GRASP on 6 representative TSPLib instances (bier127_10_4, pr136_10_4, ch150_10_5, kroB150_15_5, fl417_15_5, and pcb442_15_4). In Fig. 5, the X-axis indicates the running time in seconds, and the Y-axis indicates the best solution value obtained. We can observe that MITS is more likely to find a better solution than R-GRASP. Regarding pr136_10_4, the best solution value of MITS decreases rapidly at the beginning of the run and then much more slowly, reaching a steady value after 10 seconds. The R-GRASP has a similar running profile, but it quickly converges to a worse value than MITS. The same observation holds for the 5 other instances. These results confirm a clear performance advantage of MITS over R-GRASP.

This experiment indicates that the proposed MITS algorithm performs extremely well on the 640 large and complex TSPLib instances. The 573 new upper bounds established by MITS can serve as valuable reference values when evaluating future algorithms for MCGP. Meanwhile, MITS failed to attain the best solution values for 9 instances, suggesting room for improvement in the future.

5. Applications of the multilevel iterated tabu search to other graph partitioning problem

As mentioned in Section 1, if the weight constraint (2) is kept with $\tau = 2$, the MCGP problem becomes the balanced κ-way partitioning problem with weight constraints (BKPWC) [32]. This section applies the multilevel iterated tabu search (MITS) algorithm to solve this related graph partitioning problem. BKPWC arises naturally in a real-world application involving the realignment of the second category of the Ecuadorian football league [32]. Sports teams must be partitioned into a fixed number of groups according to some regulations, so that the total distance travelled by all teams in each group for a double round-robin tournament is minimized. Until now, two studies have focused on BKPWC: an exact method based on IP formulations of the problem [32], and a heuristic algorithm based on the general VNS algorithm [16].

Similar to MCGP, BKPWC involves partitioning a set of vertices into nonempty, pairwise-disjoint subsets. Each subset must satisfy two constraints, 1) the weights of the vertices in each subset must lie between the lower and upper bounds, and 2) the sizes of the subsets must differ by at most one. The objective is to minimize the total cost of edges with both endpoints in the same subset. To apply the MITS algorithm to BKPWC, we formulate the second balanced constraint as an attribute, assigning a unit weight to each vertex. Then, the MITS algorithm can be applied without any changes.

Two sets of 74 benchmark instances are commonly used in the literature to evaluate BKPWC algorithms: 50 small instances with up to 54 vertices [32] and 24 large instances with up to 2,000 vertices [16]. For this experiment, we recalibrate the parameters of MITS using `irace`. Specifically, the parameters β, α, and λ remain unchanged, while the parameters L, ρ, D, and tt are changed to 1, 2.5%, $0.75 \times n$, and $0.2 \times n$, respectively. We compare MITS with the exact IP approach [32] and the heuristic VNS algorithm [16], where IP was run with a time limit of 3,600 seconds and VNS was run 20 times with a time limit of n seconds per run. Their results are directly extracted from [32, 16]. Following [16], we perform 20 runs of our MITS algorithm per instance, using a time limit of n seconds. Additionally, for the set of large instances, we also use a relaxed stop condition: $2 \times n$ seconds. Studying the outcomes of different stop criteria helps to understand how MITS performs on very large instances when more computation time is available. For clarity, we denote MITS with n seconds as MITS_1 and with $2 \times n$ seconds as MITS_2.

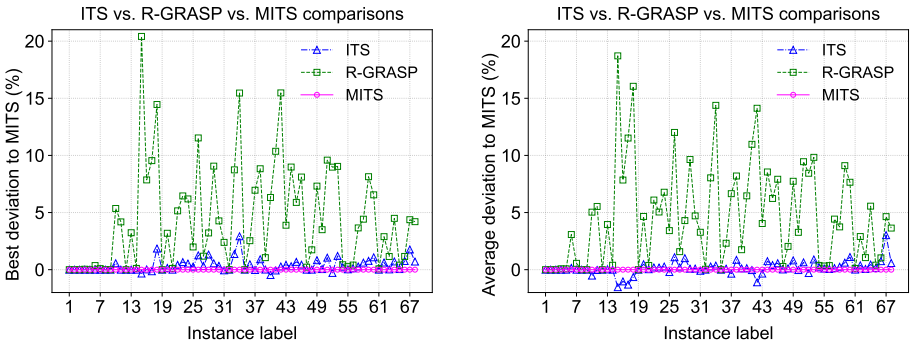

Figure 6: Comparisons of the state-of-the-art R-GRASP algorithm [20], the proposed MITS algorithm, and its variant ITS algorithm on the 68 selected instances.

The detailed computational results are listed in Tables D.1 and D.2 of Appendix D, from which the following observations can be made. For the 50 small instances, MITS finds the 47 optimal solutions proven by IP in a fraction of a second in all 20 runs, and obtains the best-known solutions for the remaining 3 instances (see Table D.1), whereas VNS fails to attain the known optimal value for two instances. For the 24 large instances with $n = 960$ and $2,000$, VNS quickly converges to its best solutions. Specifically, it attains its best solutions within the first 9 seconds in most cases and then remains blocked for the remainder of the given time budget (960 or 2,000 seconds). Only for three instances with $n = 960$, the best results are found toward the end of the search after about 900 seconds. On the other hand, the search behavior of MITS is completely different. It continually improves upon its best solution throughout the search, even when the allotted time is up. This desirable feature enables MITS to find improved best-known solutions for 21 out of the 24 large instances, as indicated by the † symbol in Table D.2.

6. Analysis

We now focus on analyzing the important components of the MITS algorithm: the multilevel optimization approach and the new *Push* and *Relocate* move operators.

6.1. Effectiveness of the multilevel optimization approach

The multilevel optimization approach is the foundation of MITS. To verify its effectiveness and robustness, we conducted a comparative experiment using 68 randomly selected instances (Table B.1). We created a variant of MITS called ITS in which the multilevel optimization approach was removed, leaving only the ITS refinement procedure. To ensure a fair comparison, ITS was run in a multistart way until the cutoff time t_{max} was reached. R-GRASP [20] was also included to allow for direct cross-comparison among R-GRASP, MITS, and ITS. All algorithms are run with the same experimental settings described in Section 4.2. Fig. 6 plots the best/average gaps between R-GRASP, MITS, and ITS on the 68 selected instances. The X-axis is the instance label sorted by their size n, and the Y-axis is the best/average deviation of R-GRASP and ITS from MITS, respectively (%).

Fig. 6 shows that MITS achieves better results than ITS in terms of the best solution values, and performs marginally well in terms of the average solution values. Meanwhile, ITS performs quite well compared to R-GRASP, demonstrating the usefulness of the ITS procedure in MITS. These results are confirmed by the very small p-values obtained from the Wilcoxon signed-rank test. This experiment validates the effectiveness of the multilevel optimization approach and the ITS procedure used by MITS.

6.2. Impact of the new Push and Relocate move operators

MITS uses four complementary move operators *Insert*, *Swap*, *Push*, and *Relocate* to thoroughly explore the search space. To gain deep insights into these operators, we created three MITS variants: NH1 with *Insert* and *Swap*, NH2 with *Insert*, *Swap*, and *Push*, and NH3 with *Insert*, *Swap*, and *Relocate*, and left the other MITS components unchanged. We tested MITS and these three variants under the same experimental conditions described in Section 6.1. Fig. 7 plots the best/average gaps between MITS and NH1, NH2, and NH3 respectively on the 68 selected instances.

From Fig. 7, we can make the following observations. First, MITS performs significantly better than NH1, which demonstrates the effectiveness of the new *Push* and *Relocate* move operators. Second, the results of MITS vs. NH2 and MITS vs. NH3 further show that commenting out *Push* or *Relocate* move operators deteriorate the overall performance of MITS. These observations are confirmed by the very small p-values $\ll 0.01$. This experiment demonstrates the contribution of the new *Push* and *Relocate* move operators to the performance of MITS.

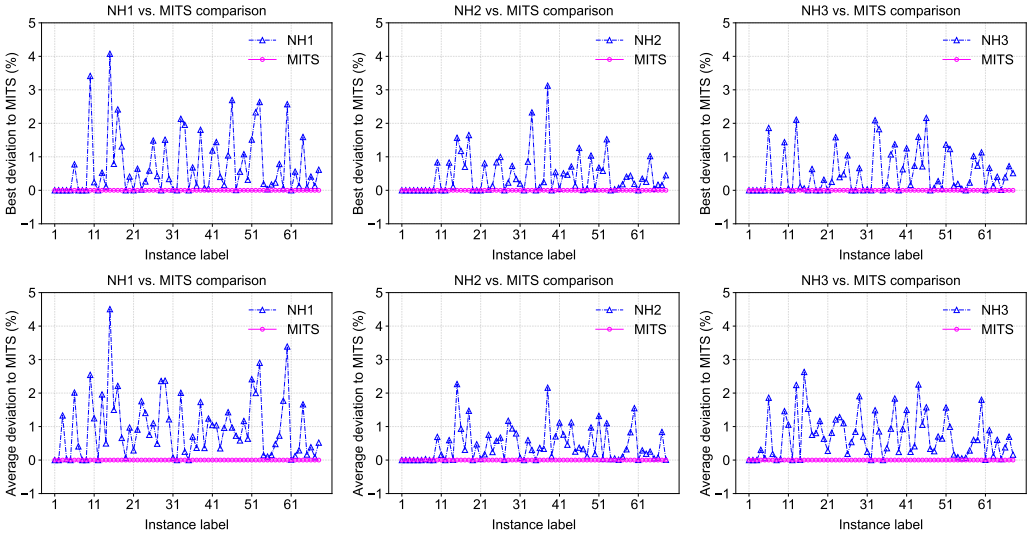

Figure 7: Comparisons of MITS (with *Insert*, *Swap*, *Push*, and *Relocate*) and its variants NH1 (with *Insert* and *Swap*), NH2 (with *Insert*, *Swap*, and *Push*), and NH3 (with *Insert*, *Swap*, and *Relocate*) on the 68 selected instances.

7. Conclusion and perspectives

Graph partitioning encompasses a class of challenging NP-hard problems. Considerable effort has been devoted to developing practical algorithms for various graph partitioning problems. However, there has been limited research on the multi-constraint graph partitioning (MCGP) problem, which generalizes several other well-known graph partitioning problems and remains computationally difficult.

To efficiently tackle MCGP, we presented the first multilevel iterated tabu search (MITS) algorithm that uses a problem-specific coarsening method to reduce progressively the input graph and relies on a dedicated Iterated Tabu Search (ITS) procedure to refine the solution to each intermediate graph. Specifically, ITS benefits from three innovative ideas: it dynamically explores both feasible and infeasible solutions with four complementary move operators (including two new ones *Push* and *Relocate*) to attain high-quality solutions; it integrates a fast incremental evaluation technique to ensure high computation efficiency, and it applies an informed greedy randomized perturbation to escape deep local optima and diversify the search.

Extensive experiments on two sets of 665 benchmark instances demon-

strated that MITS significantly outperforms state-of-the-art algorithms in terms of both solution quality and computation time. Specifically, MITS found new upper bounds for 573 large benchmark instances and matched the best-known upper bounds on 83 instances. We also applied the proposed MITS algorithm to the balanced κ-way partitioning problem with weight constraints and obtained 21 new best results. Additional analyses were performed to verify the usefulness of the key components of the algorithm.

Future work could focus on three main aspects. First, although the MITS algorithm performed well on most benchmark instances, its results on some instances did not match the best-known results. To improve the proposed algorithm further, it would be interesting to study other contraction methods for graph coarsening. Additionally, solution refinement is another critical aspect of multilevel optimization. Therefore, studying other optimization frameworks, such as population-based methods, where a meaningful recombination mechanism can be designed, would be beneficial. This would further boost the search capacity of the algorithm. Finally, the MCGP model focuses on partitioning the vertices of a given graph, ignoring route optimization within each partition. An interesting study would be to consider partitioning and routing simultaneously within a more elaborate model.

Acknowledgment

We are grateful to the reviewers for their valuable comments and suggestions, which helped us to improve the paper. We thank Professor J. Manuel Colmenar for sharing the source code of their algorithm [20], which was very helpful in ensuring a fair comparative study. This work was partially supported by the National Natural Science Foundation of China (Grant no. 72101149) and the Shanghai Pujiang Program (Grant no. 22PJC080).

References

[1] A. Abou-Rjeili, G. Karypis, Multilevel algorithms for partitioning power-law graphs, in: Proceedings 20th IEEE International Parallel & Distributed Processing Symposium, IEEE, 2006, pp. 10–pp.

[2] C. Aykanat, B. B. Cambazoglu, B. Uçar, Multi-level direct k-way hypergraph partitioning with multiple constraints and fixed vertices, Journal of Parallel and Distributed Computing 68 (5) (2008) 609–625.

[3] U. Benlic, J.-K. Hao, A multilevel memetic approach for improving graph k-partitions, IEEE Transactions on Evolutionary Computation 15 (5) (2011) 624–642.

[4] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, C. Schulz, Recent Advances in Graph Partitioning, Springer International Publishing, Cham, 2016, pp. 117–158.

[5] Ü. Çatalyürek, K. Devine, M. Faraj, L. Gottesbüren, T. Heuer, H. Meyerhenke, P. Sanders, S. Schlag, C. Schulz, D. Seemaier, et al., More recent advances in (hyper) graph partitioning, ACM Computing Surveys 55 (12) (2023) 1–38.

[6] H. Chen, B. Perozzi, Y. Hu, S. Skiena, HARP: Hierarchical representation learning for networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2018, pp. 2127–2134.

[7] M. Chernoskutov, Y. Ineichen, C. Bekas, Heuristic algorithm for approximation betweenness centrality using graph coarsening, Procedia Computer Science 66 (2015) 83–92.

[8] C. Chevalier, I. Safro, Comparison of coarsening schemes for multilevel graph partitioning, in: International Conference on Learning and Intelligent Optimization, Springer, 2009, pp. 191–205.

[9] S. Chopra, M. R. Rao, The partition problem, Mathematical Programming 59 (1) (1993) 87–115.

[10] N. Christofides, P. Brooker, The optimal partitioning of graphs, SIAM Journal on Applied Mathematics 30 (1) (1976) 55–69.

[11] J. M. Colmenar, M. Laguna, R. Martín-Santamaría, Tabu search: an application to the minimum dominating set problem, TOP (2025) 1–23.

[12] M. D. Dias, M. R. Mansour, F. Dias, F. Petronetto, C. T. Silva, L. G. Nonato, A hierarchical network simplification via non-negative matrix factorization, in: 30th SIBGRAPI Conference on Graphics, Patterns and Images, IEEE, 2017, pp. 119–126.

[13] N. Fan, P. M. Pardalos, Linear and quadratic programming approaches for the general graph partitioning problem, Journal of Global Optimization 48 (2010) 57–71.

[14] R. Glantz, H. Meyerhenke, C. Schulz, Tree-based coarsening and partitioning of complex networks, Journal of Experimental Algorithmics 21 (2016) 1–20.

[15] F. W. Glover, J.-K. Hao, The case for strategic oscillation, Annals of Operations Research 183 (1) (2011) 163–173.

[16] W. Gong, X. Lai, Q. Yao, S. Zheng, A variable neighborhood search algorithm for the balanced k-way partitioning problem with weight constraints, in: China Automation Congress, IEEE, 2021, pp. 1106–1110.

[17] S. Gutierrez, A. Miniguano, D. Recalde, L. Torres, R. Torres, P. Zuleta, Integrated vehicle and pollster routing, Tech. rep., Escuela Politécnica Nacional, Department of Mathematics, 07 (2019).

[18] F. Hausberger, M. Fonseca Faraj, C. Schulz, A distributed multilevel memetic algorithm for signed graph clustering, in: Proceedings of the Companion Conference on Genetic and Evolutionary Computation, 2023, pp. 207–210.

[19] M. He, Q. Wu, U. Benlic, Y. Lu, Y. Chen, An effective multi-level memetic search with neighborhood reduction for the clustered team orienteering problem, European Journal of Operational Research 318 (3) (2024) 778–801.

[20] A. Herrán, J. M. Colmenar, M. G. Resende, Two-phase GRASP for the multi-constraint graph partitioning problem, Computers & Operations Research (2024) 106946.

[21] Hexaly, Hexaly 13.5 documentation (2025).
URL https://www.hexaly.com/docs/last/index.html

[22] G. Karypis, V. Kumar, Analysis of multilevel graph partitioning, in: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing, 1995, pp. 29–es.

[23] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM Journal on Scientific Computing 20 (1) (1998) 359–392.

[24] G. Karypis, V. Kumar, Multilevel algorithms for multi-constraint graph partitioning, in: Proceedings of the 1998 ACM/IEEE Conference on Supercomputing, IEEE, 1998, pp. 28–28.

[25] G. Karypis, V. Kumar, Multilevel k-way partitioning scheme for irregular graphs, Journal of Parallel and Distributed Computing 48 (1) (1998) 96–129.

[26] B. W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, The Bell System Technical Journal 49 (2) (1970) 291–307.

[27] M. Labbé, F. A. Özsoy, Size-constrained graph partitioning polytopes, Discrete Mathematics 310 (24) (2010) 3473–3493.

[28] D. LaSalle, G. Karypis, Multi-threaded graph partitioning, in: 27th International Symposium on Parallel and Distributed Processing, IEEE, 2013, pp. 225–236.

[29] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, T. Stützle, The irace package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (2016) 43–58.

[30] Z. Lu, J.-K. Hao, U. Benlic, D. Lesaint, Iterated multilevel simulated annealing for large-scale graph conductance minimization, Information Sciences 572 (2021) 182–199.

[31] J. E. Mitchell, Branch-and-cut for the k-way equipartition problem, Tech. rep., Mathematical Sciences, Rensselaer Polytechnic Institute (2001).

[32] D. Recalde, D. Severín, R. Torres, P. Vaca, An exact approach for the balanced k-way partitioning problem with weight constraints and its application to sports team realignment, Journal of Combinatorial Optimization 36 (3) (2018) 916–936.

[33] D. Recalde, R. Torres, P. Vaca, An exact approach for the multi-constraint graph partitioning problem, EURO Journal on Computational Optimization 8 (3-4) (2020) 289–308.

[34] P. Sanders, C. Schulz, Engineering multilevel graph partitioning algorithms, in: European Symposium on Algorithms, Springer, 2011, pp. 469–480.

[35] K. Schloegel, G. Karypis, V. Kumar, Parallel static and dynamic multi-constraint graph partitioning, Concurrency and Computation: Practice and Experience 14 (3) (2002) 219–240.

[36] G. M. Slota, K. Madduri, S. Rajamanickam, PuLP: Scalable multi-objective multi-constraint partitioning for small-world networks, in: IEEE International Conference on Big Data, IEEE, 2014, pp. 481–490.

[37] W. Sun, J.-K. Hao, Y. Zang, X. Lai, A solution-driven multilevel approach for graph coloring, Applied Soft Computing 104 (2021) 107174.

[38] M. Tanaka, O. Tatebe, Workflow scheduling to minimize data movement using multi-constraint graph partitioning, in: 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE, 2012, pp. 65–72.

[39] S.-H. Teng, Coarsening, sampling, and smoothing: Elements of the multilevel method, in: Algorithms for Parallel Processing, Springer, 1999, pp. 247–276.

[40] TSPLIB (2025).
URL http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

[41] B. Uçar, C. Aykanat, Partitioning sparse matrices for parallel preconditioned iterative methods, SIAM Journal on Scientific Computing 29 (4) (2007) 1683–1709.

[42] A. Valejo, M. C. F. de Oliveira, P. Geraldo Filho, A. de Andrade Lopes, Multilevel approach for combinatorial optimization in bipartite network, Knowledge-Based Systems 151 (2018) 45–61.

[43] A. Valejo, V. Ferreira, R. Fabbri, M. C. F. de Oliveira, A. de Andrade Lopes, A critical survey of the multilevel method in complex networks, ACM Computing Surveys 53 (2) (2020) 1–35.

[44] C. Walshaw, Multilevel refinement for combinatorial optimisation problems, Annals of Operations Research 131 (2004) 325–372.

[45] C. Walshaw, Multilevel refinement for combinatorial optimisation: Boosting metaheuristic performance, in: Hybrid Metaheuristics: an Emerging Approach to Optimization, Springer, 2008, pp. 261–289.

[46] Q. Zhan, W. Zhu, X. He, A GRASP based algorithm for multi-constraint circuit partitioning, in: sixth International Conference on Natural Computation, 2010, pp. 3110–3114.

Appendix A. The fast incremental evaluation technique

As indicated in Section 3.6.1, the MITS algorithm relies on four move operators to explore the search space: the *Insert* and *Swap* moves used in the previous study [20] and two new move operators *Push* and *Relocate* introduced in this work. To ensure a high computation efficiency in exploring the resulting neighborhoods, we adapt the incremental evaluation technique [20] for the *Insert* and *Swap* moves and design a fast incremental evaluation technique for the two new search operators.

1. *Insert* operator. We first define the sum of the distances between a vertex $v \in S_i$ and the vertices that belong to the subset $S_j \in \mathcal{P}$ $(i, j \in K)$ as follows,

$$\delta(\mathcal{P}, v, S_j) = \sum_{u \in S_j} d(v, u) \tag{A.1}$$

The move value of the objective function F after performing an *Insert* move on $\mathcal{P}$ is given by,

$$\Delta_F(Insert(v, S_i, S_j)) = \delta(\mathcal{P}, v, S_j) - \delta(\mathcal{P}, v, S_i) \tag{A.2}$$

The degree of constraint violations in terms of lower and upper limits for subsets S_i and S_j, assuming that v moves from S_i to S_j in $\mathcal{P}$, is defined by,

$$\begin{aligned}\sigma_{Insert}(\mathcal{P}, v, S_i, S_j) = &\sum_{t=1}^{\tau}(\max\{W_{it}^L - (w(\mathcal{P}, S_i, t) - w_v^t), 0\} \\ &+ \max\{w(\mathcal{P}, S_i, t) - w_v^t - W_{it}^U, 0\}) + \\ &\sum_{t=1}^{\tau}(\max\{W_{jt}^L - (w(\mathcal{P}, S_j, t) + w_v^t), 0\} \\ &+ \max\{w(\mathcal{P}, S_j, t) + w_v^t - W_{jt}^U, 0\})\end{aligned} \tag{A.3}$$

where the first term calculates the degree of constraint violations reduced by extracting v from S_i, and the second term calculates the result by adding v to S_j.

The move value of the penalty function C after performing an *Insert* move on $\mathcal{P}$ is defined by,

$$\Delta_C(Insert(v, S_i, S_j)) = \sigma_{Insert}(\mathcal{P}, v, S_i, S_j) - \sigma(\mathcal{P}, S_i) - \sigma(\mathcal{P}, S_j) \quad \text{(A.4)}$$

which is computed by subtracting the current σ values for both S_i and S_j from $\sigma_{Insert}(\mathcal{P}, v, S_i, S_j)$.

Thus, the move value of the extended evaluation function F^* (denoted as $\Delta_{F^*}(Insert)$) is a linear combination of $\Delta_F(Insert)$ and $\Delta_C(Insert)$ as follows,

$$\triangle_{F^*}(Insert) = \triangle_F(Insert) + \phi \times \triangle_C(Insert) \quad \text{(A.5)}$$

Note that the move value computations only consider subsets affected by the *Insert* move, i.e., edges where one of the endpoints is v. Once the move explored by $N_{Insert}(\mathcal{P})$ is performed, the values of δ and w are updated in $\mathcal{O}(1)$ and $\mathcal{O}(\tau)$ as follows,

$$\delta(\mathcal{P}, c, S_x) = \begin{cases} \delta(\mathcal{P}, c, S_i) - d(v, c) & x = i \ \wedge \ \forall c \in V \\ \delta(\mathcal{P}, c, S_j) + d(v, c) & x = j \ \wedge \ \forall c \in V \end{cases} \quad \text{(A.6)}$$

$$w(\mathcal{P}, S_x, t) = \begin{cases} w(\mathcal{P}, S_i, t) - w_v^t & x = i \ \wedge \ \forall t \in T \\ w(\mathcal{P}, S_j, t) + w_v^t & x = j \ \wedge \ \forall t \in T \end{cases} \quad \text{(A.7)}$$

2. *Swap* operator. A *Swap* move can be decomposed into two successive *Insert* moves, where one of the swap vertices moves from S_i to S_j, and the other way around. The move value of the objective function F is given by,

$$\begin{aligned} \Delta_F(Swap(v, u, S_i, S_j)) = {} & \delta(\mathcal{P}, v, S_j) - \delta(\mathcal{P}, v, S_i) \\ & + \delta(\mathcal{P}, u, S_i) - \delta(\mathcal{P}, u, S_j) - 2d(v, u) \end{aligned} \quad \text{(A.8)}$$

after exchanging $v \in S_i$ to S_j and $u \in S_j$ to S_i.

The degree of constraint violations in terms of lower and upper limits for subsets S_i and S_j, assuming that v moves from S_i to S_j and u moves from

S_j to S_i, is defined by,

$$\begin{aligned}\sigma_{Swap}(\mathcal{P}, v, u, S_i, S_j) = &\sum_{t=1}^{\tau}(\max\{W_{it}^L - (w(\mathcal{P}, S_i, t) - w_v^t + w_u^t), 0\} \\ &+ \max\{w(\mathcal{P}, S_i, t) - w_v^t + w_u^t - W_{it}^U, 0\})+ \\ &\sum_{t=1}^{\tau}(\max\{W_{jt}^L - (w(\mathcal{P}, S_j, t) + w_v^t - w_u^t), 0\} \\ &+ \max\{w(\mathcal{P}, S_j, t) + w_v^t - w_u^t - W_{jt}^U, 0\})\end{aligned} \tag{A.9}$$

The move value of the penalty function C after performing a *Swap* move on $\mathcal{P}$ is defined by,

$$\Delta_C(Swap(v, u, S_i, S_j)) = \sigma_{Swap}(\mathcal{P}, v, u, S_i, S_j) - \sigma(\mathcal{P}, S_i) - \sigma(\mathcal{P}, S_j) \tag{A.10}$$

Thus, the move value of the extended evaluation function F^* (denoted as $\Delta_{F^*}(Swap)$) is a linear combination of $\Delta_F(Swap)$ and $\Delta_C(Swap)$ as follows,

$$\triangle_{F^*}(Swap) = \triangle_F(Swap) + \phi \times \triangle_C(Swap) \tag{A.11}$$

Once the move explored by $N_{Swap}(\mathcal{P})$ is performed, the values of δ and w are updated in $\mathcal{O}(1)$ and $\mathcal{O}(\tau)$ as follows,

$$\delta(\mathcal{P}, c, S_x) = \begin{cases}\delta(\mathcal{P}, c, S_i) - d(v, c) + d(u, c) & x = i \ \wedge \ \forall c \in V \\ \delta(\mathcal{P}, c, S_j) + d(v, c) - d(u, c) & x = j \ \wedge \ \forall c \in V\end{cases} \tag{A.12}$$

$$w(\mathcal{P}, S_x, t) = \begin{cases}w(\mathcal{P}, S_i, t) - w_v^t + w_u^t & x = i \ \wedge \ \forall t \in T \\ w(\mathcal{P}, S_j, t) + w_v^t - w_u^t & x = j \ \wedge \ \forall t \in T\end{cases} \tag{A.13}$$

3. *Push* operator. A *Push* move can be decomposed into two successive *Insert* moves related to three subsets. The move value of the objective function F is defined as,

$$\begin{aligned}\Delta_F(Push(v, u, S_i, S_j, S_l)) = &\delta(\mathcal{P}, v, S_j) - \delta(\mathcal{P}, v, S_i) \\ &+ \delta(\mathcal{P}, u, S_l) - \delta(\mathcal{P}, u, S_j) - d(v, u)\end{aligned} \tag{A.14}$$

after pushing $v \in S_i$ to S_j and $u \in S_j$ to S_l.

The degree of constraint violations in terms of lower and upper limits for subsets S_i, S_j, and S_l, assuming that $v \in S_i$ pushes $u \in S_j$ to S_l, is defined by,

$$\begin{aligned}\sigma_{Push}(\mathcal{P}, v, u, S_i, S_j, S_l) = &\sum_{t=1}^{\tau}(\max\{W_{it}^L - (w(\mathcal{P}, S_i, t) - w_v^t), 0\} \\ &+ \max\{w(\mathcal{P}, S_i, t) - w_v^t - W_{it}^U, 0\})+ \\ &\sum_{t=1}^{\tau}(\max\{W_{jt}^L - (w(\mathcal{P}, S_j, t) - w_u^t + w_v^t), 0\} \\ &+ \max\{w(\mathcal{P}, S_j, t) - w_u^t + w_v^t - W_{jt}^U, 0\})+ \\ &\sum_{t=1}^{\tau}(\max\{W_{lt}^L - (w(\mathcal{P}, S_l, t) + w_u^t), 0\} \\ &+ \max\{w(\mathcal{P}, S_l, t) + w_u^t - W_{lt}^U, 0\})\end{aligned} \tag{A.15}$$

The move value of the penalty function C after performing a *Push* move on $\mathcal{P}$ is defined by,

$$\begin{aligned}\Delta_C(Push(v, u, S_i, S_j, S_l)) = &\sigma_{Push}(\mathcal{P}, v, u, S_i, S_j, S_l) \\ &- \sigma(\mathcal{P}, S_i) - \sigma(\mathcal{P}, S_j) - \sigma(\mathcal{P}, S_l)\end{aligned} \tag{A.16}$$

Thus, the move value of the extended evaluation function F^* (denoted as $\Delta_{F^*}(Push)$) is a linear combination of $\Delta_F(Push)$ and $\Delta_C(Push)$ as follows,

$$\triangle_{F^*}(Push) = \triangle_F(Push) + \phi \times \triangle_C(Push) \tag{A.17}$$

Once the move explored by $N_{Push}(\mathcal{P})$ is performed, the values of δ and w are updated in $\mathcal{O}(1)$ and $\mathcal{O}(\tau)$ as follows,

$$\delta(\mathcal{P}, c, S_x) = \begin{cases}\delta(\mathcal{P}, c, S_i) - d(v, c) & x = i \ \wedge \ \forall c \in V \\ \delta(\mathcal{P}, c, S_j) + d(v, c) - d(u, c) & x = j \ \wedge \ \forall c \in V \\ \delta(\mathcal{P}, c, S_l) + d(u, c) & x = l \ \wedge \ \forall c \in V\end{cases} \tag{A.18}$$

$$w(\mathcal{P}, S_x, t) = \begin{cases} w(\mathcal{P}, S_i, t) - w_v^t & x = i \ \wedge \ \forall t \in T \\ w(\mathcal{P}, S_j, t) + w_v^t - w_u^t & x = j \ \wedge \ \forall t \in T \\ w(\mathcal{P}, S_l, t) + w_u^t & x = l \ \wedge \ \forall t \in T \end{cases} \tag{A.19}$$

4. *Relocate* operator. A *Relocate* move can be decomposed into three successive *Insert* moves or two successive *Push* moves. The move value of the objective function F is defined as,

$$\begin{aligned} \Delta_F(Relocate(v, u, g, S_i, S_j, S_l)) = \delta(\mathcal{P}, v, S_j) - \delta(\mathcal{P}, v, S_i) + \delta(\mathcal{P}, u, S_l) \\ - \delta(\mathcal{P}, u, S_j) + \delta(\mathcal{P}, g, S_i) - \delta(\mathcal{P}, g, S_l) \\ - d(v, u) - d(u, g) - d(g, v) \end{aligned} \tag{A.20}$$

after moving $v \in S_i$ to S_j, $u \in S_j$ to S_l, and $g \in S_l$ to S_i.

The degree of constraint violations in terms of lower and upper limits for subsets S_i, S_j, and S_l, assuming that v moves from S_i to S_j, u moves from S_j to S_l, and g moves from S_l to S_i, is defined by,

$$\begin{aligned} \sigma_{Relocate}(\mathcal{P}, v, u, g, S_i, S_j, S_l) = & \sum_{t=1}^{\tau} (\max\{W_{it}^L - (w(\mathcal{P}, S_i, t) - w_v^t + w_g^t), 0\} \\ & + \max\{w(\mathcal{P}, S_i, t) - w_v^t + w_g^t - W_{it}^U, 0\}) + \\ & \sum_{t=1}^{\tau} (\max\{W_{jt}^L - (w(\mathcal{P}, S_j, t) - w_u^t + w_v^t), 0\} \\ & + \max\{w(\mathcal{P}, S_j, t) - w_u^t + w_v^t - W_{jt}^U, 0\}) + \\ & \sum_{t=1}^{\tau} (\max\{W_{lt}^L - (w(\mathcal{P}, S_l, t) - w_g^t + w_u^t), 0\} \\ & + \max\{w(\mathcal{P}, S_l, t) - w_g^t + w_u^t - W_{lt}^U, 0\}) \end{aligned} \tag{A.21}$$

The move value of the penalty function C after performing a *Relocate* move on $\mathcal{P}$ is defined by,

$$\begin{aligned} \Delta_C(Relocate(v, u, g, S_i, S_j, S_l)) = \sigma_{Relocate}(\mathcal{P}, v, u, g, S_i, S_j, S_l) \\ - \sigma(\mathcal{P}, S_i) - \sigma(\mathcal{P}, S_j) - \sigma(\mathcal{P}, S_l) \end{aligned} \tag{A.22}$$

Thus, the move value of the extended evaluation function F^* (denoted as $\Delta_{F^*}(Relocate)$) is a linear combination of $\Delta_F(Relocate)$ and $\Delta_C(Relocate)$ as follows,

$$\triangle_{F^*}(Relocate) = \triangle_F(Relocate) + \phi \times \triangle_C(Relocate) \tag{A.23}$$

Once the move explored by $N_{Relocate}(\mathcal{P})$ is performed, the values of δ and w are updated in $\mathcal{O}(1)$ and $\mathcal{O}(\tau)$ as follows,

$$\delta(\mathcal{P}, c, S_x) = \begin{cases} \delta(\mathcal{P}, c, S_i) + d(g, c) - d(v, c) & x = i \ \wedge \ \forall c \in V \\ \delta(\mathcal{P}, c, S_j) + d(v, c) - d(u, c) & x = j \ \wedge \ \forall c \in V \\ \delta(\mathcal{P}, c, S_l) + d(u, c) - d(g, c) & x = l \ \wedge \ \forall c \in V \end{cases} \tag{A.24}$$

$$w(\mathcal{P}, S_x, t) = \begin{cases} w(\mathcal{P}, S_i, t) + w_g^t - w_v^t & x = i \ \wedge \ \forall t \in T \\ w(\mathcal{P}, S_j, t) + w_v^t - w_u^t & x = j \ \wedge \ \forall t \in T \\ w(\mathcal{P}, S_l, t) + w_u^t - w_g^t & x = l \ \wedge \ \forall t \in T \end{cases} \tag{A.25}$$

Appendix B. The 68 benchmark instances used for parameter tuning and analyses for the MCGP problem

We used 68 instances for parameter tuning and algorithm analysis in Sections 4.3 and 6. These instances are randomly selected from the 665 benchmark instances and listed in Table B.1 and sorted by size n.

Table B.1: The 68 selected benchmark instances.

muestra3_40_5	pr107_15_5	gr137_15_5	rat195_20_3	gil262_10_5
muestra4_50_6	pr107_20_3	pr144_15_2	d198_20_4	gil262_15_3
muestra1_55_6	pr124_20_4	pr144_15_3	kroA200_10_5	gil262_20_4
kroA100_10_2	pr124_20_5	ch150_05_2	kroB200_15_4	pr264_20_2
kroB100_05_3	bier127_05_3	ch150_15_4	gr202_05_4	a280_05_2
kroB100_20_2	bier127_20_3	ch150_20_5	gr202_10_3	a280_15_2
kroD100_15_2	ch130_05_4	kroB150_05_3	gr202_15_5	pr299_10_2
rd100_05_3	ch130_10_5	kroB150_05_5	tsp225_10_4	fl417_20_2
rd100_05_4	ch130_15_3	pr152_10_5	tsp225_15_5	gr431_05_4
rd100_10_5	ch130_15_4	u159_10_5	tsp225_20_4	gr431_10_2
rd100_20_3	pr136_20_2	u159_15_2	pr226_15_4	gr431_10_5
eil101_05_2	pr136_20_4	u159_15_3	gr229_05_2	d493_20_3
eil101_20_3	gr137_10_3	u159_20_4	gr229_05_3	
lin105_10_2	gr137_15_3	u159_20_5	gr229_05_4	

Appendix C. Detailed computational results on the large TSPLib set for the MCGP problem

Table C.1 presents detailed computational results of the proposed MITS algorithm and the state-of-the-art R-GRASP algorithm [20] on the 640 large instances from the TSPLib set. As indicated in Section 4.2, both algorithms were run 10 times to solve each instance on the same computing platform, using their default parameter settings. The time limit per run for both algorithms was set to 10 seconds for instances with $n \in [0, 100)$, 60 seconds for instances with $n \in [100, 200)$, 300 seconds for instances with $n \in [200, 400)$, and 1,500 seconds for instances with $n \in [400, 500)$.

In Table C.1, columns 1-4 show the characteristics of the instance: the instance name ($Name$), the number of vertices (n), the number of partitions (κ), and the number of attributes (τ). The remaining columns show the detailed results of R-GRASP and MITS, including the best ($Best$) and the average (Avg) solution values over 10 independent runs, the average computation time in seconds to reach the best solution value ($t(s)$), and the average deviation of each algorithm from the best solution found among them ($Dev(\%)$), calculated as $(Avg - Prev)/Prev \times 100\%$, where $Prev$ is the best solution found among all compared algorithms. The best of the $Best$ (Avg) values among the compared results for each instance is highlighted in boldface. A dagger (†) indicates a strictly best solution value among the $Best$ results. Thus, bold and dagger values correspond to new best upper bounds compared to the literature. As shown in Table C.1, our proposed MITS algorithm can find 573 new upper bounds and match 58 best-known upper bounds on these 640 large TSPLib instances.

Table C.1: Detailed computational results of the proposed MITS algorithm and the state-of-the-art R-GRASP algorithm [20] on the 640 large instances from the TSPLib set.

Instance				R-GRASP [20]				**MITS**			
Name	n	κ	τ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
kroA100_05_2	100	5	2	**679873.90**	**679873.90**	2.28	0	**679873.90**	682276.74	1.33	0.35
kroA100_05_3	100	5	3	**729716.18**	**733227.72**	22.09	0.48	**729716.18**	736160.66	18.54	0.88
kroA100_05_4	100	5	4	757453.72	763242.17	27.39	1.13	**754704.56**†	**757968.21**	9.38	0.43
kroA100_05_5	100	5	5	767042.98	771994.56	30.52	1.53	**760353.35**†	**764527.19**	12.63	0.55
kroA100_10_2	100	10	2	216665.98	216747.80	34.23	0.08	**216577.40**†	**216577.40**	2.58	0
kroA100_10_3	100	10	3	241420.02	245357.68	20.11	2.57	**239215.97**†	**240216.21**	13.23	0.42
kroA100_10_4	100	10	4	281147.04	286884.77	35.59	5.03	**273139.09**†	**278295.49**	27.72	1.89
kroA100_10_5	100	10	5	292570.17	298221.80	26.19	5.87	**281694.05**†	**285443.25**	19.11	1.33
kroA100_15_2	100	15	2	**103942.79**	104028.42	24.48	0.08	**103942.79**	**104013.38**	6.79	0.07
kroA100_15_3	100	15	3	128378.15	132024.60	34.08	6.42	**124064.78**†	**125166.02**	14.86	0.89
kroA100_15_4	100	15	4	147570.71	157416.09	39.77	10.44	**142539.55**†	**146413.20**	17.87	2.72

Table C.1: Continued

Instance				R-GRASP [20]				**MITS**			
Name	n	κ	τ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
kroA100_15_5	100	15	5	161664.81	165825.27	39.94	15.69	**143337.73**†	**147809.26**	14.98	3.12
kroA100_20_2	100	20	2	64911.74	65200.01	34.93	2.19	**63804.69**†	**64620.36**	13.33	1.28
kroA100_20_3	100	20	3	79447.10	81596.30	30.39	6.25	**76797.35**†	**78197.10**	12.09	1.82
kroA100_20_4	100	20	4	96243.17	99839.38	34.10	10.15	**90636.97**†	**92489.92**	24.83	2.04
kroA100_20_5	100	20	5	101594.86	105026.06	26.95	12.77	**93133.22**†	**95049.98**	31.90	2.06
kroB100_05_2	100	5	2	**664914.16**	**664914.16**	5.04	0	**664914.16**	**664914.16**	0.56	0
kroB100_05_3	100	5	3	**666400.39**	666895.93	24.77	0.07	**666400.39**	**666400.39**	2.08	0
kroB100_05_4	100	5	4	703266.87	709458.65	37.39	1.28	**700526.05**†	**703283.50**	21.61	0.39
kroB100_05_5	100	5	5	772902.42	782998.87	23.86	2.11	**766805.56**†	**771056.92**	23.58	0.55
kroB100_10_2	100	10	2	**213907.37**	214747.96	18.24	0.39	**213907.37**	**214216.63**	14.72	0.14
kroB100_10_3	100	10	3	234931.67	237180.69	35.10	2.46	**231486.94**†	**232694.01**	16.87	0.52
kroB100_10_4	100	10	4	258001.88	261574.51	25.44	4.38	**250596.72**†	**254529.74**	4.07	1.57
kroB100_10_5	100	10	5	301319.14	307275.89	33.83	8.10	**284256.48**†	**291041.65**	24.60	2.39
kroB100_15_2	100	15	2	110132.85	111390.80	32.32	1.57	**109664.15**†	**109678.08**	1.89	0.01
kroB100_15_3	100	15	3	128262.16	129456.56	22.70	4.29	**124134.21**†	**125168.11**	17.75	0.83
kroB100_15_4	100	15	4	144962.32	146436.77	42.94	7.27	**136515.52**†	**137290.35**	19.23	0.57
kroB100_15_5	100	15	5	170477.04	176286.95	35.38	12.30	**156979.34**†	**162810.16**	16.39	3.71
kroB100_20_2	100	20	2	65230.46	66996.43	31.45	3.08	**64997.35**†	**64997.35**	3.17	0
kroB100_20_3	100	20	3	76157.30	79543.86	27.04	7.20	**74198.98**†	**76033.01**	11.72	2.47
kroB100_20_4	100	20	4	88274.72	90611.87	34.69	9.91	**82442.41**†	**84508.62**	12.41	2.51
kroB100_20_5	100	20	5	109975.98	112841.67	21.88	11.72	**101003.61**†	**102751.56**	16.76	1.73
kroC100_05_2	100	5	2	**669029.38**	**669029.38**	3.59	0	**669029.38**	**669029.38**	1.84	0
kroC100_05_3	100	5	3	**663221.40**	**664856.43**	20.67	0.25	**663221.40**	665127.93	5.60	0.29
kroC100_05_4	100	5	4	742886.94	751464.62	34.08	1.59	**739737.38**†	**743463.68**	14.01	0.50
kroC100_05_5	100	5	5	680268.33	685753.49	33.67	1.31	**676879.93**†	**681542.84**	4.96	0.69
kroC100_10_2	100	10	2	217217.89	**218196.49**	25.96	0.58	**216948.29**†	218746.34	16.45	0.83
kroC100_10_3	100	10	3	224763.06	228290.90	16.71	2.00	**223807.97**†	**225040.11**	6.79	0.55
kroC100_10_4	100	10	4	265334.06	270725.48	37.54	5.59	**256396.21**†	**260084.56**	24.81	1.44
kroC100_10_5	100	10	5	245636.14	247938.08	35.51	7.25	**231167.38**†	**235855.94**	17.64	2.03
kroC100_15_2	100	15	2	**109936.26**	111984.02	27.26	1.86	**109936.26**	**110983.85**	11.89	0.95
kroC100_15_3	100	15	3	119927.25	123331.01	28.59	4.75	**117733.02**†	**118260.79**	13.50	0.45
kroC100_15_4	100	15	4	147085.20	149339.30	25.79	10.12	**135614.24**†	**137819.24**	24.58	1.63
kroC100_15_5	100	15	5	135924.13	141004.52	32.46	8.63	**129804.00**†	**131987.88**	10.17	1.68
kroC100_20_2	100	20	2	69108.31	70286.57	27.14	1.75	**69077.04**†	**69270.19**	21.25	0.28
kroC100_20_3	100	20	3	78497.61	79689.08	32.95	5.73	**75367.62**†	**76131.84**	9.39	1.01
kroC100_20_4	100	20	4	92085.08	94031.04	36.44	11.87	**84054.34**†	**86002.55**	9.91	2.32
kroC100_20_5	100	20	5	86300.90	91171.05	34.80	14.46	**79654.02**†	**81170.54**	16.41	1.90
kroD100_05_2	100	5	2	**663126.41**	**663126.41**	0.49	0	**663126.41**	666093.00	1.15	0.45
kroD100_05_3	100	5	3	**711251.82**	**713003.73**	29.88	0.25	**711251.82**	715811.74	8.79	0.64
kroD100_05_4	100	5	4	715176.27	718728.25	30.15	0.80	**713025.86**†	**717313.30**	17.39	0.60
kroD100_05_5	100	5	5	**712806.42**	**718316.07**	24.52	0.77	**712806.42**	718604.95	6.16	0.81
kroD100_10_2	100	10	2	**207321.11**	207695.83	27.43	0.18	**207321.11**	**207321.11**	2.61	0
kroD100_10_3	100	10	3	242148.88	244507.45	27.44	3.96	**235183.69**†	**237532.33**	10.31	1.00
kroD100_10_4	100	10	4	249011.50	252145.32	22.57	5.34	**239359.14**†	**242866.31**	22.08	1.47
kroD100_10_5	100	10	5	243323.25	252379.56	34.50	6.95	**235985.42**†	**237979.72**	19.84	0.85
kroD100_15_2	100	15	2	104896.31	105380.09	34.85	0.55	**104799.12**†	**104799.12**	7.69	0
kroD100_15_3	100	15	3	123864.45	127176.67	33.61	7.48	**118330.53**†	**118616.73**	13.90	0.24
kroD100_15_4	100	15	4	132538.97	136199.46	28.81	6.27	**128164.10**†	**129589.15**	23.64	1.11
kroD100_15_5	100	15	5	142186.09	146013.86	29.95	12.85	**129386.19**†	**135150.29**	19.46	4.45
kroD100_20_2	100	20	2	67183.04	67490.82	31.81	2.07	**66120.97**†	**66773.14**	11.35	0.99
kroD100_20_3	100	20	3	81342.32	82747.06	32.84	7.11	**77255.92**†	**78259.10**	13.05	1.30
kroD100_20_4	100	20	4	85771.90	88060.81	27.67	12.27	**78434.40**†	**80454.25**	16.85	2.58
kroD100_20_5	100	20	5	89551.79	91991.86	22.04	12.44	**81817.81**†	**83768.05**	19.80	2.38
rd100_05_2	100	5	2	**229619.71**	**229619.71**	0.47	0	**229619.71**	229986.04	1.19	0.16
rd100_05_3	100	5	3	**231572.32**	**231572.32**	6.02	0	**231572.32**	**231572.32**	0.80	0
rd100_05_4	100	5	4	**232378.70**	**232378.70**	6.86	0	**232378.70**	**232378.70**	1.98	0

Table C.1: Continued

Instance				R-GRASP [20]				**MITS**			
Name	n	κ	τ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
rd100_05_5	100	5	5	259200.52	**260975.02**	22.95	1.37	**257459.93**†	261987.46	17.80	1.76
rd100_10_2	100	10	2	**74677.17**	74944.59	32.06	0.36	**74677.17**	**74677.17**	1.75	0
rd100_10_3	100	10	3	81866.06	82345.22	25.17	0.95	**81573.99**†	**82199.97**	9.55	0.77
rd100_10_4	100	10	4	82892.47	83800.31	27.64	3.62	**80869.49**†	**81410.34**	19.18	0.67
rd100_10_5	100	10	5	96963.84	99712.05	30.36	8.34	**92034.48**†	**94932.90**	12.72	3.15
rd100_15_2	100	15	2	40536.36	41078.47	32.17	2.68	**40005.85**†	**40883.41**	19.76	2.19
rd100_15_3	100	15	3	43135.35	44052.50	26.90	5.34	**41817.95**†	**42057.41**	13.85	0.57
rd100_15_4	100	15	4	45172.82	45864.96	38.96	10.11	**41653.97**†	**42915.22**	18.95	3.03
rd100_15_5	100	15	5	54640.50	55738.09	32.94	10.72	**50341.43**†	**51375.78**	16.24	2.05
rd100_20_2	100	20	2	23788.72	24083.05	26.99	1.26	**23783.79**†	**23788.23**	2.83	0.02
rd100_20_3	100	20	3	28070.74	28501.59	35.80	5.78	**26943.91**†	**27007.30**	11.93	0.24
rd100_20_4	100	20	4	29042.94	29852.39	28.20	7.09	**27874.94**†	**28146.42**	25.65	0.97
rd100_20_5	100	20	5	35558.57	36229.07	24.48	17.64	**30796.90**†	**31913.75**	22.53	3.63
eil101_05_2	101	5	2	**15863.34**	**15863.34**	1.72	0	**15863.34**	**15863.34**	1.49	0
eil101_05_3	101	5	3	**16529.70**	16566.38	22.30	0.22	**16529.70**	**16539.37**	10.89	0.06
eil101_05_4	101	5	4	**16639.99**	**16680.60**	20.61	0.24	**16639.99**	16704.35	17.86	0.39
eil101_05_5	101	5	5	**16500.68**	**16514.22**	29.61	0.08	**16500.68**	16521.73	22.16	0.13
eil101_10_2	101	10	2	**5720.45**†	5736.73	38.64	0.28	5720.61	**5726.40**	7.44	0.10
eil101_10_3	101	10	3	6024.28	6106.66	43.66	2.77	**5942.03**†	**6010.47**	26.12	1.15
eil101_10_4	101	10	4	6126.01	6161.43	31.50	3.63	**5945.50**†	**5977.33**	5.32	0.54
eil101_10_5	101	10	5	6307.37	6427.69	30.25	4.94	**6125.15**†	**6214.67**	9.80	1.46
eil101_15_2	101	15	2	3029.92	3047.44	28.24	1.80	**2993.60**†	**3024.21**	23.21	1.02
eil101_15_3	101	15	3	3366.99	3397.42	39.46	4.03	**3265.79**†	**3298.01**	14.67	0.99
eil101_15_4	101	15	4	3505.83	3549.76	31.82	8.16	**3281.88**†	**3381.03**	14.60	3.02
eil101_15_5	101	15	5	3626.08	3697.14	24.75	10.27	**3352.68**†	**3402.28**	18.49	1.48
eil101_20_2	101	20	2	**1888.02**†	1916.45	34.92	1.51	1901.01	**1903.77**	10.56	0.83
eil101_20_3	101	20	3	2110.51	2147.36	23.29	5.04	**2044.40**†	**2065.76**	17.90	1.04
eil101_20_4	101	20	4	2221.83	2263.55	29.69	6.97	**2116.07**†	**2137.43**	25.64	1.01
eil101_20_5	101	20	5	2356.16	2443.59	29.36	10.61	**2209.13**†	**2268.70**	14.81	2.70
lin105_05_2	105	5	2	**512253.03**	**512334.23**	18.85	0.02	**512253.03**	512650.57	9.48	0.08
lin105_05_3	105	5	3	**489862.62**	**489862.62**	15.49	0	**489862.62**	491955.00	1.08	0.43
lin105_05_4	105	5	4	516002.44	518134.84	28.14	0.55	**515278.68**†	**515664.05**	17.22	0.07
lin105_05_5	105	5	5	593320.30	599572.10	30.78	4.68	**572788.86**†	**583851.35**	20.91	1.93
lin105_10_2	105	10	2	170906.97	171547.79	34.75	0.49	**170718.75**†	**170888.15**	10.35	0.10
lin105_10_3	105	10	3	172111.70	174927.54	33.90	2.45	**170747.71**†	**171018.83**	22.90	0.16
lin105_10_4	105	10	4	181708.97	185582.85	30.35	3.57	**179179.71**†	**179776.33**	20.98	0.33
lin105_10_5	105	10	5	223346.42	229181.57	39.25	10.37	**207645.79**†	**211469.44**	24.60	1.84
lin105_15_2	105	15	2	86903.92	87297.60	31.02	2.93	**84813.89**†	**84889.26**	11.76	0.09
lin105_15_3	105	15	3	94119.93	95403.49	34.27	4.43	**91358.72**†	**92574.44**	31.16	1.33
lin105_15_4	105	15	4	96429.71	100565.55	26.58	11.17	**90463.92**†	**93629.72**	15.33	3.50
lin105_15_5	105	15	5	122876.00	127381.54	42.77	14.05	**111684.54**†	**115411.59**	25.92	3.34
lin105_20_2	105	20	2	52162.66	52851.91	39.66	4.37	**50640.30**†	**50903.88**	22.27	0.52
lin105_20_3	105	20	3	59521.62	60826.04	32.97	6.63	**57043.07**†	**58488.74**	7.67	2.53
lin105_20_4	105	20	4	60569.86	61705.52	33.38	9.76	**56219.61**†	**56975.28**	10.68	1.34
lin105_20_5	105	20	5	77073.78	80165.01	35.02	14.73	**69874.72**†	**71539.18**	26.31	2.38
pr107_05_2	107	5	2	**1850536.57**	1850992.23	11.00	0.02	**1850536.57**	**1850536.57**	13.74	0
pr107_05_3	107	5	3	1924802.38	1938612.96	32.09	0.81	**1922972.60**†	**1938606.37**	15.43	0.81
pr107_05_4	107	5	4	1976492.47	**1979152.05**	35.48	0.23	**1974688.08**†	1982028.41	12.23	0.37
pr107_05_5	107	5	5	2114352.38	2143350.49	32.33	2.89	**2083110.83**†	**2136436.26**	7.98	2.56
pr107_10_2	107	10	2	536927.02	543034.78	28.64	4.76	**518337.47**†	**520347.12**	13.82	0.39
pr107_10_3	107	10	3	478575.04	481895.83	29.64	2.12	**471885.01**†	**473694.11**	5.96	0.38
pr107_10_4	107	10	4	467862.04	475159.16	29.28	6.17	**447536.14**†	**452157.70**	6.74	1.03
pr107_10_5	107	10	5	735147.03	772495.61	36.05	12.19	**688533.46**†	**706806.95**	21.25	2.65
pr107_15_2	107	15	2	236210.37	247924.91	26.29	6.65	**232456.83**†	**239671.60**	8.35	3.10
pr107_15_3	107	15	3	293851.42	297659.10	16.45	6.80	**278699.15**†	**285148.37**	14.53	2.31
pr107_15_4	107	15	4	298411.54	306678.91	31.68	15.93	**264548.94**†	**279722.55**	17.27	5.74

Table C.1: Continued

Instance				R-GRASP [20]				**MITS**			
Name	n	κ	τ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
pr107_15_5	107	15	5	388001.72	403536.00	26.52	25.23	**322225.57**†	**339926.76**	21.41	5.49
pr107_20_2	107	20	2	150986.98	154936.17	38.86	13.23	**136827.39**†	**143638.03**	5.72	4.98
pr107_20_3	107	20	3	155706.29	158329.21	24.86	9.67	**144364.65**†	**146811.95**	9.24	1.70
pr107_20_4	107	20	4	151685.05	157923.70	24.50	15.08	**137227.91**†	**138951.93**	14.76	1.26
pr107_20_5	107	20	5	206083.34	231177.97	22.41	36.86	**168919.03**†	**187849.88**	23.72	11.21
gr120_05_2	120	5	2	**55121.44**	55148.49	15.73	0.05	**55121.44**	**55121.44**	2.82	0
gr120_05_3	120	5	3	**55730.57**†	55814.64	31.13	0.15	55796.52	**55800.56**	12.22	0.13
gr120_05_4	120	5	4	57018.12	57211.18	29.20	0.54	**56901.21**†	**57064.89**	13.80	0.29
gr120_05_5	120	5	5	58563.91	58820.38	28.62	0.90	**58298.39**†	**58433.31**	17.65	0.23
gr120_10_2	120	10	2	18981.08	19006.93	27.36	0.49	**18914.94**†	**18940.22**	12.35	0.13
gr120_10_3	120	10	3	19941.00	20023.70	35.08	2.55	**19526.67**†	**19711.51**	34.63	0.95
gr120_10_4	120	10	4	20536.28	20726.73	19.57	4.34	**19864.73**†	**20055.51**	19.08	0.96
gr120_10_5	120	10	5	22604.56	22942.68	34.56	5.99	**21646.21**†	**21979.26**	14.34	1.54
gr120_15_2	120	15	2	10048.33	10137.71	33.53	2.08	**9931.47**†	**10014.73**	7.95	0.84
gr120_15_3	120	15	3	10697.12	10851.69	30.07	4.79	**10355.30**†	**10486.74**	24.80	1.27
gr120_15_4	120	15	4	11507.95	11675.02	35.53	7.60	**10850.40**†	**10945.54**	27.29	0.88
gr120_15_5	120	15	5	13140.38	13369.75	22.92	9.98	**12156.49**†	**12377.31**	18.50	1.82
gr120_20_2	120	20	2	6312.07	6375.34	19.13	3.60	**6153.52**†	**6176.71**	14.72	0.38
gr120_20_3	120	20	3	7072.81	7153.91	40.56	5.64	**6772.16**†	**6866.99**	11.83	1.40
gr120_20_4	120	20	4	7522.07	7714.04	22.32	9.56	**7040.95**†	**7156.51**	20.54	1.64
gr120_20_5	120	20	5	8458.64	8812.23	42.81	12.16	**7856.61**†	**7991.49**	16.58	1.72
pr124_05_2	124	5	2	**3108030.55**	**3108030.55**	0.87	0	**3108030.55**	**3108030.55**	2.87	0
pr124_05_3	124	5	3	3023671.58	3024888.38	21.25	0.12	**3021378.31**†	**3021378.31**	0.97	0
pr124_05_4	124	5	4	3522418.08	3561275.64	32.53	1.44	**3510808.50**†	**3524594.51**	22.72	0.39
pr124_05_5	124	5	5	**3219857.30**	3262042.77	40.52	1.31	**3219857.30**	**3229477.60**	11.04	0.30
pr124_10_2	124	10	2	**951298.74**†	956195.03	15.86	0.51	951653.55	**951675.67**	12.43	0.04
pr124_10_3	124	10	3	1009154.56	1031188.34	36.00	3.92	**992298.48**†	**1000019.73**	10.28	0.78
pr124_10_4	124	10	4	1207682.95	1244110.33	28.79	6.16	**1171963.34**†	**1192374.29**	18.13	1.74
pr124_10_5	124	10	5	1153617.53	1194196.50	28.31	10.68	**1078927.30**†	**1101319.08**	13.36	2.08
pr124_15_2	124	15	2	517208.23	527934.19	21.77	7.20	**492490.91**†	**507640.99**	29.85	3.08
pr124_15_3	124	15	3	554209.30	564356.33	28.78	5.18	**536577.25**†	**539834.98**	17.17	0.61
pr124_15_4	124	15	4	642936.21	666754.11	32.55	13.16	**589191.61**†	**606893.20**	18.94	3.00
pr124_15_5	124	15	5	643208.54	659609.50	27.23	12.02	**588858.14**†	**598919.86**	15.39	1.71
pr124_20_2	124	20	2	325306.54	332421.29	27.34	6.67	**311632.76**†	**316954.40**	20.96	1.71
pr124_20_3	124	20	3	335687.98	344034.32	25.52	10.07	**312552.79**†	**321434.06**	14.25	2.84
pr124_20_4	124	20	4	393562.58	414151.76	32.35	15.28	**359271.19**†	**371376.22**	18.96	3.37
pr124_20_5	124	20	5	399118.49	417236.09	30.74	19.65	**348709.64**†	**359537.79**	17.03	3.11
bier127_05_2	127	5	2	**4351115.89**	**4361312.30**	9.72	0.23	**4351115.89**	4362629.83	8.20	0.26
bier127_05_3	127	5	3	**4470967.22**	**4470967.22**	19.47	0	**4470967.22**	**4470967.22**	6.03	0
bier127_05_4	127	5	4	**4443811.17**†	4480536.35	31.74	0.83	4453405.51	**4476120.60**	31.04	0.73
bier127_05_5	127	5	5	4374759.09	4382072.72	34.82	0.31	**4368704.52**†	**4379368.51**	12.17	0.24
bier127_10_2	127	10	2	1546125.16	1551928.49	27.06	1.74	**1525336.63**†	**1529147.30**	15.68	0.25
bier127_10_3	127	10	3	1643799.75	1651439.23	27.65	2.39	**1612817.61**†	**1616569.58**	26.77	0.23
bier127_10_4	127	10	4	1720062.10	1731392.14	28.87	2.67	**1686428.82**†	**1698368.80**	22.29	0.71
bier127_10_5	127	10	5	1674849.01	1698810.56	25.98	4.40	**1627151.36**†	**1636026.96**	20.25	0.55
bier127_15_2	127	15	2	809491.25	815952.89	33.36	3.11	**791325.57**†	**797881.49**	18.62	0.83
bier127_15_3	127	15	3	881817.40	893690.37	34.92	2.74	**869868.84**†	**873948.80**	16.72	0.47
bier127_15_4	127	15	4	947902.00	959627.30	22.27	6.16	**903974.50**†	**913695.76**	28.10	1.08
bier127_15_5	127	15	5	961187.87	978092.30	35.46	7.83	**907095.66**†	**916502.49**	20.26	1.04
bier127_20_2	127	20	2	501210.15	509900.12	23.66	3.48	**492759.53**†	**497885.43**	17.37	1.04
bier127_20_3	127	20	3	570418.71	582472.45	32.27	5.37	**552808.07**†	**556549.48**	15.18	0.68
bier127_20_4	127	20	4	637760.95	650786.53	30.55	7.69	**604310.99**†	**621128.17**	20.02	2.78
bier127_20_5	127	20	5	627734.67	648200.51	27.95	8.88	**595359.47**†	**602627.51**	24.99	1.22
ch130_05_2	130	5	2	**277375.22**	**277375.22**	7.38	0	**277375.22**	280501.67	3.01	1.13
ch130_05_3	130	5	3	**286443.03**	**287171.80**	24.90	0.25	**286443.03**	287599.55	16.94	0.40

Table C.1: Continued

Instance				R-GRASP [20]				**MITS**			
Name	n	κ	τ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
ch130_05_4	130	5	4	278582.64	279371.78	27.60	0.39	**278278.63**†	**278339.44**	10.16	0.02
ch130_05_5	130	5	5	297133.61	298283.58	25.72	1.03	**295248.89**†	**296013.34**	14.66	0.26
ch130_10_2	130	10	2	**92929.36**†	93946.55	17.82	1.09	92962.15	**93571.61**	9.10	0.69
ch130_10_3	130	10	3	102774.64	103515.73	41.82	3.10	**100404.53**†	**101574.47**	16.68	1.17
ch130_10_4	130	10	4	100621.57	102088.14	22.68	5.42	**96836.35**†	**98073.37**	21.68	1.28
ch130_10_5	130	10	5	109001.45	111358.04	37.63	7.42	**103663.13**†	**104962.08**	22.64	1.25
ch130_15_2	130	15	2	48446.54	49282.71	35.28	2.99	**47850.13**†	**48239.30**	16.67	0.81
ch130_15_3	130	15	3	55911.69	56144.71	30.83	6.90	**52521.41**†	**53446.81**	23.93	1.76
ch130_15_4	130	15	4	56653.70	57665.42	23.45	8.09	**53350.09**†	**54005.13**	16.85	1.23
ch130_15_5	130	15	5	63839.49	65000.72	31.04	8.96	**59654.34**†	**60465.28**	22.84	1.36
ch130_20_2	130	20	2	30815.53	31095.20	36.73	3.87	**29935.85**†	**30065.03**	7.70	0.43
ch130_20_3	130	20	3	35043.97	36005.96	33.01	7.33	**33547.83**†	**34227.28**	27.13	2.03
ch130_20_4	130	20	4	35963.76	36646.39	25.41	10.14	**33273.17**†	**33894.26**	20.27	1.87
ch130_20_5	130	20	5	41516.60	42171.65	25.31	13.15	**37269.33**†	**37772.22**	26.70	1.35
pr136_05_2	136	5	2	**4454790.40**	**4454790.40**	1.00	0	**4454790.40**	**4454790.40**	6.03	0
pr136_05_3	136	5	3	**4494365.93**	**4494365.93**	16.47	0	**4494365.93**	4495356.53	15.09	0.02
pr136_05_4	136	5	4	4757986.44	4781391.31	29.30	0.79	**4744090.45**†	**4751670.55**	16.67	0.16
pr136_05_5	136	5	5	4761305.61	4778075.24	35.33	1.58	**4703852.40**†	**4722741.73**	10.72	0.40
pr136_10_2	136	10	2	1471275.56	1482992.25	28.78	1.21	**1465265.15**†	**1465849.98**	18.61	0.04
pr136_10_3	136	10	3	1540982.10	1555185.26	29.23	3.75	**1498982.54**†	**1521246.65**	18.66	1.49
pr136_10_4	136	10	4	1730064.33	1753574.72	31.59	8.93	**1609866.07**†	**1656303.28**	13.97	2.88
pr136_10_5	136	10	5	1760008.90	1790842.07	31.67	6.79	**1676916.34**†	**1696675.89**	22.70	1.18
pr136_15_2	136	15	2	770762.20	780680.15	35.70	2.92	**758509.89**†	**760338.38**	20.39	0.24
pr136_15_3	136	15	3	778366.45	799319.39	33.54	6.02	**753921.65**†	**760046.54**	11.81	0.81
pr136_15_4	136	15	4	936359.05	957572.34	21.63	13.91	**840609.64**†	**863766.10**	19.65	2.75
pr136_15_5	136	15	5	1016694.31	1031933.51	34.97	13.81	**906737.11**†	**930136.12**	20.40	2.58
pr136_20_2	136	20	2	486680.36	498129.17	25.93	4.39	**477189.23**†	**481662.70**	24.59	0.94
pr136_20_3	136	20	3	525752.79	532912.56	27.09	7.15	**497369.17**†	**505222.94**	17.62	1.58
pr136_20_4	136	20	4	615418.72	629849.84	39.65	14.14	**551798.16**†	**562310.44**	20.46	1.91
pr136_20_5	136	20	5	641383.58	668690.32	29.53	17.70	**568146.46**†	**580218.19**	23.59	2.12
gr137_05_2	137	5	2	**34543.89**†	**34552.47**	24.81	0.02	34545.63	34554.22	6.08	0.03
gr137_05_3	137	5	3	**34812.57**	**34818.61**	27.96	0.02	**34812.57**	**34818.61**	6.69	0.02
gr137_05_4	137	5	4	36058.25	36195.07	22.17	0.76	**35922.28**†	**36122.39**	24.27	0.56
gr137_05_5	137	5	5	35325.92	35532.86	29.13	1.20	**35110.11**†	**35212.29**	15.99	0.29
gr137_10_2	137	10	2	**11191.83**	11250.63	31.92	0.53	**11191.83**	**11211.12**	14.89	0.17
gr137_10_3	137	10	3	12292.82	12374.69	19.12	1.85	**12149.49**†	**12182.46**	16.56	0.27
gr137_10_4	137	10	4	12738.22	12854.39	23.29	5.06	**12235.01**†	**12337.60**	20.32	0.84
gr137_10_5	137	10	5	12957.91	13177.75	25.36	5.64	**12474.33**†	**12655.57**	28.84	1.45
gr137_15_2	137	15	2	5904.68	5980.71	16.09	1.59	**5886.99**†	**5910.36**	12.14	0.40
gr137_15_3	137	15	3	6625.59	6746.97	25.98	5.11	**6419.13**†	**6469.03**	17.33	0.78
gr137_15_4	137	15	4	7009.41	7145.18	18.44	10.06	**6492.00**†	**6571.41**	13.66	1.22
gr137_15_5	137	15	5	7220.79	7384.54	31.88	11.54	**6620.82**†	**6735.20**	25.21	1.73
gr137_20_2	137	20	2	3685.59	3733.87	34.77	4.43	**3575.36**†	**3600.91**	15.77	0.71
gr137_20_3	137	20	3	4239.81	4330.16	17.51	5.86	**4090.30**†	**4140.80**	27.64	1.23
gr137_20_4	137	20	4	4580.59	4666.78	26.02	13.01	**4129.35**†	**4192.03**	23.87	1.52
gr137_20_5	137	20	5	4805.43	4934.10	28.14	15.39	**4276.09**†	**4394.57**	23.98	2.77
pr144_05_2	144	5	2	**4800033.65**	**4800033.65**	27.01	0	**4800033.65**	4828792.16	15.10	0.60
pr144_05_3	144	5	3	**4707627.97**	**4713981.31**	21.67	0.13	**4707627.97**	4730494.02	4.98	0.49
pr144_05_4	144	5	4	**5006088.48**	**5015834.21**	23.67	0.19	**5006088.48**	5036394.32	19.73	0.61
pr144_05_5	144	5	5	5185503.85	5260877.62	32.75	2.30	**5142770.04**†	**5160227.62**	20.17	0.34
pr144_10_2	144	10	2	1413119.82	**1421561.95**	20.29	0.76	**1410795.14**†	1428500.52	10.26	1.25
pr144_10_3	144	10	3	**1371129.95**	1383754.54	22.00	0.92	**1371129.95**	**1374569.80**	14.84	0.25
pr144_10_4	144	10	4	1691498.84	1715408.52	31.62	7.19	**1600391.87**†	**1623605.82**	15.99	1.45
pr144_10_5	144	10	5	1900432.99	1936508.69	22.70	12.24	**1725368.90**†	**1785912.81**	22.72	3.51
pr144_15_2	144	15	2	750426.31	765557.44	34.90	6.37	**719687.73**†	**731127.61**	15.84	1.59
pr144_15_3	144	15	3	682246.90	689499.94	26.77	3.48	**666326.08**†	**667642.31**	18.49	0.20

Table C.1: Continued

Instance				R-GRASP [20]				**MITS**			
Name	n	κ	τ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
pr144_15_4	144	15	4	903304.82	925452.31	35.39	10.91	**834433.69**†	**845853.05**	20.56	1.37
pr144_15_5	144	15	5	1077619.53	1093693.14	24.93	17.98	**927000.20**†	**940236.03**	34.56	1.43
pr144_20_2	144	20	2	433541.98	439255.86	39.24	5.17	**417680.10**†	**421604.73**	30.84	0.94
pr144_20_3	144	20	3	434430.11	441684.69	36.81	10.74	**398856.12**†	**408110.41**	26.22	2.32
pr144_20_4	144	20	4	565332.06	584506.72	36.21	21.46	**481240.73**†	**517691.86**	21.99	7.57
pr144_20_5	144	20	5	690066.98	709066.37	29.39	24.90	**567703.73**†	**586596.11**	33.49	3.33
ch150_05_2	150	5	2	**361120.88**	361250.74	24.88	0.04	**361120.88**	**361120.88**	3.88	0
ch150_05_3	150	5	3	364583.05	364736.35	25.35	0.31	**363594.60**†	**364125.60**	5.01	0.15
ch150_05_4	150	5	4	383493.35	385045.06	26.75	1.32	**380022.11**†	**380352.16**	18.10	0.09
ch150_05_5	150	5	5	391014.28	395530.02	32.26	3.19	**383292.99**†	**389442.55**	16.30	1.60
ch150_10_2	150	10	2	124367.41	125758.53	20.45	1.34	**124090.07**†	**124431.69**	12.24	0.28
ch150_10_3	150	10	3	126887.86	127730.31	21.04	2.33	**124819.07**†	**125687.03**	12.27	0.70
ch150_10_4	150	10	4	139273.12	142210.44	34.85	4.73	**135793.43**†	**137637.69**	20.72	1.36
ch150_10_5	150	10	5	148940.36	152881.85	27.94	8.05	**141492.25**†	**142663.94**	22.14	0.83
ch150_15_2	150	15	2	65899.83	66838.72	29.10	2.77	**65037.61**†	**65514.91**	23.91	0.73
ch150_15_3	150	15	3	68706.63	69848.15	33.36	5.54	**66179.26**†	**66712.97**	24.43	0.81
ch150_15_4	150	15	4	79947.32	80843.40	26.17	9.96	**73524.01**†	**74827.95**	24.74	1.77
ch150_15_5	150	15	5	85917.48	87758.22	29.14	11.52	**78694.10**†	**80614.47**	24.23	2.44
ch150_20_2	150	20	2	42012.72	42489.92	28.64	4.49	**40664.37**†	**41085.10**	17.23	1.03
ch150_20_3	150	20	3	43333.05	44754.14	27.38	9.42	**40902.00**†	**41378.65**	25.34	1.17
ch150_20_4	150	20	4	53139.36	53737.15	25.08	11.17	**48336.59**†	**49510.62**	26.66	2.43
ch150_20_5	150	20	5	57677.90	59114.63	19.86	18.34	**49955.02**†	**51682.22**	32.11	3.46
kroA150_05_2	150	5	2	**1553615.17**	**1553615.17**	8.68	0	**1553615.17**	1557468.16	6.64	0.25
kroA150_05_3	150	5	3	**1598427.14**	1605009.62	29.21	0.41	**1598427.14**	**1600892.60**	16.49	0.15
kroA150_05_4	150	5	4	1693875.26	1700213.84	31.69	1.32	**1678031.00**†	**1682870.50**	26.19	0.29
kroA150_05_5	150	5	5	1638790.16	1650659.77	40.24	1.55	**1625432.75**†	**1632718.94**	14.42	0.45
kroA150_10_2	150	10	2	**520112.40**	526345.30	28.01	1.20	**520112.40**	**523941.92**	13.41	0.74
kroA150_10_3	150	10	3	550079.18	557585.09	32.38	2.57	**543601.90**†	**545682.30**	24.67	0.38
kroA150_10_4	150	10	4	629806.74	640914.28	25.86	5.38	**608186.79**†	**616341.70**	28.24	1.34
kroA150_10_5	150	10	5	598288.61	608325.71	23.73	8.95	**558339.64**†	**573279.95**	20.08	2.68
kroA150_15_2	150	15	2	266361.75	269984.89	35.97	2.42	**263614.67**†	**266458.09**	26.05	1.08
kroA150_15_3	150	15	3	289918.12	292870.36	30.81	5.35	**277992.24**†	**280749.43**	19.24	0.99
kroA150_15_4	150	15	4	343903.04	353247.95	39.93	11.90	**315671.16**†	**324054.65**	21.49	2.66
kroA150_15_5	150	15	5	333452.56	346077.77	33.03	11.27	**311034.49**†	**316837.52**	24.12	1.87
kroA150_20_2	150	20	2	170758.69	173056.94	37.74	4.86	**165042.28**†	**167819.77**	18.62	1.68
kroA150_20_3	150	20	3	181970.44	187221.56	30.27	9.04	**171696.48**†	**174354.03**	17.29	1.55
kroA150_20_4	150	20	4	226003.90	230092.15	21.08	12.12	**205220.92**†	**208361.89**	19.85	1.53
kroA150_20_5	150	20	5	229798.60	234550.88	31.63	18.01	**198747.19**†	**205254.24**	29.60	3.27
kroB150_05_2	150	5	2	**1456138.75**	**1456138.75**	12.71	0	**1456138.75**	1456566.24	10.16	0.03
kroB150_05_3	150	5	3	**1446759.17**	**1446759.17**	7.02	0	**1446759.17**	**1446759.17**	5.46	0
kroB150_05_4	150	5	4	1571982.60	1582830.75	31.86	1.42	**1560709.50**†	**1567534.82**	16.07	0.44
kroB150_05_5	150	5	5	1651170.87	1662021.47	27.57	3.21	**1610291.05**†	**1624458.15**	22.28	0.88
kroB150_10_2	150	10	2	485169.52	**486959.34**	35.82	0.59	**484085.68**†	488065.68	13.62	0.82
kroB150_10_3	150	10	3	507500.42	511072.11	33.03	2.00	**501028.94**†	**502161.54**	16.00	0.23
kroB150_10_4	150	10	4	579561.74	585523.94	37.23	6.71	**548721.51**†	**558163.35**	22.15	1.72
kroB150_10_5	150	10	5	620986.96	636385.77	29.43	10.33	**576811.88**†	**589492.59**	25.58	2.20
kroB150_15_2	150	15	2	251749.74	254439.18	21.65	1.69	**250207.54**†	**250625.09**	19.88	0.17
kroB150_15_3	150	15	3	279256.80	281264.80	31.69	3.21	**272513.99**†	**273663.52**	24.97	0.42
kroB150_15_4	150	15	4	322179.87	326944.17	31.17	11.06	**294384.23**†	**300825.55**	22.64	2.19
kroB150_15_5	150	15	5	341943.60	359513.66	28.15	14.05	**315215.97**†	**319145.12**	29.72	1.25
kroB150_20_2	150	20	2	160537.02	162453.18	31.32	4.32	**155723.55**†	**156614.69**	24.04	0.57
kroB150_20_3	150	20	3	175586.51	179594.30	23.82	5.85	**169672.20**†	**171732.97**	26.09	1.21
kroB150_20_4	150	20	4	211348.28	215164.59	25.70	12.34	**191529.00**†	**196692.76**	22.34	2.70
kroB150_20_5	150	20	5	232235.09	238647.77	45.98	20.94	**197322.75**†	**203789.52**	34.47	3.28
pr152_05_2	152	5	2	6359853.19	**6359853.19**	14.96	0.29	**6341719.45**†	6437002.11	11.91	1.50

Table C.1: Continued

Instance				R-GRASP [20]				**MITS**			
Name	n	κ	τ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
pr152_05_3	152	5	3	**6470708.58**†	**6474737.87**	19.42	0.06	6484158.87	6506772.45	29.45	0.56
pr152_05_4	152	5	4	6634220.02	**6665081.57**	26.88	0.94	**6603079.30**†	6741056.53	14.95	2.09
pr152_05_5	152	5	5	6655086.83	6687294.80	17.83	1.44	**6592574.39**†	**6652923.00**	11.63	0.92
pr152_10_2	152	10	2	1788579.83	1790063.26	36.65	0.36	**1783721.98**†	**1789004.89**	24.52	0.30
pr152_10_3	152	10	3	1915593.79	1934806.38	26.54	1.85	**1899642.75**†	**1905927.73**	20.16	0.33
pr152_10_4	152	10	4	1996078.39	2064356.01	40.86	6.25	**1942837.02**†	**1977047.64**	22.86	1.76
pr152_10_5	152	10	5	2159279.18	2196274.35	30.79	8.78	**2018915.01**†	**2059284.91**	27.24	2.00
pr152_15_2	152	15	2	846370.21	850797.18	31.58	3.34	**823295.35**†	**833438.86**	16.99	1.23
pr152_15_3	152	15	3	978641.42	990178.85	25.66	8.59	**911871.15**†	**926898.80**	17.86	1.65
pr152_15_4	152	15	4	1069448.57	1102425.11	29.77	9.77	**1004348.62**†	**1019634.48**	30.15	1.52
pr152_15_5	152	15	5	1085740.04	1132268.79	32.67	20.06	**943101.70**†	**992215.42**	27.62	5.21
pr152_20_2	152	20	2	436392.66	448203.46	23.66	8.15	**414420.71**†	**419115.45**	12.76	1.13
pr152_20_3	152	20	3	587055.56	600670.82	34.03	8.44	**553925.76**†	**565514.16**	25.14	2.09
pr152_20_4	152	20	4	616636.22	663647.64	31.46	16.87	**567859.33**†	**588980.38**	27.16	3.72
pr152_20_5	152	20	5	748983.43	766944.71	37.52	19.86	**639883.45**†	**655213.78**	30.10	2.40
u159_05_2	159	5	2	**2729418.15**	**2729418.15**	10.80	0	**2729418.15**	2729557.12	4.52	0.01
u159_05_3	159	5	3	2875934.00	2886756.99	28.89	0.90	**2860916.83**†	**2863325.74**	15.42	0.08
u159_05_4	159	5	4	2853662.45	2867884.35	29.49	0.83	**2844233.41**†	**2850317.80**	21.80	0.21
u159_05_5	159	5	5	3091448.74	3113491.94	29.05	2.62	**3033913.27**†	**3047188.39**	20.58	0.44
u159_10_2	159	10	2	922853.28	924806.35	32.41	0.41	**921061.60**†	**922230.78**	26.59	0.13
u159_10_3	159	10	3	1000809.84	1007502.69	25.14	3.15	**976779.23**†	**984677.65**	22.60	0.81
u159_10_4	159	10	4	989890.65	1006847.42	35.76	4.92	**959629.28**†	**964712.59**	15.59	0.53
u159_10_5	159	10	5	1176561.20	1189438.14	40.24	10.02	**1081062.86**†	**1099304.30**	24.78	1.69
u159_15_2	159	15	2	489602.46	493433.21	31.16	1.84	**484497.72**†	**484953.87**	11.90	0.09
u159_15_3	159	15	3	548404.43	553261.35	21.77	7.26	**515824.64**†	**519679.73**	17.01	0.75
u159_15_4	159	15	4	541797.45	552558.83	23.90	11.12	**497261.74**†	**500638.33**	25.69	0.68
u159_15_5	159	15	5	666277.35	684242.12	36.80	13.69	**601849.29**†	**612422.96**	33.84	1.76
u159_20_2	159	20	2	317219.12	320431.59	28.30	4.76	**305877.76**†	**307257.66**	14.94	0.45
u159_20_3	159	20	3	353908.08	358317.75	29.74	11.66	**320909.11**†	**330078.43**	16.68	2.86
u159_20_4	159	20	4	359076.11	367141.27	29.94	12.84	**325361.68**†	**330844.43**	22.02	1.69
u159_20_5	159	20	5	442841.58	452000.41	32.97	17.86	**383505.44**†	**396053.87**	29.30	3.27
rat195_05_2	195	5	2	**187044.99**	**187103.42**	26.19	0.03	**187044.99**	187613.11	24.43	0.30
rat195_05_3	195	5	3	186772.81	187254.60	36.00	0.29	**186710.76**†	**187073.11**	19.27	0.19
rat195_05_4	195	5	4	189794.69	190524.45	22.88	0.40	**189771.38**†	**190317.69**	24.50	0.29
rat195_05_5	195	5	5	208680.60	212310.44	27.52	3.70	**204727.06**†	**207558.82**	37.75	1.38
rat195_10_2	195	10	2	64379.58	64616.63	38.96	1.28	**63802.09**†	**64203.13**	32.77	0.63
rat195_10_3	195	10	3	63508.84	64037.41	32.82	2.22	**62646.05**†	**63339.10**	31.89	1.11
rat195_10_4	195	10	4	66973.65	68121.77	16.95	3.80	**65630.47**†	**66548.05**	27.51	1.40
rat195_10_5	195	10	5	85834.58	86478.25	31.87	7.92	**80129.05**†	**82035.98**	52.54	2.38
rat195_15_2	195	15	2	35433.76	35876.60	27.03	2.50	**35001.36**†	**35328.94**	30.89	0.94
rat195_15_3	195	15	3	35568.65	35878.18	35.02	3.11	**34795.07**†	**35280.36**	32.85	1.39
rat195_15_4	195	15	4	38642.85	39064.61	28.42	6.10	**36819.82**†	**37383.43**	36.28	1.53
rat195_15_5	195	15	5	50509.55	50887.27	21.76	10.09	**46222.15**†	**47960.25**	55.67	3.76
rat195_20_2	195	20	2	23322.66	23603.56	36.35	5.30	**22415.37**†	**22693.94**	40.82	1.24
rat195_20_3	195	20	3	23030.06	23401.53	32.52	5.57	**22167.59**†	**22492.86**	35.53	1.47
rat195_20_4	195	20	4	25829.03	26033.21	33.35	10.38	**23584.90**†	**24138.06**	35.78	2.35
rat195_20_5	195	20	5	33074.47	33939.28	33.70	13.65	**29862.23**†	**31450.41**	57.92	5.32
d198_05_2	198	5	2	1328469.96	1329617.73	21.96	0.20	**1326975.85**†	**1327726.99**	21.52	0.06
d198_05_3	198	5	3	1420021.52	1426846.99	32.65	1.37	**1407582.04**†	**1415681.10**	30.29	0.58
d198_05_4	198	5	4	1451141.89	1454732.22	29.69	0.54	**1446874.95**†	**1448053.57**	30.67	0.08
d198_05_5	198	5	5	1679906.52	1703564.91	28.42	3.50	**1646034.07**†	**1665196.81**	40.56	1.16
d198_10_2	198	10	2	420769.41	423034.62	31.11	1.51	**416757.28**†	**419472.03**	22.77	0.65
d198_10_3	198	10	3	452731.63	455203.91	31.04	2.72	**443165.04**†	**447387.38**	35.76	0.95
d198_10_4	198	10	4	486271.93	495696.97	32.49	6.40	**465894.26**†	**469472.69**	38.09	0.77
d198_10_5	198	10	5	607505.73	634799.05	25.89	12.49	**564338.58**†	**583682.34**	56.88	3.43

Table C.1: Continued

Instance				R-GRASP [20]				**MITS**			
Name	n	κ	τ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
d198_15_2	198	15	2	195252.95	197168.89	31.80	3.17	**191101.84**†	**192093.09**	26.74	0.52
d198_15_3	198	15	3	234961.41	242230.80	31.13	7.99	**224302.49**†	**226994.07**	39.66	1.20
d198_15_4	198	15	4	250478.91	259814.49	31.01	17.07	**221937.90**†	**233366.62**	53.87	5.15
d198_15_5	198	15	5	354733.65	370893.92	39.79	11.13	**333735.52**†	**345682.10**	56.17	3.58
d198_20_2	198	20	2	131207.01	133653.43	19.89	7.09	**124809.37**†	**127053.44**	39.85	1.80
d198_20_3	198	20	3	164516.49	166013.02	30.92	6.47	**155924.27**†	**158366.40**	50.96	1.57
d198_20_4	198	20	4	173346.54	175919.39	24.39	10.60	**159060.38**†	**162060.27**	55.96	1.89
d198_20_5	198	20	5	233014.23	241226.87	20.91	13.45	**212621.37**†	**225130.58**	56.51	5.88
kroA200_05_2	200	5	2	**2741887.18**	2742216.19	97.34	0.01	**2741887.18**	**2742143.08**	107.85	0.01
kroA200_05_3	200	5	3	2749123.77	2750233.28	109.66	0.10	**2747445.47**†	**2747782.10**	64.16	0.01
kroA200_05_4	200	5	4	2732894.66	2735041.85	105.43	0.23	**2728802.40**†	**2730848.53**	93.45	0.07
kroA200_05_5	200	5	5	2843507.61	2854179.97	177.45	1.44	**2813683.13**†	**2829820.70**	96.30	0.57
kroA200_10_2	200	10	2	919128.69	922353.84	147.53	0.84	**914701.00**†	**918670.64**	84.77	0.43
kroA200_10_3	200	10	3	953857.73	964559.40	126.62	2.15	**944231.66**†	**953144.77**	100.59	0.94
kroA200_10_4	200	10	4	980606.72	984463.80	136.57	4.22	**944617.98**†	**952199.13**	66.38	0.80
kroA200_10_5	200	10	5	1017834.17	1028105.47	79.79	6.96	**961242.24**†	**967786.97**	89.28	0.68
kroA200_15_2	200	15	2	481742.10	492097.18	160.85	2.66	**479351.19**†	**483892.88**	60.91	0.95
kroA200_15_3	200	15	3	526286.14	531225.44	179.56	5.15	**505197.53**†	**509082.19**	85.16	0.77
kroA200_15_4	200	15	4	541240.61	552915.67	107.27	6.28	**520264.84**†	**526924.88**	55.80	1.28
kroA200_15_5	200	15	5	579844.07	590830.84	160.23	12.54	**524999.04**†	**540313.61**	124.75	2.92
kroA200_20_2	200	20	2	307577.25	311499.49	127.01	3.56	**300778.17**†	**302659.19**	114.14	0.63
kroA200_20_3	200	20	3	338510.35	344602.23	126.55	7.15	**321594.80**†	**323927.50**	129.25	0.73
kroA200_20_4	200	20	4	357168.41	363095.71	157.82	10.90	**327410.31**†	**334918.42**	90.70	2.29
kroA200_20_5	200	20	5	395286.67	402103.68	190.89	16.20	**346040.38**†	**354395.09**	129.95	2.41
kroB200_05_2	200	5	2	**2739885.48**	**2739990.20**	68.22	<0.01	**2739885.48**	2750294.34	58.41	0.38
kroB200_05_3	200	5	3	2784282.11	2786391.62	87.05	0.35	**2776720.23**†	**2786256.05**	27.58	0.34
kroB200_05_4	200	5	4	2914264.92	2925579.95	201.35	1.41	**2884950.18**†	**2898139.98**	112.13	0.46
kroB200_05_5	200	5	5	2839698.34	2847079.60	158.55	0.72	**2826601.62**†	**2833828.86**	76.66	0.26
kroB200_10_2	200	10	2	898279.39	903773.81	129.18	1.18	**893257.73**†	**896237.64**	91.26	0.33
kroB200_10_3	200	10	3	916741.86	928637.97	127.38	1.75	**912667.82**†	**920427.66**	87.84	0.85
kroB200_10_4	200	10	4	1032670.79	1054400.42	135.20	5.13	**1002919.10**†	**1012215.48**	83.60	0.93
kroB200_10_5	200	10	5	996129.37	1006704.95	183.40	5.08	**958011.42**†	**966724.27**	109.14	0.91
kroB200_15_2	200	15	2	483320.44	488997.06	101.77	2.84	**475505.94**†	**482057.30**	119.01	1.38
kroB200_15_3	200	15	3	506096.37	507907.57	191.30	3.96	**488542.80**†	**492397.56**	85.22	0.79
kroB200_15_4	200	15	4	583388.84	595325.40	141.89	10.32	**539638.71**†	**551670.33**	71.90	2.23
kroB200_15_5	200	15	5	578735.66	585682.09	156.72	9.68	**533976.19**†	**539215.89**	104.35	0.98
kroB200_20_2	200	20	2	312448.16	316034.34	171.01	4.88	**301326.81**†	**304027.53**	92.33	0.90
kroB200_20_3	200	20	3	319983.78	327993.73	129.57	6.12	**309065.91**†	**313534.29**	90.71	1.45
kroB200_20_4	200	20	4	388295.32	393739.95	126.91	10.63	**355905.93**†	**363873.30**	71.48	2.24
kroB200_20_5	200	20	5	390569.34	396243.19	151.23	13.83	**348086.75**†	**355261.48**	170.65	2.06
gr202_05_2	202	5	2	34019.57	**34057.49**	183.94	0.24	**33977.27**†	34148.79	70.18	0.50
gr202_05_3	202	5	3	33514.23	33579.91	148.03	0.53	**33403.85**†	**33494.53**	88.41	0.27
gr202_05_4	202	5	4	33567.10	33592.47	95.18	0.30	**33493.54**†	**33499.68**	51.63	0.02
gr202_05_5	202	5	5	34372.21	34555.62	164.36	0.98	**34220.17**†	**34355.87**	101.57	0.40
gr202_10_2	202	10	2	12312.72	12344.55	149.86	1.63	**12146.13**†	**12196.96**	119.30	0.42
gr202_10_3	202	10	3	12398.25	12470.17	110.94	2.33	**12186.45**†	**12221.22**	118.90	0.29
gr202_10_4	202	10	4	12598.53	12681.44	117.34	2.65	**12354.47**†	**12387.39**	111.72	0.27
gr202_10_5	202	10	5	13092.30	13299.61	136.33	5.49	**12607.62**†	**12753.02**	99.27	1.15
gr202_15_2	202	15	2	6810.13	6861.98	156.79	3.09	**6656.17**†	**6723.70**	79.87	1.01
gr202_15_3	202	15	3	7136.72	7199.67	102.36	4.00	**6922.44**†	**6963.80**	117.33	0.60
gr202_15_4	202	15	4	7274.04	7318.53	174.04	5.12	**6962.31**†	**7022.56**	135.75	0.87
gr202_15_5	202	15	5	7647.11	7781.79	117.56	9.20	**7126.02**†	**7222.93**	147.05	1.36
gr202_20_2	202	20	2	4338.80	4403.06	174.67	4.19	**4226.12**†	**4260.12**	109.62	0.80
gr202_20_3	202	20	3	4764.41	4791.15	140.35	5.62	**4536.42**†	**4581.30**	117.66	0.99
gr202_20_4	202	20	4	4901.37	4922.65	131.87	7.63	**4573.57**†	**4627.44**	136.10	1.18

Table C.1: Continued

Instance				R-GRASP [20]				**MITS**			
Name	n	κ	τ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
gr202_20_5	202	20	5	5228.07	5287.39	146.13	10.90	**4767.63**†	**4829.94**	94.63	1.31
ts225_05_2	225	5	2	**16581273.69**	**16582675.68**	116.02	0.01	**16581273.69**	16589179.16	72.46	0.05
ts225_05_3	225	5	3	16975799.84	16984205.65	172.96	0.16	**16957895.27**†	**16970193.26**	95.15	0.07
ts225_05_4	225	5	4	**16705101.95**	16732458.37	179.27	0.16	**16705101.95**	**16728771.43**	36.29	0.14
ts225_05_5	225	5	5	17258172.51	17331409.45	184.01	0.93	**17170947.61**†	**17271107.49**	79.59	0.58
ts225_10_2	225	10	2	5854585.78	5866557.93	156.31	0.91	**5813932.26**†	**5837140.59**	62.88	0.40
ts225_10_3	225	10	3	6043126.76	6116381.49	152.00	2.24	**5982337.96**†	**5996838.07**	50.87	0.24
ts225_10_4	225	10	4	6021987.00	6084861.38	120.71	2.85	**5916035.88**†	**5975268.82**	91.29	1.00
ts225_10_5	225	10	5	6415910.55	6469507.55	179.38	5.30	**6143824.56**†	**6209895.12**	91.14	1.08
ts225_15_2	225	15	2	3068830.08	3084180.96	145.22	2.98	**2994793.46**†	**3025844.75**	93.63	1.04
ts225_15_3	225	15	3	3312981.37	3353777.58	172.50	5.68	**3173503.37**†	**3199031.50**	105.08	0.80
ts225_15_4	225	15	4	3396965.05	3430960.98	135.55	6.04	**3235431.93**†	**3279328.70**	61.40	1.36
ts225_15_5	225	15	5	3633916.62	3701639.10	166.84	11.71	**3313562.05**†	**3410015.61**	129.40	2.91
ts225_20_2	225	20	2	1888839.93	1903437.01	163.55	6.69	**1784052.03**†	**1811059.28**	143.87	1.51
ts225_20_3	225	20	3	2139814.45	2165589.25	122.01	9.35	**1980474.69**†	**2010773.27**	69.79	1.53
ts225_20_4	225	20	4	2213406.58	2234709.55	165.81	11.96	**1996018.18**†	**2038110.69**	110.46	2.11
ts225_20_5	225	20	5	2455240.33	2506964.68	160.63	14.56	**2188324.76**†	**2242117.78**	100.15	2.46
tsp225_05_2	225	5	2	**410912.57**	411310.50	150.01	0.10	**410912.57**	**411140.25**	45.98	0.06
tsp225_05_3	225	5	3	**405313.87**	**405450.41**	125.90	0.03	**405313.87**	405861.05	53.79	0.14
tsp225_05_4	225	5	4	**407285.17**	409976.20	142.66	0.66	**407285.17**	**408125.54**	36.45	0.21
tsp225_05_5	225	5	5	**429426.70**†	433842.91	180.88	1.03	430310.49	**431552.22**	116.80	0.49
tsp225_10_2	225	10	2	138536.38	138925.36	177.09	1.51	**136863.37**†	**137482.51**	97.95	0.45
tsp225_10_3	225	10	3	141128.78	141954.90	152.10	2.31	**138750.70**†	**139511.05**	85.55	0.55
tsp225_10_4	225	10	4	145862.12	146356.84	143.51	3.85	**140929.81**†	**141725.91**	90.92	0.56
tsp225_10_5	225	10	5	161314.95	163032.43	149.55	6.92	**152480.43**†	**155180.98**	121.80	1.77
tsp225_15_2	225	15	2	73885.70	75009.13	140.12	2.92	**72879.19**†	**73624.70**	134.26	1.02
tsp225_15_3	225	15	3	79208.31	79754.27	113.30	4.69	**76181.61**†	**77010.79**	99.87	1.09
tsp225_15_4	225	15	4	80107.47	80746.65	171.62	7.08	**75404.72**†	**76581.15**	130.49	1.56
tsp225_15_5	225	15	5	90826.99	92606.15	152.55	11.73	**82881.57**†	**84609.57**	138.74	2.08
tsp225_20_2	225	20	2	47458.03	47781.37	138.54	4.03	**45928.92**†	**46439.95**	106.25	1.11
tsp225_20_3	225	20	3	51127.68	51631.02	192.62	7.17	**48175.57**†	**48930.38**	100.45	1.57
tsp225_20_4	225	20	4	52522.80	53292.69	191.28	10.57	**48197.83**†	**49145.72**	106.16	1.97
tsp225_20_5	225	20	5	60870.89	61901.90	185.01	16.43	**53166.97**†	**54270.38**	131.19	2.08
pr226_05_2	226	5	2	**14069981.15**	**14069981.15**	6.91	0	**14069981.15**	14108559.92	114.66	0.27
pr226_05_3	226	5	3	14230609.48	**14252600.14**	186.62	0.42	**14193409.39**†	14301762.60	41.95	0.76
pr226_05_4	226	5	4	**13947467.74**	**13961683.36**	181.11	0.10	**13947467.74**	13977077.31	55.09	0.21
pr226_05_5	226	5	5	16320509.74	16473884.81	161.54	1.96	**16157418.42**†	**16290178.71**	65.22	0.82
pr226_10_2	226	10	2	3959244.03	3985454.40	160.91	2.05	**3905356.53**†	**3916972.53**	102.28	0.30
pr226_10_3	226	10	3	4119444.02	4207324.21	170.54	3.97	**4046556.07**†	**4085150.33**	133.50	0.95
pr226_10_4	226	10	4	4037524.60	4059038.53	110.58	4.32	**3890851.76**†	**3910233.54**	87.42	0.50
pr226_10_5	226	10	5	5992358.26	6106607.55	137.33	9.46	**5578780.92**†	**5667652.42**	99.36	1.59
pr226_15_2	226	15	2	2005856.44	2028067.75	142.83	5.15	**1928695.87**†	**1942938.72**	97.48	0.74
pr226_15_3	226	15	3	2144821.84	2178751.13	170.86	8.19	**2013739.57**†	**2042859.63**	93.24	1.45
pr226_15_4	226	15	4	2182534.97	2234296.84	172.98	11.62	**2001611.12**†	**2034299.12**	166.90	1.63
pr226_15_5	226	15	5	3369332.77	3423931.33	154.21	19.82	**2857546.97**†	**2984373.09**	129.56	4.44
pr226_20_2	226	20	2	1228727.69	1240144.83	127.77	6.58	**1163616.50**†	**1176314.73**	105.63	1.09
pr226_20_3	226	20	3	1281621.54	1329752.76	127.39	15.36	**1152738.77**†	**1178666.18**	114.24	2.25
pr226_20_4	226	20	4	1444585.53	1469438.16	98.92	15.91	**1267711.44**†	**1310769.35**	120.99	3.40
pr226_20_5	226	20	5	2236686.72	2263871.22	153.35	25.56	**1803048.93**†	**1902436.33**	178.46	5.51
gr229_05_2	229	5	2	166169.84	166304.24	183.93	0.51	**165455.85**†	**165687.13**	134.86	0.14
gr229_05_3	229	5	3	171807.69	172157.55	176.86	0.47	**171346.32**†	**171610.16**	94.05	0.15
gr229_05_4	229	5	4	171884.46	172170.51	124.00	0.57	**171192.72**†	**171556.88**	107.54	0.21
gr229_05_5	229	5	5	184127.55	184864.61	154.28	1.39	**182325.50**†	**182792.44**	99.62	0.26
gr229_10_2	229	10	2	57409.24	57607.06	146.11	0.89	**57098.35**†	**57127.07**	110.13	0.05
gr229_10_3	229	10	3	64342.56	64495.79	182.20	2.70	**62799.38**†	**63139.17**	151.99	0.54

Table C.1: Continued

Instance				R-GRASP [20]				**MITS**			
Name	n	κ	τ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
gr229_10_4	229	10	4	63239.84	63527.79	119.54	2.92	**61728.12**†	**61872.75**	148.62	0.23
gr229_10_5	229	10	5	70628.09	71456.48	162.91	5.90	**67476.29**†	**67951.91**	137.31	0.70
gr229_15_2	229	15	2	30328.11	30400.53	91.10	1.18	**30047.43**†	**30128.11**	102.52	0.27
gr229_15_3	229	15	3	34136.00	34525.70	163.63	5.94	**32589.21**†	**33018.64**	165.37	1.32
gr229_15_4	229	15	4	35461.37	35752.56	196.99	6.97	**33424.05**†	**33814.83**	108.70	1.17
gr229_15_5	229	15	5	39585.57	40208.09	154.46	7.87	**37274.19**†	**37717.56**	125.84	1.19
gr229_20_2	229	20	2	19417.90	19549.24	152.30	2.48	**19076.85**†	**19163.99**	128.80	0.46
gr229_20_3	229	20	3	22077.86	22323.01	110.01	5.92	**21076.20**†	**21329.73**	143.54	1.20
gr229_20_4	229	20	4	22943.67	23081.80	152.17	6.66	**21640.96**†	**21906.38**	130.24	1.23
gr229_20_5	229	20	5	25809.35	26161.68	133.17	10.69	**23635.94**†	**24069.25**	146.77	1.83
gil262_05_2	262	5	2	**314121.97**	**314192.77**	128.02	0.02	**314121.97**	314590.55	32.82	0.15
gil262_05_3	262	5	3	323062.78	323612.13	169.85	0.41	**322305.34**†	**322935.29**	117.94	0.20
gil262_05_4	262	5	4	329325.64	329751.80	158.04	0.70	**327475.02**†	**329355.15**	84.04	0.57
gil262_05_5	262	5	5	326620.06	327133.43	174.16	0.94	**324078.51**†	**326657.27**	84.69	0.80
gil262_10_2	262	10	2	107571.15	107756.76	157.01	1.04	**106646.86**†	**107509.96**	71.80	0.81
gil262_10_3	262	10	3	116502.90	117354.08	158.32	2.00	**115050.74**†	**115893.57**	143.18	0.73
gil262_10_4	262	10	4	115080.77	116493.55	160.59	3.70	**112335.22**†	**113332.22**	152.75	0.89
gil262_10_5	262	10	5	121544.30	123556.69	159.41	5.37	**117259.37**†	**118319.62**	163.92	0.90
gil262_15_2	262	15	2	57511.69	57919.95	176.40	1.84	**56874.01**†	**57416.81**	173.84	0.95
gil262_15_3	262	15	3	65155.79	65606.24	166.05	5.14	**62398.64**†	**63237.93**	118.51	1.35
gil262_15_4	262	15	4	65774.72	66751.48	133.01	7.79	**61926.17**†	**62799.15**	153.06	1.41
gil262_15_5	262	15	5	70126.09	70971.22	167.39	9.77	**64654.04**†	**65215.74**	183.43	0.87
gil262_20_2	262	20	2	36720.19	36994.06	193.43	3.58	**35716.71**†	**36042.52**	87.77	0.91
gil262_20_3	262	20	3	43272.64	43397.25	119.71	7.39	**40412.64**†	**40630.41**	141.27	0.54
gil262_20_4	262	20	4	43128.03	44076.64	139.95	10.52	**39882.44**†	**40398.68**	185.25	1.29
gil262_20_5	262	20	5	47664.85	48050.19	164.48	11.97	**42911.67**†	**43316.61**	159.57	0.94
pr264_05_2	264	5	2	9463807.89	**9509398.27**	185.07	1.63	**9357274.82**†	9697438.20	104.39	3.64
pr264_05_3	264	5	3	9330752.32	**9388589.84**	133.63	1.53	**9247332.04**†	9423249.70	83.93	1.90
pr264_05_4	264	5	4	8903385.32	**8980222.67**	127.05	1.47	**8850196.36**†	9216315.25	81.34	4.14
pr264_05_5	264	5	5	9564929.94	9605408.96	170.28	3.75	**9258415.41**†	**9380370.06**	182.64	1.32
pr264_10_2	264	10	2	2459116.26	2472811.86	152.13	4.16	**2374079.70**†	**2388831.40**	86.07	0.62
pr264_10_3	264	10	3	2644013.25	2671804.23	178.30	10.66	**2414343.64**†	**2480848.49**	151.46	2.75
pr264_10_4	264	10	4	2176416.15	2247765.11	120.76	7.92	**2082893.84**†	**2094324.05**	136.87	0.55
pr264_10_5	264	10	5	3059423.22	3158638.62	154.34	12.44	**2809239.35**†	**2959737.66**	168.88	5.36
pr264_15_2	264	15	2	1498388.13	1509439.28	162.96	3.94	**1452254.57**†	**1469048.91**	157.82	1.16
pr264_15_3	264	15	3	1484570.62	1517344.72	169.31	7.62	**1409900.04**†	**1444249.45**	160.84	2.44
pr264_15_4	264	15	4	1472627.38	1487744.50	121.92	7.08	**1389428.34**†	**1416347.01**	161.41	1.94
pr264_15_5	264	15	5	1846788.85	1873787.46	184.25	13.78	**1646845.72**†	**1705571.28**	190.15	3.57
pr264_20_2	264	20	2	888484.33	915505.40	145.91	9.79	**833867.79**†	**850440.86**	200.31	1.99
pr264_20_3	264	20	3	913026.50	937688.86	157.39	13.70	**824712.18**†	**855426.05**	169.00	3.72
pr264_20_4	264	20	4	839772.71	858467.23	140.97	15.41	**743864.30**†	**756783.68**	175.49	1.74
pr264_20_5	264	20	5	1201850.22	1236877.71	134.97	15.40	**1071845.96**†	**1094246.72**	204.72	2.09
a280_05_2	280	5	2	394091.33	394306.37	140.24	0.12	**393833.25**†	**393874.46**	89.18	0.01
a280_05_3	280	5	3	396397.86	397257.38	111.98	0.39	**395702.70**†	**395822.24**	92.78	0.03
a280_05_4	280	5	4	418813.71	420029.32	208.00	1.35	**414449.41**†	**417725.58**	131.04	0.79
a280_05_5	280	5	5	397295.83	399353.46	128.43	0.79	**396224.48**†	**397610.81**	141.62	0.35
a280_10_2	280	10	2	136452.82	**137111.00**	147.66	1.06	**135674.87**†	137240.93	159.13	1.15
a280_10_3	280	10	3	143005.16	143934.97	144.74	2.92	**139850.63**†	**140760.47**	195.81	0.65
a280_10_4	280	10	4	157335.73	158745.21	205.19	5.18	**150929.64**†	**152302.33**	160.52	0.91
a280_10_5	280	10	5	146899.58	148601.08	182.18	4.60	**142066.58**†	**144054.95**	177.88	1.40
a280_15_2	280	15	2	73108.72	73718.65	170.65	3.75	**71055.74**†	**71641.09**	110.52	0.82
a280_15_3	280	15	3	77982.95	79412.27	135.33	5.60	**75197.65**†	**76765.95**	143.14	2.09
a280_15_4	280	15	4	89462.42	90662.06	139.47	9.79	**82575.53**†	**84513.43**	186.90	2.35
a280_15_5	280	15	5	84480.68	85252.21	113.76	8.66	**78457.38**†	**79565.77**	234.95	1.41
a280_20_2	280	20	2	47535.02	47772.96	185.47	4.30	**45802.39**†	**46240.83**	128.94	0.96

Table C.1: Continued

Instance				R-GRASP [20]				**MITS**			
Name	n	κ	τ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
a280_20_3	280	20	3	51908.62	52275.87	134.15	7.93	**48436.23**†	**49009.02**	192.74	1.18
a280_20_4	280	20	4	60050.03	60702.12	164.72	12.84	**53796.87**†	**54775.90**	222.66	1.82
a280_20_5	280	20	5	57308.61	57965.06	121.08	14.49	**50628.04**†	**52021.02**	235.29	2.75
pr299_05_2	299	5	2	**8565157.38**	8574233.45	128.27	0.11	**8565157.38**	**8565166.60**	115.32	<0.01
pr299_05_3	299	5	3	8707346.30	8733631.79	138.00	0.51	**8689246.32**†	**8696756.98**	117.05	0.09
pr299_05_4	299	5	4	8959515.78	8989320.55	134.49	1.12	**8890127.52**†	**8907943.93**	146.52	0.20
pr299_05_5	299	5	5	8839104.31	8851556.01	137.23	1.07	**8757937.16**†	**8778413.74**	163.21	0.23
pr299_10_2	299	10	2	3014874.43	3022894.00	168.44	1.42	**2980431.55**†	**2991294.62**	139.48	0.36
pr299_10_3	299	10	3	3135205.15	3151958.92	110.53	2.81	**3065667.68**†	**3087768.71**	148.84	0.72
pr299_10_4	299	10	4	3165954.02	3193771.06	178.47	4.17	**3065874.54**†	**3089330.52**	186.24	0.77
pr299_10_5	299	10	5	3256111.75	3287400.06	139.52	4.44	**3147764.49**†	**3191896.29**	196.51	1.40
pr299_15_2	299	15	2	1593779.11	1602462.99	138.76	2.72	**1560014.33**†	**1567055.69**	137.36	0.45
pr299_15_3	299	15	3	1726929.80	1738147.68	151.28	5.90	**1641263.82**†	**1668480.73**	159.46	1.66
pr299_15_4	299	15	4	1722578.38	1752692.35	191.24	9.97	**1593750.54**†	**1640943.95**	223.66	2.96
pr299_15_5	299	15	5	1858785.63	1886208.19	223.61	9.33	**1725198.01**†	**1748329.15**	252.99	1.34
pr299_20_2	299	20	2	1011977.75	1027984.86	183.10	3.94	**989026.42**†	**996052.07**	180.87	0.71
pr299_20_3	299	20	3	1120640.07	1130232.64	122.63	7.40	**1052396.93**†	**1061812.43**	183.53	0.89
pr299_20_4	299	20	4	1145974.24	1155520.82	181.31	12.05	**1031247.34**†	**1052636.50**	238.47	2.07
pr299_20_5	299	20	5	1270087.26	1289693.61	188.16	15.03	**1121189.37**†	**1162564.60**	249.87	3.69
lin318_05_2	318	5	2	8041486.97	8051962.48	198.84	0.34	**8024961.29**†	**8035500.40**	125.71	0.13
lin318_05_3	318	5	3	8002347.90	8029540.50	185.66	0.54	**7986400.51**†	**8007390.87**	102.96	0.26
lin318_05_4	318	5	4	7984830.85	**8001881.95**	140.65	0.64	**7951153.49**†	8022103.86	152.31	0.89
lin318_05_5	318	5	5	8287702.05	8329484.36	167.18	1.68	**8191690.51**†	**8273314.39**	165.06	1.00
lin318_10_2	318	10	2	2678203.85	2692599.48	137.54	1.27	**2658819.37**†	**2664589.24**	200.66	0.22
lin318_10_3	318	10	3	2739427.20	2761604.56	137.33	2.67	**2689768.23**†	**2696613.33**	175.55	0.25
lin318_10_4	318	10	4	2810465.86	2845614.33	129.97	3.89	**2739166.68**†	**2754086.64**	153.39	0.54
lin318_10_5	318	10	5	2978876.71	2993633.24	147.84	5.96	**2825164.54**†	**2849988.84**	238.44	0.88
lin318_15_2	318	15	2	1485372.04	1497365.05	159.10	3.19	**1451016.38**†	**1459669.98**	194.22	0.60
lin318_15_3	318	15	3	1577109.98	1593223.88	186.45	4.32	**1527313.67**†	**1541801.67**	231.25	0.95
lin318_15_4	318	15	4	1619805.94	1638194.94	124.10	6.28	**1541342.12**†	**1554280.21**	245.00	0.84
lin318_15_5	318	15	5	1720777.36	1756376.82	125.12	10.79	**1585385.62**†	**1634440.05**	265.43	3.09
lin318_20_2	318	20	2	970163.35	975463.88	123.35	4.38	**934507.71**†	**941870.55**	247.26	0.79
lin318_20_3	318	20	3	1033338.04	1044849.41	158.92	6.16	**984262.66**†	**996458.10**	273.69	1.24
lin318_20_4	318	20	4	1074252.25	1087265.86	131.09	9.26	**995077.69**†	**1005150.71**	274.91	1.01
lin318_20_5	318	20	5	1158920.28	1183702.23	144.96	12.54	**1051793.02**†	**1073944.17**	286.95	2.11
rd400_05_2	400	5	2	3804334.81	**3809111.35**	582.48	0.26	**3799264.16**†	3814883.51	475.67	0.41
rd400_05_3	400	5	3	3844325.47	3846514.68	832.75	0.14	**3841241.69**†	**3843795.76**	486.82	0.07
rd400_05_4	400	5	4	3858069.24	3866895.28	657.11	0.57	**3844884.34**†	**3858668.68**	577.91	0.36
rd400_05_5	400	5	5	3877743.49	3887143.33	554.41	0.43	**3870569.50**†	**3874165.71**	465.98	0.09
rd400_10_2	400	10	2	1288217.71	1291381.42	587.78	0.61	**1283553.95**†	**1286021.75**	639.96	0.19
rd400_10_3	400	10	3	1323116.02	1327681.25	680.67	1.54	**1307557.81**†	**1312824.46**	754.99	0.40
rd400_10_4	400	10	4	1399501.29	1407945.29	811.57	2.52	**1373402.97**†	**1382892.23**	508.80	0.69
rd400_10_5	400	10	5	1426004.54	1432567.39	829.05	3.89	**1378929.71**†	**1382465.62**	587.22	0.26
rd400_15_2	400	15	2	724714.76	728660.84	603.75	1.66	**716734.71**†	**724477.22**	866.70	1.08
rd400_15_3	400	15	3	735749.62	743472.89	830.93	2.44	**725792.97**†	**731959.22**	458.60	0.85
rd400_15_4	400	15	4	790940.55	796883.19	885.91	5.23	**757291.05**†	**766293.30**	702.13	1.19
rd400_15_5	400	15	5	825245.22	833962.36	770.14	6.98	**779521.01**†	**787677.93**	855.91	1.05
rd400_20_2	400	20	2	469264.92	471400.32	561.62	3.15	**456991.45**†	**461109.54**	868.54	0.90
rd400_20_3	400	20	3	488638.03	492194.40	788.99	4.81	**469603.56**†	**474313.53**	768.33	1.00
rd400_20_4	400	20	4	530592.63	537258.38	918.16	9.23	**491858.78**†	**502614.04**	640.80	2.19
rd400_20_5	400	20	5	559452.21	567194.14	840.36	9.75	**516795.95**†	**521323.97**	846.90	0.88
fl417_05_2	417	5	2	4689046.10	4697967.29	843.43	0.68	**4666078.42**†	**4668808.46**	327.52	0.06
fl417_05_3	417	5	3	5341858.00	5364573.27	755.31	0.62	**5331344.57**†	**5341940.15**	411.25	0.20
fl417_05_4	417	5	4	5202478.22	5216490.25	677.35	1.23	**5153330.52**†	**5169507.58**	460.59	0.31

Table C.1: Continued

Instance				R-GRASP [20]				**MITS**			
Name	n	κ	τ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
fl417_05_5	417	5	5	5122555.34	5132265.92	719.87	1.19	**5072068.49**†	**5096562.23**	443.55	0.48
fl417_10_2	417	10	2	1363924.57	1372815.41	617.84	3.57	**1325512.11**†	**1333271.01**	670.58	0.59
fl417_10_3	417	10	3	1633220.72	1647583.95	786.49	4.51	**1576530.27**†	**1592081.07**	677.27	0.99
fl417_10_4	417	10	4	1633735.95	1671501.32	1087.83	8.80	**1536324.28**†	**1548829.85**	610.75	0.81
fl417_10_5	417	10	5	1626941.40	1641269.47	519.82	7.49	**1526878.52**†	**1542629.69**	990.27	1.03
fl417_15_2	417	15	2	683297.71	690589.01	754.51	4.30	**662132.56**†	**666663.19**	912.17	0.68
fl417_15_3	417	15	3	868683.69	884721.60	859.30	7.39	**823824.21**†	**829412.91**	829.07	0.68
fl417_15_4	417	15	4	912326.46	936195.60	814.18	15.39	**811355.48**†	**839583.31**	838.28	3.48
fl417_15_5	417	15	5	920350.75	933233.07	1081.78	11.28	**838656.31**†	**850418.06**	881.76	1.40
fl417_20_2	417	20	2	441974.58	451519.14	629.78	6.75	**422972.81**†	**427743.20**	940.35	1.13
fl417_20_3	417	20	3	575612.93	588227.57	656.66	11.96	**525391.56**†	**538983.40**	864.82	2.59
fl417_20_4	417	20	4	592240.43	602660.50	825.40	19.65	**503701.09**†	**515326.87**	995.09	2.31
fl417_20_5	417	20	5	610516.29	618671.44	786.84	13.94	**542960.73**†	**553709.75**	1078.66	1.98
gr431_05_2	431	5	2	488284.64	488677.93	595.56	0.20	**487699.69**†	**487875.76**	893.52	0.04
gr431_05_3	431	5	3	509010.44	509726.78	806.39	0.35	**507966.06**†	**508587.82**	945.01	0.12
gr431_05_4	431	5	4	495253.80	495710.29	1139.10	0.49	**493311.74**†	**493764.20**	752.35	0.09
gr431_05_5	431	5	5	516738.22	518222.45	915.94	1.70	**509560.59**†	**512260.33**	733.16	0.53
gr431_10_2	431	10	2	177338.52	177790.47	549.83	1.42	**175307.59**†	**175970.44**	916.96	0.38
gr431_10_3	431	10	3	182938.81	185325.83	754.90	1.96	**181759.14**†	**182599.93**	986.91	0.46
gr431_10_4	431	10	4	186336.67	187750.15	672.78	2.25	**183612.79**†	**184300.33**	990.35	0.37
gr431_10_5	431	10	5	201991.78	204808.09	809.71	5.83	**193527.89**†	**195721.31**	1098.62	1.13
gr431_15_2	431	15	2	97571.95	98037.67	696.51	3.38	**94836.07**†	**96066.96**	1209.24	1.30
gr431_15_3	431	15	3	103721.86	104417.03	706.24	4.17	**100236.93**†	**101377.73**	1013.59	1.14
gr431_15_4	431	15	4	107813.71	108888.09	670.91	4.86	**103840.77**†	**104928.24**	1082.49	1.05
gr431_15_5	431	15	5	116904.43	118316.89	1033.02	8.46	**109086.11**†	**110511.07**	1008.25	1.31
gr431_20_2	431	20	2	62034.62	62393.76	937.37	2.99	**60582.86**†	**61310.26**	1071.36	1.20
gr431_20_3	431	20	3	67649.51	67806.54	681.53	5.26	**64418.72**†	**65124.65**	1051.19	1.10
gr431_20_4	431	20	4	74664.89	75081.55	779.68	6.20	**70700.39**†	**71441.04**	987.96	1.05
gr431_20_5	431	20	5	77306.78	78248.25	746.97	9.50	**71462.32**†	**72941.82**	1159.49	2.07
pr439_05_2	439	5	2	35201587.67	35282734.50	926.20	0.48	**35112914.57**†	**35136801.48**	516.97	0.07
pr439_05_3	439	5	3	35445817.30	35496502.74	770.77	0.70	**35251253.01**†	**35317404.08**	415.25	0.19
pr439_05_4	439	5	4	35102987.04	35219863.14	598.77	0.40	**35078556.22**†	**35151588.51**	510.01	0.21
pr439_05_5	439	5	5	36589943.36	36666026.00	942.17	1.48	**36131778.36**†	**36286573.64**	688.28	0.43
pr439_10_2	439	10	2	12542467.67	12581522.65	576.21	0.93	**12465334.50**†	**12531777.65**	787.80	0.53
pr439_10_3	439	10	3	12892132.74	12929406.51	975.03	1.65	**12719409.95**†	**12823603.34**	790.28	0.82
pr439_10_4	439	10	4	13190627.37	13281814.71	927.88	2.48	**12960695.33**†	**13122794.67**	646.01	1.25
pr439_10_5	439	10	5	14037151.15	14239704.11	907.58	4.14	**13673667.96**†	**13895938.48**	906.73	1.63
pr439_15_2	439	15	2	6569945.02	6630670.22	734.10	2.63	**6460538.64**†	**6578361.33**	961.85	1.82
pr439_15_3	439	15	3	7048171.04	7088268.03	595.23	4.04	**6813227.01**†	**6899229.81**	941.39	1.26
pr439_15_4	439	15	4	7208795.99	7338484.85	681.62	5.95	**6926250.98**†	**7068191.63**	964.92	2.05
pr439_15_5	439	15	5	8044724.58	8093624.87	653.62	6.57	**7594451.36**†	**7692476.70**	900.02	1.29
pr439_20_2	439	20	2	4230145.30	4293782.20	501.08	3.03	**4167596.75**†	**4196976.31**	890.59	0.70
pr439_20_3	439	20	3	4506174.17	4565553.96	917.23	7.34	**4253160.30**†	**4312593.34**	1010.54	1.40
pr439_20_4	439	20	4	4728457.16	4810095.45	972.70	7.81	**4461752.38**†	**4528027.34**	954.05	1.49
pr439_20_5	439	20	5	5427274.08	5498781.21	858.10	11.06	**4951054.66**†	**5018888.41**	1076.00	1.37
pcb442_05_2	442	5	2	15446671.06	15457151.26	791.32	0.23	**15421775.93**†	**15435696.64**	632.94	0.09
pcb442_05_3	442	5	3	15304486.03	15319146.91	825.47	0.16	**15294900.24**†	**15299042.46**	767.88	0.03
pcb442_05_4	442	5	4	15782270.55	15821850.67	772.42	0.60	**15727978.08**†	**15798619.97**	738.31	0.45
pcb442_05_5	442	5	5	16041820.93	16064274.90	754.75	0.71	**15951070.41**†	**16004468.31**	670.89	0.33
pcb442_10_2	442	10	2	5286913.25	5293871.16	866.10	0.97	**5243043.92**†	**5262481.47**	921.16	0.37
pcb442_10_3	442	10	3	5327147.49	5348358.99	922.41	1.71	**5258429.66**†	**5278006.77**	777.40	0.37
pcb442_10_4	442	10	4	5670956.87	5692067.47	924.56	3.33	**5508764.16**†	**5552728.12**	892.21	0.80
pcb442_10_5	442	10	5	5834043.28	5903347.17	877.28	4.70	**5638469.30**†	**5725497.47**	1039.64	1.54
pcb442_15_2	442	15	2	2837239.12	2851396.82	581.58	2.22	**2789355.72**†	**2799595.33**	987.43	0.37
pcb442_15_3	442	15	3	2892217.06	2914924.11	783.27	3.45	**2817584.37**†	**2844708.91**	871.26	0.96

Table C.1: Continued

Instance				R-GRASP [20]				**MITS**			
Name	n	κ	τ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
pcb442_15_4	442	15	4	3170320.85	3182467.58	750.71	5.90	**3005245.88**†	**3045833.44**	881.81	1.35
pcb442_15_5	442	15	5	3408833.07	3430025.37	726.31	9.77	**3124702.06**†	**3215111.53**	956.47	2.89
pcb442_20_2	442	20	2	1860463.41	1868533.92	672.62	3.68	**1802129.43**†	**1819712.21**	911.52	0.98
pcb442_20_3	442	20	3	1893746.33	1912800.89	593.08	6.21	**1801000.30**†	**1827373.79**	894.10	1.46
pcb442_20_4	442	20	4	2112647.01	2134371.49	516.68	8.08	**1974874.81**†	**1996523.35**	873.76	1.10
pcb442_20_5	442	20	5	2308285.21	2325940.89	885.75	11.46	**2086759.59**†	**2126326.44**	956.73	1.90
d493_05_2	493	5	2	10601087.37	10608604.22	840.48	0.24	**10583140.54**†	**10591584.68**	676.04	0.08
d493_05_3	493	5	3	10562706.25	10579885.40	821.23	0.77	**10499134.03**†	**10520026.10**	650.66	0.20
d493_05_4	493	5	4	10566874.24	10585116.92	967.46	0.43	**10539408.28**†	**10581675.39**	758.07	0.40
d493_05_5	493	5	5	10740250.85	10753805.01	653.91	0.84	**10664037.73**†	**10727478.35**	858.44	0.59
d493_10_2	493	10	2	3756969.18	3772002.90	800.38	1.07	**3732182.12**†	**3762841.34**	1243.60	0.82
d493_10_3	493	10	3	3901062.17	3930900.75	698.31	2.06	**3851569.04**†	**3888319.74**	1032.19	0.95
d493_10_4	493	10	4	3763317.26	3779351.76	874.95	2.47	**3688394.26**†	**3747039.39**	1303.80	1.59
d493_10_5	493	10	5	3907367.70	3942249.08	681.13	3.28	**3817124.13**†	**3867200.46**	1366.92	1.31
d493_15_2	493	15	2	2071404.62	2078644.44	520.11	2.34	**2031196.60**†	**2044731.63**	1140.24	0.67
d493_15_3	493	15	3	2225987.07	2246527.67	828.06	3.28	**2175279.39**†	**2195897.91**	1109.77	0.95
d493_15_4	493	15	4	2113466.08	2130967.36	788.09	4.67	**2035874.07**†	**2044993.06**	1164.72	0.45
d493_15_5	493	15	5	2235003.22	2248358.53	643.11	4.90	**2143429.15**†	**2165709.39**	1348.34	1.04
d493_20_2	493	20	2	1347587.68	1361425.27	964.95	3.23	**1318803.70**†	**1331665.61**	1211.53	0.98
d493_20_3	493	20	3	1493390.94	1504066.89	782.48	4.95	**1433168.53**†	**1451240.58**	1265.29	1.26
d493_20_4	493	20	4	1423811.35	1430866.27	772.15	5.74	**1353248.13**†	**1368162.78**	1306.55	1.10
d493_20_5	493	20	5	1537108.49	1545109.79	1002.67	7.90	**1431995.54**†	**1458254.04**	1388.79	1.83

Appendix D. Detailed comparison results on the 74 benchmark instances for the BKPWC problem

Tables D.1 and D.2 provide detailed computational results of the proposed MITS algorithm, together with those of the IP-based approach [32] and the VNS algorithm [16] on the set of 50 small instances and the set of 24 large instances for the BKPWC problem. As indicated in Section 5, IP was run with a time limit of 3,600 seconds and VNS was run 20 times with a time limit of n seconds per run. For MITS, small instances were run 20 times with a time limit of n seconds, while large instances depended on two stop criteria: n and $2 \times n$ seconds. For clarity, we denote MITS with n seconds as MITS_1 and with $2 \times n$ seconds as MITS_2.

In each table, columns 1-3 show the characteristics of the instances, including their name (*Name*), the number of vertices (n), and the number of partitions (κ). For the single-run approach IP, we report the best result (*Best*) and the runtime in seconds ($t(s)$). For the multi-run algorithms VNS, MITS_1, and MITS_2, we report the same information as in Table C.1. Additionally, the 'Average' row shows the average results across the instances. As

shown in Tables D.1 on the 50 small benchmark instances, our MITS algorithm attains the 47 proven optimal solutions and matches the best-known solutions for the remaining instances. Table D.2 on the 24 large instances shows that MITS finds 21 improved best solutions, indicated by the † symbol.

Table D.1: Comparison results for the BKPWC problem on the set of 50 small instances of the proposed MITS_1 algorithm, the IP-based approach [32], and the VNS algorithm [16].

Instance			IP [32]		VNS [16]		**MITS_1**		
Name	n	κ	*Best*	$t(s)$	*Best*	$t(s)$	*Best*	*Avg*	$t(s)$
random34_1	34	6	**3669.60***	15.00	**3669.60***	0.01	**3669.60***	**3669.60**	0.11
random34_2	34	6	**3653.60***	18.00	**3653.60***	0	**3653.60***	**3653.60**	0.02
random34_3	34	6	**3559.30***	4.00	**3559.30***	0	**3559.30***	**3559.30**	0.01
random34_4	34	6	**3950.80***	24.00	**3950.80***	0.01	**3950.80***	**3950.80**	0.02
random34_5	34	6	**3973.50***	10.00	**3973.50***	0.01	**3973.50***	**3973.50**	0.17
random34_6	34	6	**3664.50***	6.00	**3664.50***	0.01	**3664.50***	**3664.50**	0.16
random34_7	34	6	**3487.80***	14.00	**3487.80***	0.01	**3487.80***	**3487.80**	0.03
random34_8	34	6	**3293.10***	8.00	**3293.10***	0	**3293.10***	**3293.10**	0.01
random34_9	34	6	**3937.40***	14.00	**3937.40***	0	**3937.40***	**3937.40**	0.01
random34_10	34	6	**3841.70***	19.00	**3841.70***	0.04	**3841.70***	**3841.70**	0.01
random38_1	38	7	**3344.40***	7.00	**3344.40***	0	**3344.40***	**3344.40**	0.04
random38_2	38	7	**3456.80***	8.00	**3456.80***	0.02	**3456.80***	**3456.80**	0.09
random38_3	38	7	**3361.70***	14.00	**3361.70***	0.01	**3361.70***	**3361.70**	0.16
random38_4	38	7	**3646.40***	14.00	**3646.40***	0.01	**3646.40***	**3646.40**	0.36
random38_5	38	7	**3384.70***	69.00	**3384.70***	0.01	**3384.70***	**3384.70**	0.01
random38_6	38	7	**3538.20***	14.00	**3538.20***	0.03	**3538.20***	**3538.20**	0.03
random38_7	38	7	**3366.40***	198.00	**3366.40***	0.01	**3366.40***	**3366.40**	0.99
random38_8	38	7	**3565.20***	11.00	**3565.20***	0.01	**3565.20***	**3565.20**	0.03
random38_9	38	7	**3029.00***	11.00	**3029.00***	0.01	**3029.00***	**3029.00**	0.02
random38_10	38	7	**3846.60***	63.00	**3846.60***	0.03	**3846.60***	**3846.60**	0.95
random44_1	44	8	**3706.10***	46.00	**3706.10***	0.01	**3706.10***	**3706.10**	0.01
random44_2	44	8	**3896.00***	1721.00	**3896.00***	0.04	**3896.00***	**3896.00**	0.04
random44_3	44	8	**3692.70***	53.00	**3692.70***	0.17	**3692.70***	**3692.70**	0.03
random44_4	44	8	**3778.70***	162.00	**3778.70***	0.88	**3778.70***	**3778.70**	0.02
random44_5	44	8	**4228.00***	894.00	**4228.00***	0.10	**4228.00***	**4228.00**	0.04
random44_6	44	8	**3844.30***	67.00	**3844.30***	0.02	**3844.30***	**3844.30**	0.37
random44_7	44	8	**3731.20***	721.00	**3731.20***	0.02	**3731.20***	**3731.20**	0.93
random44_8	44	8	**3567.70***	21.00	**3567.70***	0.03	**3567.70***	**3567.70**	0.02
random44_9	44	8	**4205.50***	262.00	**4205.50***	0.01	**4205.50***	**4205.50**	0.04
random44_10	44	8	**3893.20***	33.00	**3893.20***	0.02	**3893.20***	**3893.20**	0.04
random48_1	48	9	**3909.50***	15.00	**3909.50***	0.01	**3909.50***	**3909.50**	0.17
random48_2	48	9	**3884.70***	18.00	**3884.70***	0	**3884.70***	**3884.70**	0.02
random48_3	48	9	**3792.70***	4.00	**3792.70***	0	**3792.70***	**3792.70**	0.26
random48_4	48	9	**4264.90***	24.00	4286.20	0.01	**4264.90***	**4264.90**	0.13
random48_5	48	9	**3785.30***	10.00	**3785.30***	0.01	**3785.30***	**3785.30**	0.05
random48_6	48	9	**3701.50***	6.00	**3701.50***	0.01	**3701.50***	**3701.50**	0.10
random48_7	48	9	**4000.50***	14.00	**4000.50***	0.01	**4000.50***	**4000.50**	0.31
random48_8	48	9	**3885.90***	8.00	3896.60	0	**3885.90***	**3885.90**	0.05
random48_9	48	9	**3895.80***	14.00	**3895.80***	0	**3895.80***	**3895.80**	0.58
random48_10	48	9	**3627.20***	19.00	**3627.20***	0.04	**3627.20***	**3627.20**	0.03
random54_1	54	10	-	-	**4556.50**	0.06	**4556.50**	**4556.50**	0.04
random54_2	54	10	**3891.90***	767.00	**3891.90***	0.07	**3891.90***	**3891.90**	0.46
random54_3	54	10	**4153.00***	2402.00	**4153.00***	0.10	**4153.00***	**4153.00**	0.37
random54_4	54	10	-	-	**4385.90**	0.18	**4385.90**	**4385.90**	0.13
random54_5	54	10	**4321.10***	578.00	**4321.10***	0.09	**4321.10***	**4321.10**	0.61
random54_6	54	10	-	-	**4648.70**	0.05	**4648.70**	**4648.70**	0.04
random54_7	54	10	**4203.20***	2084.00	**4203.20***	0.08	**4203.20***	**4203.20**	0.13
random54_8	54	10	**4039.10***	3173.00	**4039.10***	0.12	**4039.10***	**4039.10**	0.09
random54_9	54	10	**4270.90***	664.00	**4270.90***	0.10	**4270.90***	**4270.90**	0.14
random54_10	54	10	**3916.50***	1247.00	**3916.50***	0.06	**3916.50***	**3916.50**	0.11
Average					**3825.62**	0.05	**3824.98**	**3824.98**	0.17

The '$*$' symbol indicates a known optimal value. The $t(s)$ value of 0 means less than 0.01 seconds.
The '-' symbol indicates that the corresponding result is not available.

Table D.2: Comparison results for the BKPWC problem on the set of 24 large instances of the proposed $MITS_1$ and $MITS_2$ algorithm, and the VNS algorithm [16].

Instance			VNS [16]				**$MITS_1$**				**$MITS_2$**			
Name	n	κ	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)	*Best*	*Avg*	$t(s)$	*Dev* (%)
RanReal960_01.30	960	30	**504467.40**†	508083.20	889.73	0.72	515385.37	516778.78	944.39	2.44	505668.68	**507658.35**	1778.43	0.63
RanReal960_01.40	960	40	352016.50	355740.50	2.78	4.03	349704.91	351443.47	947.21	2.77	**341968.83**†	**343509.24**	1737.12	0.45
RanReal960_01.60	960	60	197607.50	201326.70	2.40	8.37	194225.76	195743.06	941.46	5.36	**185783.51**†	**187957.77**	1740.49	1.17
RanReal960_05.30	960	30	**503375.40**†	**505703.40**	907.58	0.46	512864.31	515643.87	923.97	2.44	505084.61	506986.62	1752.12	0.72
RanReal960_05.40	960	40	352148.60	355192.70	3.47	4.23	348078.86	350822.22	934.27	2.95	**340785.24**†	**342581.19**	1717.73	0.53
RanReal960_05.60	960	60	197847.10	199860.70	2.33	7.79	192391.51	194502.91	932.10	4.90	**185413.82**†	**187347.19**	1691.92	1.04
RanReal960_10.30	960	30	**504844.00**†	**507805.00**	925.13	0.59	515319.54	518310.82	925.85	2.67	506261.88	508662.23	1712.73	0.76
RanReal960_10.40	960	40	353900.60	356595.30	3.32	4.62	348082.12	351731.03	940.09	3.19	**340854.81**†	**343490.78**	1683.70	0.77
RanReal960_10.60	960	60	199432.70	201272.20	3.53	7.65	194690.38	195988.52	946.20	4.82	**186976.38**†	**188077.25**	1715.69	0.59
MDG-a_21.25	2000	25	311775.00	312599.00	5.96	0.58	313286.00	314752.00	1999.89	1.27	**310791.00**†	**312033.90**	3978.98	0.40
MDG-a_21.40	2000	40	175313.00	176835.20	5.58	0.91	178485.00	179309.90	2000.51	2.32	**175237.00**†	**175876.80**	3987.46	0.37
MDG-a_21.50	2000	50	132721.00	133203.10	5.18	1.53	134829.00	136350.80	2000.72	3.93	**131200.00**†	**131944.20**	3983.98	0.57
MDG-a_25.25	2000	25	311214.00	312468.80	3.90	0.43	313775.00	314846.00	2000.72	1.19	**311143.00**†	**311748.90**	3976.66	0.19
MDG-a_25.40	2000	40	175536.00	176865.20	4.60	0.80	178522.00	180319.85	2000.79	2.77	**175463.00**†	**176215.50**	3987.87	0.43
MDG-a_25.50	2000	50	132595.00	133360.50	7.36	1.46	133830.00	134958.80	2000.36	2.67	**131444.00**†	**132220.05**	3991.04	0.59
MDG-a_30.25	2000	25	311300.00	312302.50	5.57	0.46	313404.00	315119.45	2000.35	1.37	**310871.00**†	**311794.30**	3974.91	0.30
MDG-a_30.40	2000	40	176292.00	176927.40	6.16	0.85	177745.00	180193.75	2000.65	2.71	**175440.00**†	**176143.50**	3991.08	0.40
MDG-a_30.50	2000	50	132732.00	133308.00	5.27	1.31	133895.00	136721.30	2000.47	3.91	**131582.00**†	**132637.20**	3996.39	0.80
MDG-a_35.25	2000	25	311321.00	312324.50	3.75	0.36	314104.00	315303.60	2000.63	1.32	**311202.00**†	**311758.90**	3977.25	0.18
MDG-a_35.40	2000	40	176031.00	176832.40	8.32	0.61	178355.00	179296.25	2000.52	2.02	**175752.00**†	**176334.25**	3990.37	0.33
MDG-a_35.50	2000	50	132894.00	133398.10	5.50	1.15	134238.00	135058.65	2000.35	2.41	**131881.00**†	**133022.70**	3997.04	0.87
MDG-a_40.25	2000	25	311883.00	312534.20	4.65	0.34	313743.00	314918.10	2000.58	1.11	**311468.00**†	**312118.50**	3984.78	0.21
MDG-a_40.40	2000	40	175872.00	177010.50	4.84	0.68	177476.00	180124.15	2000.59	2.45	**175823.00**†	**176590.00**	3993.82	0.44
MDG-a_40.50	2000	50	132307.00	133270.80	4.84	1.01	135016.00	136031.55	2000.68	3.10	**131943.00**†	**132628.55**	3994.23	0.52
Average			261059.41	262700.83	117.57	2.12	262560.24	264344.53	1601.81	2.75	**257918.24**	**259139.08**	3138.99	0.55

1046