

Advanced probabilistic tabu search for the fair max-min diversity problem

He Zheng^a, Anna Martínez-Gavara^a,^{*} Rafael Martí^a, Fred Glover^b

^a Department of Statistics and Operations Research, University of València, Dr. Moliner 50, Valencia, 46100, Spain

^b Entanglement, Inc., Boulder, CO 80305, USA

ARTICLE INFO

Keywords:

Max-min diversity
Fairness
Tabu search
GRASP

ABSTRACT

The concept of fairness in optimization models has opened up a wide range of practical applications, ranging from the equitable access to services in location, to team formation problems in human resources. Recently, the concept of fairness has found applications in the important area of maximizing diversity, giving rise to the Fair Max-Min Diversity (FMMD) problem which incorporates group fairness constraints. In this paper, we target this novel variant of the well-known Max-Min Diversity Problem (MMDP) that constitutes a challenge for today's solvers.

We explore two metaheuristic paradigms to develop new solution methods for the FMMD problem: Greedy Randomized Adaptive Search Procedure (GRASP) and Probabilistic Tabu Search (PTS). A computational study reveals that incorporating memory mechanisms makes PTS particularly effective in solving FMMD instances. To further enhance performance, we propose an advanced version of PTS that integrates memory-based perturbation and strategic oscillation mechanisms. Extensive computational experiments demonstrate that our advanced PTS consistently outperforms basic metaheuristics on all problems and achieves significantly superior performance relative to the exact approaches for larger problems. Additional analysis highlights the contributions of each proposed component, offering insights into the design of efficient methods for diversity optimization under fairness constraints.

1. Introduction

In this paper, we focus on a family of problems in combinatorial optimization known as diversity problems. They consist of selecting a subset of elements from a given set in a way that the overall distance among the selected elements is maximized. In this context, different objectives can be considered to evaluate the diversity of the subset, and different side constraints can be included to target specific situations (Martí and Martínez-Gavara, 2023). We consider here a challenging diversity problem that was recently proposed when including fairness in the subset selection (Moumoulidou et al., 2021).

Consider a set V containing n elements and a matrix d that records the distances between these elements. The max-min diversity problem (MMDP) focuses on selecting a subset U of k elements from V , with the goal of maximizing the minimum distance between any pair of distinct elements in U . This optimization problem finds applications in a variety of areas, such as data summarization or recommendation systems. An additional difficulty arises when elements in V can be divided into C disjoint groups $V_1, V_2, \dots, V_C$ based on specific attributes of the application (such as gender or education level when selecting individuals from a population). In this situation, if no additional constraint is included in the model, algorithmic solutions may result in

under- or over-representation of certain groups. Fig. 1 illustrates such a scenario, where the subset selected without fairness constraints tends to favor particular groups, whereas applying fairness constraints leads to a more balanced selection.

To address this issue, researchers in diversity maximization have introduced the concept of fairness constraints, giving rise to the fair max-min diversity (FMMD) problem, where the number of elements in each group has to be between some pre-established limits. We study this problem that combines the concept of diversity with the concept of fairness. As mentioned in Moumoulidou et al. (2021), combining fairness and diversity is a difficult task because it basically consists in comparing numerical and categorical attributes and requires ad hoc decisions in discretizing numerical attributes or defining a distance function involving numeric and categorical attributes. We follow the approach of these authors who proposed an optimization model that extends the max-min diversity to include a fairness constraint. Considering that this is an NP-complete optimization model, our motivation to study this problem is to obtain high-quality solutions to large instances by applying metaheuristic algorithms to solve it.

To introduce the problem formally, consider a set of elements divided into C subsets, $V = \bigcup_{c=1}^C V_c$, the FMMD problem keeps the

^{*} Corresponding author.

E-mail addresses: hzzheng65@gmail.com (H. Zheng), gavara@uv.es (A. Martínez-Gavara), rmarti@uv.es (R. Martí), Fredwglover@yahoo.com (F. Glover).

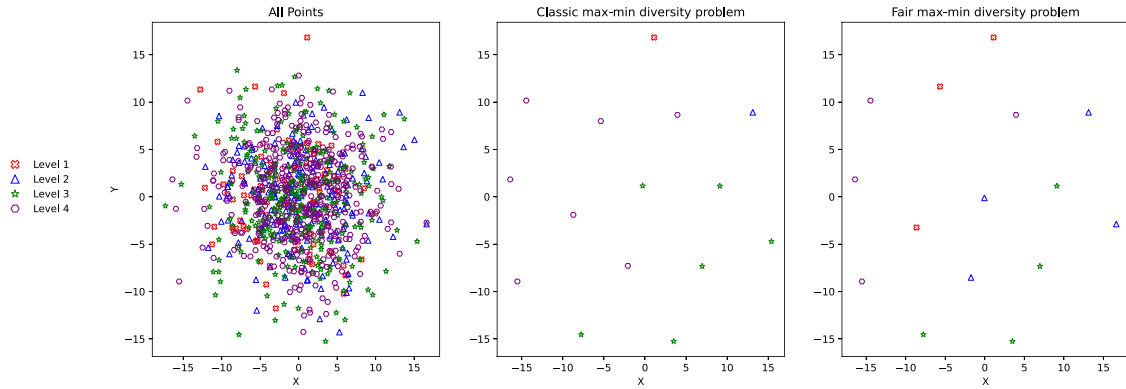


Fig. 1. Comparison of selected subsets under unconstrained and fairness-constrained settings.

objective function of the classic max-min diversity problem, identifying k elements that maximize the minimum pairwise distance between them, but includes an additional constraint. In the original formulation (Moumoulidou et al., 2021), the number of selected elements from each group is fixed, i.e., $|S \cap V_c| = k_c$ for each group c . In this work, we adopt a more general formulation in which the number of selected elements from each group V_c is allowed to lie within a predetermined range $[l_c, p_c]$, for $c \in \{1, 2, \dots, C\}$ to ensure fairness among groups. The original formulation can be obtained as a special case when $l_c = p_c = k_c$. Then, a solution U can be represented as $U = \bigcup_{c=1}^C U_c$, where each $U_c \subseteq V_c$ and the cardinality of U_c satisfies the fairness constraint: $l_c \leq |U_c| \leq p_c$. We can easily formulate the FMMD with binary variables, x_i for each $i \in V$, taking the value 1 if element i is selected, and 0 otherwise. In mathematical terms, the FMMD model is expressed as follows:

$$f(U) = \max_{i,j \in U, i \neq j} \min_{i,j \in U, i \neq j} d_{ij} x_i x_j \quad (1)$$

$$\text{s.t.} \quad \sum_{i=1}^n x_i = k \quad (2)$$

$$l_c \leq \sum_{i \in V_c} x_i \leq p_c \quad \forall c \in \{1, \dots, C\} \quad (3)$$

$$x_i \in \{0, 1\} \quad \forall i \in \{1, \dots, n\} \quad (4)$$

$$U = \{i : x_i = 1, i = 1, \dots, n\} \quad (5)$$

where d_{ij} is the distance between elements i and j , $i, j \in V$.

Previous approaches for the FMMD, as Moumoulidou et al. (2021), Addanki et al. (2022), Wang et al. (2023a,b) and Kurkure et al. (2024), focused on approximation algorithms that guarantee a certain deviation from the optimal solution. Wang et al. (2023b) proposed two algorithms based on Integer Linear Programming (ILP): an exact algorithm (FMMD-E) and a scalable approximation algorithm (FMMD-S). FMMD-E transforms the problem into finding a “Fair Independent Set” via binary search and ILP, guaranteeing optimality but with exponential complexity. FMMD-S uses a “coreset” strategy to reduce the problem size before applying the ILP, achieving a $\frac{1-\epsilon}{5}$ approximation ratio. Recently, Kurkure et al. (2024) introduced a faster algorithm to improve computational efficiency. However, it does not consistently outperform the solution quality provided by the ILP-based mechanism of Wang et al. (2023b) in practice.

We consider here a different approach, designing a metaheuristic algorithm with no approximation guarantee, but that obtains very good solutions in practice. To assess its performance, we include an exact method adapted from the binary search approach developed for the max-min diversity problem (Sayyady and Fathi, 2016). We note that this approach yields a formulation equivalent to the one used in the exact algorithm FMMD-E proposed by Wang et al. (2023b), thus providing a well-established and reliable benchmark for evaluating our heuristic.

Beyond the resolution of a given problem, we also compare two metaheuristic methodologies based on different paradigms. On the one hand, we consider GRASP (Festa and Resende, 2018) and on the other hand, we consider Probabilistic Tabu Search (PTS) (Glover and Laguna, 1998). Note that GRASP is based on random sampling, where solutions are generated by combining randomization with a greedy evaluation of the candidates. PTS is based on memory structures with a selective randomization based on probabilities according to the quality of the candidates. These two approaches, random-based and memory-based, have been extensively applied within metaheuristics, and their comparison is of interest to researchers in this field.

A preliminary version of this work was presented in Zheng et al. (2025). This paper significantly extends that work by incorporating an exact binary search model, proposing additional PTS variants, introducing advanced search mechanisms, and conducting a substantially expanded computational study. In sum, the main contributions of this paper to solving the FMMD problem are twofold:

- First, we compare two algorithmic paradigms: GRASP and Probabilistic Tabu Search (PTS). GRASP is implemented as a baseline without any memory structures, while PTS resembles GRASP during its construction phase when memory is disabled. Based on PTS principles, we propose two novel variants to solve the FMMD problem:
 - (1) PTS1 incorporates short-term and long-term memory during the improvement phase. Moves are guided by a tabu list to avoid cycling and a movement frequency memory to encourage diversification;
 - (2) PTS2 incorporates memory guidance into the construction phase, where information in elite solutions is used to bias the evaluation, and an entropy-based mechanism adaptively controls the influence of intensification based on the diversity of recently generated solutions.
 Our empirical results demonstrate that memory-based mechanisms in PTS significantly enhance performance, especially during the improvement phase.
- Second, we introduce advanced strategies aimed at obtaining high-quality solutions. A Repair-and-Relax mechanism is embedded within a Strategic Oscillation framework to control the transitions between feasible and infeasible regions. In addition, we propose a memory-based perturbation strategy, in which elements are scored based on both their frequency in the elite solutions and the number of times elements have been moved. Computational experiments demonstrate that the advanced design consistently outperforms other metaheuristics and competes favorably with exact models, especially in the large and real-world instances.

In the following section, we describe the previous approaches to the FMMD problem. Then, we describe two exact methods to solve it to optimality, a standard linear integer formulation and a binary search

based on an auxiliary problem (see Section 3). Then, in Section 4, we propose our adaptation of GRASP to the FMMD. We then describe in the following two sections the PTS algorithms for the FMMD (a basic version in Section 5, and an advanced design in Section 6). The paper concludes with our computational testing of all the methods described in the previous sections and with our associated conclusions.

2. Literature review

The main goal of this section is to review similar optimization problems in the context of location and diversity with the fairness consideration. Additionally, we also comment on the metaheuristic methodologies that are connected with the strategies applied in our solving methods.

Although fairness has been very recently considered in the context of diversity problems, we can find this concept in other optimization problems taking completely different forms. For example, in the facility location problem (Marsh and Schilling, 1994; Filippi et al., 2021; Blanco and Gázquez, 2023), fairness constraints ensure equitable access to essential services, such as healthcare centers, by minimizing the disparities in the distances customers must travel to reach these facilities. In FMMD, fairness concerns the representation of predefined groups within the selected subset, whereas distance is used to quantify diversity. By contrast, in many location problems, fairness typically refers to the distribution of costs or benefits, for example by balancing travel distances across population groups. Fairness also plays a key role in resource allocation problems (Li and Liu, 2006; Argyris et al., 2022), where the objective is to balance efficiency with fairness. These problems aim to ensure that resources are distributed in a way that maximizes overall benefits, while also ensuring that different groups receive a fair share, thereby avoiding unequal distribution. In a similar vein, the team formation problem (Machado and Stefanidis, 2019) uses fairness to ensure that skilled individuals are distributed evenly across multiple projects, preventing an over-concentration of talent in a few teams and thereby fostering more balanced team compositions.

Similarly, the fair k -center clustering problem (Chierichetti et al., 2017; Bera et al., 2022) seeks to select a set of centers such that the maximum distance between any point and its nearest center is minimized, while ensuring fairness through the equitable distribution of centers across different groups. Abbassi et al. (2013) studied the fair max-sum diversification problem (assuming disjoint groups) under matroid constraints, where the retrieved subset needs to be an independent set of a matroid of size k . Finally, in the fair variant of the knapsack problem (Patel et al., 2021), the goal is to select a subset of items from multiple categories to maximize the total value, while ensuring that the number, weight, or value of items from each category stays within a specified range, and that the total weight does not exceed the knapsack's capacity. While these problems differ from our primary focus, they underscore the growing importance of incorporating fairness into optimization problems and demonstrate its relevance across diverse domains.

The metaheuristic algorithms for the fairness variant of the max-min diversity problem still represent a significant margin of unexplored research. However, several related diversity problems provide valuable insights and methods that can help address this challenge. The maximally diverse grouping problem (MDGP) (Gallego et al., 2013; Brimberg et al., 2015; Lai and Hao, 2016; Lai et al., 2021; Yang et al., 2022; Radanović et al., 2025) and its generalized version, the capacitated clustering problem (CCP) (Martínez-Gavara et al., 2017; Brimberg et al., 2019; Zhou et al., 2019), are closely related to the fair max-min diversity problem (FMMD), particularly in terms of defining group capacities. The MDGP focuses on partitioning elements into multiple groups to maximize the diversity within each group, while the CCP aims to partition elements into clusters under a capacity constraint, with the same goal of maximizing the diversity within each cluster. Unlike FMMD, which involves selecting an optimal subset from

the entire set (divided into several disjoint groups), MDGP and CCP highlight the challenge of grouping elements effectively. Moreover, the MMDP problem and its variant, the capacitated dispersion problem (CDP) (Martínez-Gavara et al., 2021; Mladenović et al., 2022; Lu et al., 2023), share the same objective as FMMD — maximizing the minimum diversity of a subset. The methods and principles developed for these problems thus offer valuable contributions that can enhance the approaches to solving FMMD.

In terms of methodologies related to the probabilistic tabu search considered here, Glover and Laguna (1993) introduced a variety of adaptive memory strategies for tabu search that have mostly gone unnoticed in the literature. We consider some of them in the following sections, but we want to highlight their potential to be considered in future developments. These include options for creating tabu restrictions using recency-based memory (see Sections 3.2.5 and 3.2.6 in the book) and options for creating aspiration criteria (Section 3.2.7). Strategies for frequency-based memory may also be the basis of advanced metaheuristic in diversity problems. In this regard, we may consider recency, frequency weighted, or decomposed into subclasses by taking into account solution quality and move influence. A distinction is made between residence frequencies and transition frequencies, where the literature generally overlooks the relevance of residence frequencies.

In Glover (2016) we can find diversity-based and intensity-based starting methods to generate an initial set of starting points for creating clusters, which can be applied for generating starting points for a variety of other problems. Diversity-based methods come from ideas originally suggested in Glover (1994) and have been incorporated into more complex starting methods to generate starting sets of k points. The intensity-based methods of the Pseudo-Centroid Clustering approach, as well as many other parts of the approach, provide an opportunity to improve existing methods in location and diversity.

Strategic oscillation (SO) is another advanced search strategy that operates by moves defined in relation to a critical boundary that identifies regions of the search space that are expected to contain solutions of particular interest. Originally proposed in the context of tabu search as a long term strategy (Glover, 1989, 2000), it now constitutes a well-established method within the family of memory-based methodologies known as adaptive memory programming. It has produced competitive results in application fields such as binary quadratic programs in general, and maximally diverse grouping in particular (Gallego et al., 2013). In this paper, we extend the application of this strategy by considering the fairness diversity problem.

3. Mathematical models

The straightforward formulation introduced in Section 1 can be linearized to solve the problem optimally using a linear solver such as Gurobi 12.0. A standard linearization introduces auxiliary variables t and z , and is formally defined as follows (Martí et al., 2022):

$$f(U) = \max \quad t \quad (6)$$

$$\text{s.t.} \quad \sum_{i=1}^n x_i = k \quad (7)$$

$$l_c \leq \sum_{i \in V_c} x_i \leq p_c \quad \forall c \in \{1, \dots, C\} \quad (8)$$

$$z_{ij} \leq x_i \quad \forall 1 \leq i < j \leq n \quad (9)$$

$$z_{ij} \leq x_j \quad \forall 1 \leq i < j \leq n \quad (10)$$

$$z_{ij} \geq x_i + x_j - 1 \quad \forall 1 \leq i < j \leq n \quad (11)$$

$$t \leq d_{ij} z_{ij} + M(1 - z_{ij}) \quad \forall 1 \leq i < j \leq n \quad (12)$$

$$x_i \in \{0, 1\} \quad \forall i \in \{1, \dots, n\} \quad (13)$$

$$z_{ij} \in \{0, 1\} \quad \forall 1 \leq i < j \leq n. \quad (14)$$

In this formulation, t is a continuous variable representing the minimum pairwise distance among the selected elements; z_{ij} is a binary

variable that equals 1 if both elements i and j are selected (i.e., $x_i = x_j = 1$), and 0 otherwise; and M is a sufficiently large number used to deactivate the constraint when $z_{ij} = 0$.

However, due to the computational limitations of the standard formulation, we employ an alternative exact method to efficiently solve the FMMD problem to optimality. This approach is based on the node packing problem and adapts the binary search method originally developed by Sayyady and Fathi (2016) for the unconstrained max-min diversity problem. Given the original graph $G = (V, E)$, we consider a threshold value l , and define the graph $G(l)$ with the set V and the set of edges $E(l) = \{(i, j) \in E : d_{ij} < l\}$. The node packing problem consists of finding a maximum cardinality subset of nodes so that no two nodes in this subset are adjacent. Let $v(l)$ denote the optimal value of this node packing problem, that is, the maximum number of points whose pairwise distances are larger than or equal to l . The authors proposed solving a sequence of node packing problems by varying the values of l until the optimal solution of the max-min model is obtained (a set of $v(l) = k$ points). In mathematical terms, the method iteratively solves the following problem:

$$v(l) = \max \sum_{i=1}^n x_i \quad (15)$$

$$\text{s.t. } x_i + x_j \leq 1 \quad \forall (i, j) \in E(l) \quad (16)$$

$$x_i \in \{0, 1\} \quad i \in \{1, \dots, n\}. \quad (17)$$

To adapt this method to the FMMD, the group-wise constraint (8) is incorporated into the model (15)–(17) to ensure that the selected solution satisfies the group composition requirements. Although we derived this approach from the principles in Sayyady and Fathi (2016), it is important to note that this methodology aligns structurally with the FMMD-E algorithm presented by Wang et al. (2023b).

We implemented a binary search over the ordered set of different distances in the graph, in particular in the interval $l \in [d_{\min}, d_{\max}]$, where $d_{\min}$ and $d_{\max}$ are the minimum and maximum distance values, respectively. The binary search selects the subinterval to explore based on the fact that if two l values l_1 and l_2 verify that $l_1 < l_2$, $v(l_1) \geq k$ and $v(l_2) < k$, then the optimal value of the max-min problem f^* verifies $l_1 \leq f^* < l_2$. We refer the reader to Martí et al. (2022) for detail.

4. GRASP

GRASP is a well-established multi-start local search algorithm originally proposed by Feo et al. (1994). Its flexibility allows it to be effectively applied to a wide range of problems, including several diversity maximization problems, such as MMDP (Resende et al., 2010), CDP (Martínez-Gavara et al., 2021), and CCP (Martínez-Gavara et al., 2017). The success of GRASP in these contexts, along with its ability to adapt to various constraints and objective functions, makes it a promising candidate for addressing the FMMD problem. Each iteration of GRASP consists of two main phases: first, the construction of an initial solution using a randomized, greedy, and adaptive procedure, and then a local search phase that improves the solution to reach a local optimum. This process is repeated until a predefined stopping criterion is met, which is a cutoff time in the case of FMMD.

4.1. Construction method

The construction phase generates an initial solution by iteratively selecting elements according to a greedy criterion, while incorporating randomness to ensure diversity. This randomized greedy procedure builds the solution incrementally until the desired size is attained.

Specifically, let $U = \bigcup_{c=1}^C U_c$ represent a partially constructed solution, and let d_i^* denote the minimum distance of i to the selected elements (excluding itself), which is formally expressed as:

$$d_i^* = \min_{j \in U, i \neq j} d_{ij}. \quad (18)$$

This measure is crucial because it directly influences the objective function value, denoted as $f(U)$, which is the minimum d_i^* among the selected elements, in mathematical terms expressed as

$$f(U) = \min_{i \in U} d_i^*. \quad (19)$$

It is clear that elements with larger d_i^* are more likely to contribute favorably to the overall solution quality. To facilitate the selection of such elements, we further define $\max d_i^*$ as the maximum d_i^* among the candidate elements. Note that the candidate elements do not include all unselected elements due to the fairness constraint.

Initially, the solution set U is initialized as $\emptyset$. In the first step of our construction procedure, the algorithm selects an element u at random, then $k - 1$ successive addition steps are applied to incrementally build a complete solution.

To ensure the feasibility of the solution, candidate elements at each step are determined based on group-specific criteria, following strategies similar to those in Gallego et al. (2013) and Yang et al. (2022). Specifically, during the first $\sum_{c=1}^C l_c$ steps, the candidate set of elements is composed of the unselected elements from each group V_c that currently has fewer than l_c elements included in U . This phase ensures that the minimum required number l_c of elements is selected from each group. In the remaining $k - \sum_{c=1}^C l_c$ steps, the candidate set consists of the union of the unselected elements from groups where the number of selected elements remains below the upper bound p_c .

At each iteration, all candidate elements satisfying the condition $d_i^* \geq \alpha \max d_i^*$ are included in the Restricted Candidate List (RCL). The parameter $\alpha \in [0, 1]$ controls the trade-off between randomness and greediness in the construction process. Higher values of α result in a more selective RCL, favoring candidates with better objective contributions, while lower α values increase the level of randomness by allowing a broader range of candidate elements.

Once the RCL is constructed, an element i' is randomly selected from it and added to U . The objective function value of the updated partial solution is then set to the minimum of $d_{i'}^*$ and $f(U)$, where $f(U)$ denotes the objective function value before the insertion. Additionally, for all elements $i \in V$, the d_i^* value is adjusted to the lesser of d_i^* and $d_{i'}^*$ when $i \neq i'$; otherwise, it remains unchanged.

4.2. Local search procedure

Once an initial solution U is constructed, the algorithm applies a best-improvement local search strategy to refine it. In each iteration, the algorithm exhaustively explores the entire neighborhood to identify the most beneficial move. This process is repeated until no improving move can be found, that is, when all potential moves would deteriorate the current objective function value.

4.2.1. Move operator and neighborhood structure

To maintain the cardinality constraint of selecting exactly k elements, we define the $\text{Swap}(i, j)$ operator as the movement where the element i is removed from the solution and j is inserted. Given the current solution U , for all $i \in U$, we compute the corresponding d_i^* values. According to the definition of the objective function, removing an element i such that $d_i^* = f(U)$ is likely to improve the solution, since the objective function value will either increase or remain unchanged, while reducing the number of elements attaining the current minimum value.

This strategy allows us to focus on the elements that most significantly limit the solution's quality, a technique also employed in related dispersion problems (Sánchez-Oro et al., 2025; Martínez-Gavara et al., 2021; Resende et al., 2010). Instead of evaluating all elements in U , our approach restricts the set of removal candidates in each iteration to a smaller subset X , defined as

$$X = \{i \in U \mid d_i^* = f(U)\}, \quad (20)$$

i.e., those elements currently contributing to the minimum pairwise distance in the solution.

Taking the fairness constraint into account, the set of elements that can be swapped with element $i \in X$, denoted as Y_i , depends on the selection conditions of the group to which i belongs. Let $\hat{c}$ represent the group of element i . If the number of selected elements in group $\hat{c}$ is equal to its lower bound, removing i would violate the constraint. In this case, i can only be swapped with unselected elements within the same group $\hat{c}$. Otherwise, if the number of selected elements in group $\hat{c}$ exceeds its lower bound $l_{\hat{c}}$, then i can be swapped with either an unselected element from $\hat{c}$, or from any other group $c \in C$ for which the number of selected elements has not yet reached the upper bound p_c . Formally, the group candidate list for element i is defined as $GCL_i = \{\hat{c}\} \cup \{c \in C \mid c \neq \hat{c}, |U_c| < p_c\}$, and the corresponding set of swap candidates Y_i is expressed as:

$$Y_i = \begin{cases} \bigcup_{c \in GCL_i} (V_c \setminus U_c) & \text{if } |U_{\hat{c}}| > l_{\hat{c}} \\ V_{\hat{c}} \setminus U_{\hat{c}} & \text{otherwise.} \end{cases} \quad (21)$$

The neighborhood $N(U)$ induced by the $Swap(i, j)$ moves involving elements from X and Y is defined as:

$$N(U) = \{U' \mid U' = U \setminus \{i\} \cup \{j\}, i \in X, j \in Y_i\}. \quad (22)$$

To assess the impact of a potential $Swap(i, j)$ operation, where $i \in X$ and $j \in Y_i$, we define the evaluation function $\Delta_{i,j}$ as the difference between the minimum distance from element j to the set $U \setminus \{i\}$ and d_i^* . This evaluation function guides our search strategy by replacing an element i with a low d_i^* value with an element j with a larger minimum distance to the remaining solution elements. Then, we can potentially improve the overall diversity of the solution.

4.2.2. Streamlined calculations

To reduce the computational complexity associated with evaluating the impact of a swap move, specifically computing the evaluation value $\Delta_{i,j}$ and updating the objective value, we use a streamlined calculation strategy tailored to the FMMD problem. For each element i , we define the set A_i , which contains the neighbors of element i whose distance to it is equal to d_i^* . This structure enables selective recalculation of minimum distances and objective values only when necessary.

In particular, the minimum distance from j to the updated solution $U \setminus \{i\}$ is recomputed only if two conditions are met: (1) $i \in A_j$ and (2) $|A_j| = 1$. In other words, recalculation only occurs when i is the sole element in U with $d_{ij} = d_j^*$. If these conditions are not satisfied, the original d_j^* value remains unchanged.

The auxiliary sets A_i also facilitate efficient data updates following a swap move. This includes recalculating each element's minimum distance d_i^* , updating the corresponding sets A_i , recomputing the objective function value $f(U')$, and identifying the new set of critical elements X for subsequent iterations. Algorithm 1 illustrates the update procedure of the relevant data involved after performing a swap move.

4.2.3. Complexity analysis

The computational efficiency of the local search is primarily governed by the neighborhood exploration and the update procedure. In each iteration, we evaluate the neighborhood $N(U)$, which involves up to $|X| \cdot \max_{i \in X} |Y_i|$ potential moves. For each move, the evaluation function $\Delta_{i,j}$ can be computed in $O(1)$ time by leveraging the current d_i^* and A_i structures. Following a swap move, the data update procedure (Algorithm 1) is performed. The update of distances d_i^* and sets A_i for all $i \in V$ requires $O(n)$ time in the worst case, as we iterate through all elements. Consequently, the worst-case time complexity per iteration is $O(|X| \cdot \max_{i \in X} |Y_i| + n)$. Note that under this streamlined calculation, a full recomputation of d_i^* is not performed for every element after an accepted swap. Instead, recomputation is triggered only for elements whose unique nearest selected element is the one removed by the swap. As a result, most d_i^* values are preserved or updated with constant-time checks, avoiding exhaustive recomputation in practice and yielding faster runtimes.

Algorithm 1 Update the relevant data after performing a swap move

Require: Current solution U ; swap elements i' (out) and j' (in); arrays or sets d_i^* , A_i ($\forall i \in V$), and X .
Ensure: Updated solution U' ; Updated objective function value $f(U')$; Updated arrays or sets d_i^* , A_i $\forall i$, and X .

```

1:  $U' \leftarrow U \setminus \{i'\} \cup \{j'\}$  ▷ Update the solution
2: for  $i \in V$  do ▷ Update  $d_i^*$  and  $A_i$   $\forall i$ 
3:   if  $d_{ij'} < d_i^*$  and  $i \neq j'$  then
4:      $d_i^* = d_{ij'}$ 
5:      $A_i \leftarrow \{j'\}$ 
6:   else if  $d_{ij'} > d_i^*$  then
7:     if  $i' \in A_i$  and  $|A_i| = 1$  then
8:        $d_i^* \leftarrow \min\{d_{ij} \mid j \in U'\}$ 
9:        $A_i \leftarrow \{j \in U' \mid d_{ij} = d_i^*\}$ 
10:    else if  $i' \in A_i$  and  $|A_i| > 1$  then
11:       $A_i \leftarrow A_i \setminus \{i'\}$ 
12:    else if  $d_{ij'} = d_i^*$  then
13:      if  $i' \in A_i$  then
14:         $A_i \leftarrow A_i \setminus \{i'\} \cup \{j'\}$ 
15:      else
16:         $A_i \leftarrow A_i \cup \{j'\}$ 
17:  $f(U') \leftarrow \infty$  ▷ Update  $f(U')$  and  $X$ 
18: for  $i \in U'$  do
19:   if  $f(U') > d_i^*$  then
20:      $f(U') = d_i^*$ 
21:      $X \leftarrow \{i\}$ 
22:   else if  $f(U') = d_i^*$  then
23:      $X \leftarrow X \cup \{i\}$ 
return  $U'$ ,  $f(U')$ ,  $d_i^*$ ,  $A_i$  ( $\forall i \in V$ ), and  $X$ 

```

5. Probabilistic Tabu Search

Although GRASP is traditionally designed as a memoryless multi-start method that repeatedly generates solutions using a randomized greedy heuristic followed by local search, its flexibility has allowed the integration of memory-based ideas in various modern extensions. For example, Reactive GRASP (Prais and Ribeiro, 2000) adjusts its greediness parameter dynamically based on the performance of past iterations, while Cost-Perturbation GRASP (Ribeiro et al., 2002) modifies element evaluations using history-driven noise to promote solution diversity. These enhancements illustrate that GRASP can, in certain forms, incorporate mechanisms originally proposed in probabilistic tabu search (PTS). In fact, when PTS is implemented without memory and restricted to a constructive framework, it closely resembles GRASP (Fleurent and Glover, 1999). However, PTS distinguishes itself by explicitly integrating memory structures and probabilistic decision rules, allowing it to guide the search more effectively.

PTS is an extension of the classical Tabu Search (TS), both of which follow the same basic search process (Glover, 1989). In TS, the algorithm selects the best possible move in each iteration based on a defined evaluation function, while restricting certain moves via a tabu list to avoid cycling. PTS enhances this process by incorporating a probabilistic mechanism, allowing the algorithm to accept non-optimal moves with a certain probability. This mechanism enables PTS to explore a wider range of solutions, helping it escape local optima and diversify the search.

The concept of PTS has been successfully applied in both the construction and improvement phases, demonstrating promising performance across diverse applications. For the construction phase, Fleurent and Glover (1999) proposed a variant of the PTS multi-start approach for the quadratic assignment problem (QAP), where the improvement phase follows a standard descent procedure. In their approach, memory-based information is leveraged to bias the evaluation of candidate moves, and these biased evaluations are then mapped

into selection probabilities. This implementation of adaptive memory search processes significantly enhances the effectiveness of multi-start algorithms, including GRASP.

In the improvement phase, various strategies for integrating probabilistic mechanisms have been explored in the literature, often yielding better performance than standard tabu search. Xu et al. (1996) proposed a PTS approach for the network design problem in the telecommunications industry. In this method, the tabu restrictions and aspiration criteria are incorporated as penalties and incentives to modify the standard evaluations. The adjusted evaluations are subsequently transformed into probabilities that bias the selection toward high-quality solutions while simultaneously introducing diversification into the search process. Mori and Muroi (2011) proposed a PTS method for the restoration of distribution network services. To reduce the computational effort of evaluating all neighboring solutions, their approach uses uniform random numbers to select a subset of candidates from the tabu search neighborhood, instead of evaluating all possible moves. Similarly, Kulturel-Konak (2012) applied PTS to the facility layout problem and introduced a more adaptive mechanism: the acceptance probability of admissible moves is dynamically adjusted based on the recent improvement history of the best-found solution. Wang and Zhao (2022) proposed a PTS-based algorithm for the container liner shipping problem, in which the values stored in the tabu list are used directly as weights in a roulette wheel selection mechanism. In this approach, candidate moves with higher tabu values, indicating that they have not been selected recently, are assigned greater selection probabilities, while the recently chosen moves are given lower weights.

The broad applicability of PTS motivates us to develop two algorithmic variants for the FMMD problem. The key difference between them lies in the phase where the PTS principles are applied. The first variant, PTS1, employs a greedy construction phase followed by a PTS-based improvement procedure, as presented in Section 5.1. The second variant, PTS2, follows a multi-start framework in which PTS principles are integrated directly into the construction phase to generate diverse initial solutions. Each constructed solution is then refined by a conventional local search, as described in Section 5.2. For solving the FMMD problem, in PTS variants, we adopt the same move operator $Swap(i, j)$ and neighborhood structure presented in Section 4.2.1.

While most implementations rely on adaptive probabilities or complex evaluation mechanisms, we adopt a simpler yet effective selection strategy proposed by Guemri et al. (2019), and extend it, for the first time, to both the construction and improvement phases of the FMMD problem.

5.1. PTS1

5.1.1. Key components in TS

To better understand the enhancements introduced by PTS, it is essential to first review the fundamental components of TS. At its core, TS operates through an iterative process where the algorithm identifies a set of candidate moves and selects the best move based on an evaluation function. To prevent cycling and promote exploration, TS incorporates several key components (Glover, 1989; Xu et al., 1996; Martínez-Gavara et al., 2021): short-term memory, which restricts recently performed moves; long-term memory, which encourages diversification by guiding the search toward less explored regions; and aspiration criteria, which allow tabu moves to be accepted if they lead to a significant improvement. These components work together to balance intensification and diversification in the search process.

Short-term memory prevents the immediate reversal of recently performed moves, reducing the likelihood of cycling within the search space. Specifically, when a move $Swap(i, j)$ is performed, its reverse move $Swap(j, i)$ is temporarily designated as forbidden (tabu) for the next γ iterations, a period known as the tabu tenure. It is well documented in the tabu search metaheuristic literature that a variable tabu tenure is more robust than a fixed one (Xu et al., 1996; Benlic

and Hao, 2013; Martínez-Gavara et al., 2021). Accordingly, the tabu tenure in this study is randomly selected from a predefined range. To track the tabu status of moves, we maintain a tabu list H , initialized with zeros. Upon executing $Swap(i, j)$, we update the list as $H[i][j] \leftarrow iter + \gamma$, where $iter$ denotes the current iteration number. A move remains prohibited until the iteration count surpasses its stored value, encouraging exploration by preventing immediate revisits to previous solutions.

The aspiration criterion provides flexibility in the search by allowing tabu moves under specific conditions. If a tabu move results in a solution that improves upon the best found so far, the tabu restriction is overridden, and the move is accepted. Consequently, a move is considered eligible if it is either non-tabu or satisfies the aspiration condition, ensuring that high-quality solutions are not discarded due to short-term restrictions.

Long-term memory is designed to guide the search toward unexplored regions and to enhance diversification. To this end, we employ a frequency-based memory structure L , which records the historical occurrence of swaps between elements. Each time $Swap(i, j)$ or $Swap(j, i)$ is performed, the corresponding entry in the frequency matrix is updated as $L[i][j] \leftarrow L[i][j] + 1$. Frequently used swaps incur a penalty in the evaluation function to discourage repetitive moves. However, following Chiang and Kouvelis (1996), this penalty is applied only to non-improving moves, ensuring that promising solutions are not unnecessarily penalized. This mechanism prevents the search from being trapped in a limited region while still allowing promising moves to be accepted without additional constraints.

5.1.2. Main framework of PTS1

PTS introduces probabilistic mechanisms to improve the search. More precisely, instead of strictly selecting the best admissible move, PTS occasionally allows suboptimal moves to be accepted. This probabilistic approach not only increases the flexibility of the search but also helps PTS escape deep local optima that might trap classical TS. By integrating the probabilistic mechanism with the core components of TS, PTS achieves a more robust and adaptive search strategy.

The main process of the PTS1 for solving the FMMD problem is outlined in Algorithm 2. It is well established that starting with a good initial solution improves the performance of PTS (Guan et al., 2018; Guemri et al., 2019; Wang and Zhao, 2022). Therefore, PTS1 starts with a greedily constructed initial solution U (line 1) and records it as the initial best solution U^* (line 2). The search continues until a predefined stopping criterion is met (line 3), which in this case is a fixed time limit. At each iteration, PTS1 identifies a swap pair (i, j) (line 4) according to the rule defined in Algorithm 3. The selected swap is then applied to update the solution (line 5), and the tabu list H along with the swap frequency matrix L are updated accordingly (line 6–7). If the new solution improves upon the best solution found so far, it is recorded as the new best solution (lines 8–9).

In Algorithm 3, the move candidate list MCL is generated by identifying all feasible swap pairs (i, j) , as defined in Section 4.2, that are either non-tabu or satisfy the aspiration criterion (lines 1–5). A move selection strategy proposed by Guemri et al. (2019) is then applied to probabilistically choose a swap from MCL (lines 6–14). Specifically, PTS1 first sorts the candidates in descending order according to the evaluation function $\Delta_{i,j}$, incorporating the swap frequency penalty L (line 6). Then, a random number p is generated to determine the selection mechanism (line 7): with probability $prob_1$, the best candidate is chosen (line 9); with probability $prob_2 - prob_1$, a candidate is randomly selected from the top β -fraction of MCL (lines 11–12); otherwise, a completely random selection is made from the entire candidate list (line 14).

Algorithm 2 PTS1($prob_1, prob_2, \beta$)

Require: Graph $G(V, E)$; selection probabilities $prob_1$ and $prob_2$; parameter β determining the fraction of top candidates to consider.

Ensure: The best solution found (U^*).

```

1:  $U \leftarrow \text{GREEDYCONSTRUCTION}()$ 
2:  $U^* \leftarrow U$  ▷ Record the best solution
3: while ( $\text{current\_time} - \text{start\_time}$ ) <  $\text{time\_limit}$  do
4:    $(i, j) \leftarrow \text{DETERMINEMOVE}(X, Y, prob_1, prob_2, \beta)$  ▷ See details in Algorithm 3
5:    $U \leftarrow \text{UPDATESOLUTION}(i, j)$  ▷ See details in Algorithm 1
6:    $\text{UPDATETABULIST}(H)$ 
7:    $\text{UPDATESWAPFREQUENCY}(L)$ 
8:   if  $f(U) > f(U^*)$  then
9:      $U^* \leftarrow U$ 
return  $U^*$ 

```

Algorithm 3 Determine the move $\text{DETERMINEMOVE}(X, Y, prob_1, prob_2, \beta)$

Require: Removal candidates X , addition candidates Y , selection probabilities $prob_1$ and $prob_2$, and parameter β determining the fraction of top candidates to consider.

Ensure: A swap pair (i, j) .

```

1:  $MCL \leftarrow \emptyset$  ▷ Candidate list construction in PTS
2: for  $i \in X$  do
3:   for  $j \in Y$  do
4:     if  $\text{Swap}(i, j)$  is non-tabu or satisfies the aspiration criterion then
5:        $MCL \leftarrow MCL \cup \{(i, j)\}$ 
6: Sort  $MCL$  in decreasing order based on the evaluation function  $\Delta_{i,j}$ , incorporating the swap frequency penalty  $L$ . ▷ Move selection mechanism
7: Generate a random number  $p = \text{random}(0, 1)$ 
8: if  $p \leq prob_1$  then
9:   Select the optimal candidate  $(i, j)$  from  $MCL$ .
10: else if  $p \leq prob_2$  then
11:   Let  $b = \beta \times |MCL|$ 
12:   Randomly select  $(i, j)$  from the top- $b$  candidates in  $MCL$ .
13: else
14:   Randomly select  $(i, j)$  from the entire candidate list  $MCL$ .
return  $(i, j)$ 

```

5.2. PTS2

Whereas PTS1 applies PTS as an improvement strategy, PTS2 employs the probabilistic mechanism during the construction phase within a multi-start framework. In each iteration, PTS2 constructs a solution through k consecutive addition steps. This construction phase is guided by strongly determined and consistent variables identified from a set of R previously visited elite solutions, denoted as $EliteSol$. These variables form the core of an intensification strategy that biases the evaluation function, promoting the inclusion of promising elements that frequently appear in high-quality solutions. For a detailed explanation, we refer the reader to [Fleurent and Glover \(1999\)](#).

The overall framework of the proposed PTS2 procedure is summarized in Algorithm 4. The algorithm begins by initializing the elite solution pool $EliteSol$ (line 1) and setting the best solution U^* to an empty solution (line 2). The search then proceeds iteratively until the stopping criterion is met (line 3–12). At each iteration, a solution U is constructed from scratch (line 4) through k consecutive selection steps (line 5–8). During each step, the evaluation score E_i is computed for all candidate elements $i \in V \setminus U$ (line 6). Based on these scores, a candidate element is selected according to the probabilistic rule (line 7), and added to the current solution (line 8). Once a complete solution

is obtained, a local search procedure is applied to improve its quality (line 9). The resulting solution is then used to update the elite solution pool $EliteSol$ (line 10). If the new solution improves upon the best solution found so far, the best solution is updated accordingly (lines 11–12). After the stopping condition is reached, the best solution U^* is returned.

Algorithm 4 PTS2($prob_1, prob_2, \beta$)

Require: Graph $G(V, E)$; selection probabilities $prob_1$ and $prob_2$; parameter β determining the fraction of top candidates to consider.

Ensure: The best solution found (U^*).

```

1:  $EliteSol \leftarrow \emptyset$  ▷ Initialize elite solution set
2:  $U^* \leftarrow \emptyset$ 
3: while ( $\text{current\_time} - \text{start\_time}$ ) <  $\text{time\_limit}$  do
4:    $U \leftarrow \emptyset$  ▷ Construction phase
5:   for  $t = 1$  to  $k$  do
6:     Compute  $E_i$  for all candidates  $i \in V \setminus U$ 
7:     Select candidate  $i$  probabilistically based on  $prob_1, prob_2, \beta$ 
8:      $U \leftarrow U \cup \{i\}$ 
9:   Apply local search to  $U$  ▷ Improvement phase
10:  Update  $EliteSol$  with  $U$ 
11:  if  $f(U) > f(U^*)$  then
12:     $U^* \leftarrow U$ 
return  $U^*$ 

```

Note that in the FMMD problem, we retain only solutions with unique objective values in the elite pool ([Sánchez-Oro et al., 2025](#)). The update mechanism of the elite pool primarily relies on a similarity measure, defined as the proportion of common elements between two solutions relative to the total number of selected elements. If the similarity between a new solution and all existing elite solutions is below a predefined maximum threshold ζ , and a solution with the same objective value already exists in the pool, the new solution is accepted with a probability of 50%. Otherwise, if the similarity criterion is satisfied and the objective value is unique, the new solution replaces the worst-performing solution in the pool.

To guide the construction, a new evaluation E_i is defined for each candidate i as:

$$E_i = \lambda \cdot \text{value}_i + \text{intensity}_i. \quad (23)$$

Here, value_i represents the normalized quality of candidate i based on its contribution to the objective function value:

$$\text{value}_i = \frac{d_i^*}{\max d^*}, \quad (24)$$

where d_i^* and $\max d^*$ are formally defined in Section 4.1. To take into account both the frequency with which element i appears in elite solutions and the quality of these solutions, we define intensity_i as follows:

$$\text{intensity}_i = \sum_{U \in EliteSol: i \in U} \frac{f(U)}{f(U^*)}, \quad (25)$$

where $f(U)$ denotes the objective value of elite solution U , and $f(U^*)$ is the best value among all elite solutions.

The intensification factor emphasizes solution quality at the expense of randomness. To balance diversification and intensification, a dynamically adjusted parameter λ is introduced. This parameter is updated based on the diversity of the most recent 100 constructed solutions, using an entropy-based measure defined as:

$$e = \frac{\sum_{i=1}^n [n_i \log n_i + (1 - n_i) \log(1 - n_i)]}{n \left[\frac{k}{n} \log \frac{k}{n} + \left(1 - \frac{k}{n}\right) \log \left(1 - \frac{k}{n}\right) \right]}, \quad (26)$$

where n_i denotes the frequency with which element i appears in the last 100 constructed solutions. This entropy measure $e \in [0, 1]$ quantifies the diversity of recent solutions: $e = 0$ implies that all solutions are identical, while $e = 1$ indicates uniformly distributed selection.

Following [Fleurent and Glover \(1999\)](#), every 100 constructed solutions, e is recalculated and used to adjust λ : if $e > 0.5$, λ is decreased by R to encourage intensification; otherwise, λ is increased by R to emphasize diversification.

Once the E_i values are computed for all candidates, the selection proceeds in a probabilistic manner. Specifically, with probability $prob_1$, the candidate with the highest E_i value is selected, promoting intensification by favoring the most promising option. With probability $prob_2 - prob_1$, an element is randomly selected from the top β candidates ranked by E_i , introducing moderate diversification. Finally, with the remaining probability, a candidate is selected uniformly at random from the entire candidate list, ensuring global exploration.

6. Advanced probabilistic Tabu Search

Building upon the PTS-based local search framework (PTS1), we propose an enhanced version that integrates strategic oscillation and memory-based perturbation. While PTS1 already offers effective local search capabilities, these two mechanisms further improve its ability to escape local optima and discover higher-quality solutions. Strategic oscillation facilitates the controlled exploration of feasible and infeasible solution regions by relaxing and repairing fairness constraints, thereby uncovering potentially better solutions. In parallel, memory-based perturbation employs two frequency-based memory components to diversify the search, allowing the algorithm to escape local optima and effectively explore promising regions of the solution space.

The overall procedure of the Advanced-PTS method is summarized in Algorithm 5. The algorithm begins with a greedy construction phase to generate an initial solution U (line 1), which is also recorded as the current best solution U^* (line 2). Then, the algorithm enters an iterative loop that runs until a predefined time limit is reached (line 4–6). In each iteration, the algorithm first applies the `RELAXREPAIR` procedure, which implements strategic oscillation by temporarily relaxing fairness constraints and then repairing them. Following this, a memory-based perturbation (line 5–6) is applied, thereby generating a new starting point for the next iteration.

Algorithm 5 Advanced PTS `ADVANCED-PTS(prob1, prob2, $\beta$ , $\eta$ )`

Require: Graph $G(V, E)$; selection probabilities $prob_1$ and $prob_2$; parameter β determining the fraction of top candidates to consider; parameter η to set the jump magnitude.

Ensure: The best solution found (U^*).

```

1:  $U \leftarrow \text{GREEDYCONSTRUCTION}()$ 
2:  $U^* \leftarrow U$  ▷ Record the best solution
3: while ( $\text{current\_time} - \text{start\_time}$ ) <  $\text{time\_limit}$  do
4:    $\text{RELAXREPAIR}(U, U^*)$  ▷ See details in Algorithm 6
5:    $U \leftarrow \text{RANDOMSELECT}(\text{EliteSol})$  ▷ See description in Section 6.2
6:    $U \leftarrow \text{PERTURBATION}(U, \eta)$ 
return  $U^*$ 

```

6.1. Strategic oscillation

In the FMMD problem, only removing critical elements can affect the objective value. However, enforcing fairness constraints significantly narrows the search space by limiting the set of allowable element exchanges, particularly in instances involving many groups.

To address this limitation, we relax the original fairness constraints. Instead of requiring that each group c contain exactly between l_c and p_c selected elements, we allow the number of selected elements in each group to vary within a relaxed range of $l_c - \epsilon \leq |U_c| \leq p_c + \epsilon$, while ensuring that the total selection size remains exactly k . This relaxation increases flexibility in the search process and allows the exploration of a broader solution space, potentially finding better solutions.

In the FMMD problem, swapping non-critical elements within a solution does not affect the objective value as long as the critical elements remain unchanged. Based on this insight, we hypothesize that

high-quality infeasible solutions may reside near feasible ones with comparable objective values.

These two observations inspire our Strategic Oscillation approach ([Glover, 1989](#)). In this method, the oscillation between feasible and infeasible solutions is controlled by the relax and repair operations. By relaxing fairness constraints, we can explore high-quality solutions that would otherwise be unreachable within the strictly feasible region. Since these solutions may violate fairness constraints, the repair operation is applied to restore feasibility while preserving solution quality. This controlled oscillation between feasible and infeasible regions allows the search to escape local optima and discover better solutions. Notably, similar strategies have demonstrated effectiveness in solving the swap-body vehicle routing problem ([Absi et al., 2017](#)) and the maximally diverse grouping problem ([Gallego et al., 2013](#)).

Algorithm 6 Relax-and-Repair `RELAXREPAIR( $U, U^*$ )`

Require: Initial solution (U), best solution found (U^*)

Ensure: Updated solutions U, U^* .

```

1:  $\text{ConR} = 0, \text{NonImp} = 0$ 
2:  $U' \leftarrow U$ 
3: while  $\text{NonImp} < \text{MaxNI}$  do
4:    $(i, j) \leftarrow \text{DETERMINEMOVE}(X, Y, prob_1, prob_2, \beta)$  ▷ See details in Algorithm 3
5:    $U \leftarrow \text{UPDATESOLUTION}(i, j)$  ▷ See details in Algorithm 1
6:   if  $f(U) > f(U')$  then
7:      $U' \leftarrow U, \text{NonImp} = 0$ 
8:      $U \leftarrow \text{REPAIR}(U)$ 
9:   else
10:     $\text{NonImp} = \text{NonImp} + 1$ 
11:    if  $\text{ConR} > \text{MaxCR}$  then
12:       $U \leftarrow \text{REPAIR}(U)$ 
13:    if  $U$  is feasible then
14:       $\text{ConR} = 0$ 
15:      if  $f(U) > f(U^*)$  then
16:         $U^* \leftarrow U$ 
17:        Update EliteSol with  $U$ 
18:    else
19:       $\text{ConR} = \text{ConR} + 1$ 
return Updated solution  $U^*$ 

```

The specific application of this strategy to FMMD is presented in Algorithm 6. Given a starting point for the search U , we first initialize key parameters (line 1), and set U as the best solution of the current search round (line 2). The algorithm proceeds iteratively until the stopping condition is met, i.e., when the best objective value of the current search round fails to improve for MaxNI consecutive iterations (lines 3–19). In each iteration, we first determine a swap pair (i, j) within the relaxed neighborhood using the selection mechanism described in Algorithm 3 (line 4), and the solution is updated accordingly (line 5). If the new solution improves upon the best solution found in this round, it is recorded as the new best solution and repaired, while the non-improvement counter is reset (lines 6–8). Otherwise, the counter is incremented (line 10). To prevent excessive infeasibility, if the solution remains infeasible for more than MaxCR consecutive iterations, a repair procedure is triggered to restore feasibility (lines 11–12). Next, we check whether the current solution is feasible. If it is, the consecutive infeasibility counter (ConR) is reset, and if the solution also improves upon the best overall solution U^* found so far, U^* and *EliteSol* is updated accordingly (lines 13–17). Otherwise, if the solution is still infeasible, the ConR counter is incremented to track consecutive infeasible iterations (lines 19).

In the repair phase, the swap operator is still employed, but the neighborhood structure is specifically designed to restore feasibility with respect to the group constraints. The repair procedure is summarized in Algorithm 7. A solution is infeasible if there exists a group c such that $|U_c| < l_c$ (shortage) or $|U_c| > p_c$ (excess). Therefore, the algorithm iteratively performs swap moves until all group constraints are

satisfied (line 1). At the beginning of each iteration, the best candidate move and its value are initialized (line 2). Next, all possible swap pairs (i, j) between unselected and selected elements are examined (line 3). A candidate swap is considered only if it reduces the violation of the group bounds (line 4). For each possible swap, the move value $\Delta_{i,j}$ is computed (line 5), and the best move is updated whenever a better value is found (lines 6–8). After evaluating all candidate swaps, the move with the highest $\Delta_{i,j}$ is applied to update the current solution (line 9). This process continues until all groups satisfy the bounds $l_c \leq |U_c| \leq p_c$, at which point the repaired feasible solution is returned.

Algorithm 7 Repair Procedure for FMMD

Require: Current solution U ; lower l_c and upper bounds p_c for each group $c \in C$

Ensure: A feasible solution U satisfying all group bounds

```

1: while  $\exists c \in C$  such that  $|U_c| < l_c$  or  $|U_c| > p_c$  do
2:    $bestMove \leftarrow \emptyset$ ,  $bestValue \leftarrow -\infty$ 
3:   for each  $i \in V \setminus U$  and  $j \in U$  do
4:     if swap  $(i, j)$  reduces violation of group bounds then
5:       compute  $\Delta_{i,j}$ 
6:       if  $\Delta_{i,j} > bestValue$  then
7:          $bestValue \leftarrow \Delta_{i,j}$ 
8:          $bestMove \leftarrow (i, j)$ 
9:   Perform swap defined by  $bestMove$ 
return  $U$ 

```

6.2. Memory-based perturbation

The long-term memory in PTS1 contributes to search diversification; however, it may still be insufficient to escape deep local optima. To further improve the performance of PTS1 in solving the FMMD problem, we incorporate a memory-based perturbation operator that exploits multiple memory structures (Glover et al., 2010; Lü and Hao, 2009). Specifically, the perturbation strategy utilizes the elite solution pool (*EliteSol*), as defined and updated in Section 5.2; the elite frequency (*EliteFreq_i*), which tracks the total number of times element i appears in the R elite solutions; and the flip frequency (*FlipFreq_i*), which records the number of times element i changes its status throughout the search.

Based on these memory components, we define a scoring function, denoted as $Score_i$, to evaluate each element. This function identifies high-scoring elements as critical and prioritizes them for perturbation. Mathematically, $Score_i$ is defined as follows:

$$Score_i = \frac{EliteFreq_i(CR - EliteFreq_i)}{CR^2} + \frac{\tau(MaxFreq - FlipFreq_i)}{MaxFreq} \quad (27)$$

where τ is a scoring parameter, $MaxFreq$ is the largest $FlipFreq$ value and CR is the current number of the elite solutions. The rationale behind this scoring function is that if an element consistently appears or does not appear in the elite solutions, it is assigned a lower probability of being selected for perturbation. Meanwhile, elements that rarely change their status during the search are given a higher probability of being selected for perturbation, which helps the search escape from local optimum regions.

In our Advanced PTS algorithm, perturbation is triggered when the number of consecutive non-improving iterations, tracked by the variable *NonImp* in Algorithm 6, reaches the predefined threshold $MaxNI$. At this point, we randomly select a solution from the elite pool and replace $\eta \times k$ high-scored selected elements with the same number of high-scored unselected elements, ensuring that the number of selected elements in each group remains within the specified lower and upper bounds.

7. Computational experiments

This section presents the computational experiments designed to evaluate the metaheuristic algorithms introduced earlier. We consider four metaheuristics: GRASP (Section 4), two variants of probabilistic tabu search, PTS1 and PTS2 (Section 5), and the enhanced Advanced-PTS (Section 6). The main algorithmic components and their characteristics are summarized in Table 1 for ease of comparison.

All the proposed metaheuristic algorithms for the FMMD problem are implemented in C++ and compiled using GNU GCC 10.2.0 with the -O3 optimization flag. All experiments are conducted on a machine running a Linux 3.10.x86_64 operating system with an Intel(R) Xeon Gold 6226R CPU at 2.90 GHz. In addition to our C++ implementations, we compare against two algorithms provided in Wang et al. (2023b): an exact algorithm (FMMD-E) and a scalable approximation algorithm (FMMD-S). Both of these baseline algorithms are implemented in Python and rely on Gurobi 12.0 for solving the underlying ILP formulations.

For the metaheuristic algorithms, each instance is tested independently 10 times. The time limits are set based on the instance size: for small synthetic instances with fewer than 1000 elements, a cutoff time of 30 s is applied; for instances with 1000 and 10000 elements, the time limit is increased to 60 s. For the real-world instances, a longer time limit of 600 s is used to accommodate their increased complexity. For exact and approximation algorithms, the binary search procedure serves as the termination criterion: the search stops when the upper and lower bounds converge. Each ILP subproblem generated during the binary search is solved with a maximum runtime of 300 s as Wang et al. (2023b).

To evaluate the effectiveness and efficiency of each method, we report the following performance metrics: the total number of instances solved to optimality in each dataset (*#opt*), the number of instances for which the method achieves the best-known solution at least once across multiple runs (*#best*), the average computational time-to-best in the 10 runs averaged over all instances in each dataset (*t*). The best-known solution (*BKS*) for each instance is defined as the best solution obtained among all tested methods, including the exact models as well as the heuristics (PTS1, PTS2, GRASP, and Advanced-PTS). For the mathematical models, the reported computational time refers to the time needed to prove optimality. If optimality is not reached, it reflects the time required to obtain the best-found solution. For the metaheuristics, the computational time corresponds to the average time across 10 independent runs required to reach a local optimum in each run. Additionally, we evaluate solution quality using a percentage deviation metric averaged over all instances in each dataset, which quantifies how much the obtained solution deviates from *BKS*. When f represents the model's solution, the deviation (denoted as %*dev*) is calculated as

$$\%dev = \frac{BKS - f}{BKS} \times 100\%. \quad (28)$$

For the metaheuristics, we compute two deviations using the same formula: %*dev_a* is based on the average solution f_a over ten runs, and %*dev_b* is based on the best solution f_b among the ten runs, i.e.,

$$\%dev_a = \frac{BKS - f_a}{BKS} \times 100\% \quad \text{and} \quad \%dev_b = \frac{BKS - f_b}{BKS} \times 100\%. \quad (29)$$

The remainder of this section is organized as follows. Section 7.1 introduces the benchmark instances and parameter configurations used in this study. In Section 7.2, we compare the results obtained by solving the integer linear programming formulations with those of our proposal Advanced-PTS. Section 7.3 focuses on the comparative analysis of GRASP and PTS variants. Finally, Section 7.4 evaluates the individual contributions of the advanced components integrated into the PTS framework.

Table 1
Descriptions of the proposed algorithms.

Algorithm	Description
GRASP	Multi-start framework combining randomized greedy construction and best-improvement local search. No memory is used.
PTS1	Single-start framework using greedy construction followed by local improvement based on a probabilistic tabu search mechanism.
PTS2	Multi-start variant where construction is guided by a memory mechanism and PTS principles; followed by local improvement.
Advanced-PTS	Based on PTS1 but incorporates strategic oscillation and memory-based perturbation.

7.1. Benchmark instances and parameter configurations

To evaluate the performance of the proposed methods, we conduct experiments on a set of benchmark instances. Specifically, we use synthetic datasets generated based on the rules and Python code provided by Wang et al. (2023a), varying the number of elements (n), the number of groups (C), and the number of selected elements (k). This allows us to assess the adaptability of the algorithms across different instance sizes, group structures, and selection complexities. Additionally, we include a real-world dataset used in Wang et al. (2023a) to further validate the effectiveness of our approach in practical applications.¹ The details of these datasets are as follows:

1. The synthetic dataset consists of 80 instances, where the number of elements ranges from 100 to 10000 (specifically, 100, 500, 1000, and 10000). Following the generation procedure of Wang et al. (2023a), these data are generated by sampling points from ten two-dimensional Gaussian clusters. The centers of these clusters are randomly drawn from the square region $[-10, 10]^2$, and each cluster follows an isotropic Gaussian distribution with an identity covariance matrix. After generating the data points, they are assigned to groups uniformly at random, with the number of groups set to 2, 4, 6, and 8. For each instance, we select $k = 10, 20, 30, 40, 50$ elements.
2. The real-world instance is derived from the *Adult dataset*, which contains 48842 records from the 1994 US Census database. To define the distances between individuals, we directly use the preprocessed data provided by Wang et al. (2023a). According to their preprocessing procedure, six numeric attributes were selected as features and normalized to have a zero mean and a unit standard deviation. The Euclidean distance was then used to compute the distance matrix. The group number is 2 (based on sex), 5 (based on race), and 10 (based on the combination of sex and race). k values are the same as those of the synthetic datasets, resulting in a total of 15 instances.

Given a value of k , the upper and lower bounds for the number of selected elements from each group, l_c and p_c respectively, can be set either based on proportional representation (El Halabi et al., 2020; Celis et al., 2018) or equal representation (Wang et al., 2023a; Kleindessner et al., 2019; Wang et al., 2021). In this study, we adopt the latter approach, which is more commonly used when addressing fairness constraints. To further evaluate the robustness of the proposed methods, we also report additional experiments comparing proportional and equal representation settings and analyze the impact of different fairness constraint tightness levels.

The parameters of the algorithm were optimized using the automatic parameter tuning tool IRACE (López-Ibáñez et al., 2016). The tuning process was conducted with a budget of 5000 experiments. A subset of relatively difficult synthetic evaluation instances was used as the tuning set. Specifically, we considered eight configurations based on the combinations of problem sizes $|V| \in \{1000, 10000\}$ and numbers of groups $C \in \{2, 4, 6, 8\}$. Since each configuration contains several instances with different values of k , we randomly selected exactly one

instance (i.e., one random k per configuration), resulting in a total of 8 training instances. During the tuning process, each run used the same stopping criteria as those adopted in the main computational experiments. The ranges and specific values for each parameter are summarized in Table 2. The table outlines the parameters, their descriptions, the tuning ranges, and the final values selected for the implementation. For parameters with a defined range, the values were randomly selected from within the specified intervals, and for others, specific values were chosen based on the optimization results.

7.2. Analysis of advanced designs

In this section, we present a comprehensive comparison between the mathematical models and a representative metaheuristic across all test instances. Table 3 provides the aggregated performance for different groups of instances, where each group is categorized according to the total number of elements in the graph. This grouping allows us to examine how each method scales with increasing graph size $|V|$.

We can observe from Table 3 that for small-scale instances ($|V| = 100$), both the exact algorithm FMMD-E and our Advanced-PTS algorithm are able to find the optimal or *BKS* for all instances, while the approximation algorithm FMMD-S shows a noticeable performance gap with a deviation of 9.74%. In terms of computational time, all methods are efficient, with Advanced-PTS being the fastest.

For medium-sized instances ($|V| = 500$ and $|V| = 1000$), FMMD-E maintains strong solution quality, reaching optimal solutions for all instances. However, the computational time grows substantially with problem size, averaging 10.67 s for $|V| = 500$ and 111.32 s for $|V| = 1000$. FMMD-S, while highly efficient (0.33 s and 0.56 s on average), consistently produces inferior solutions with deviations exceeding 13%. In comparison, Advanced-PTS achieves the *BKS* in all instances, with small average deviations (less than 0.02%) and moderate computational times (0.47 s and 9.62 s on average).

For large-scale instances ($|V| = 10000$), the limitations of FMMD-E become evident: it fails to reach optimality within the time limit for all instances and shows a noticeable degradation in solution quality (10.22% deviation), along with extremely high computational time. FMMD-S remains highly efficient (6.52 s on average) but produces inferior solutions (17.11% deviation). In comparison, Advanced-PTS obtains the best-known solutions in 19 out of 20 instances, with a small average deviation (1.13%) and moderate computational time (29.56 s on average).

For the real-world instances ($|V| = 48842$), FMMD-E is unable to produce results due to computational limitations. FMMD-S remains scalable but yields relatively poor solution quality (10.78% deviation). Advanced-PTS continues to demonstrate strong performance, obtaining the *BKS* in 13 out of 15 instances with an average deviation of only 1.27%, although its running time is higher.

Overall, while FMMD-E performs well on small and medium instances, its scalability is limited due to the increasing difficulty of solving the underlying ILP subproblems. FMMD-S offers good efficiency but at the expense of solution quality. In contrast to the exact and approximation algorithms, our Advanced-PTS algorithm consistently achieves high-quality solutions across all instance sizes, with the best overall balance between solution quality and computational efficiency.

¹ The C++ source code of our proposed algorithms and the complete set of benchmark instances are available at https://github.com/hzzheng65-hzz/FMMD_PTS.

Table 2
Parameter settings.

Param.	Description	Used in	Range	Value
α	Controls greediness and randomness.	GRASP	[0.1, 0.9]	0.9
σ	Base value to compute tabu tenure γ .	PTS1, Advanced-PTS	{10, 30, 50, 100}	30
γ	Tabu tenure.	PTS1, Advanced-PTS	$[0.3\sigma, 0.8\sigma]$	–
$prob_1$	Probability of selecting the best move.	All PTS variants	[0.1, 0.8]	0.8
$prob_2$	Probability of selecting top-ranked moves.	All PTS variants	[0.2, 0.9]	0.9
β	Fraction of top-ranked moves considered.	All PTS variants	[0.1, 0.9]	0.2
R	Size of the elite solution pool.	PTS2, Advanced-PTS	{10, 20, 30, 40, 50}	20
ζ	Max similarity allowed in elite pool.	PTS2, Advanced-PTS	[0.6, 0.9]	0.6
ϵ	Relaxation for group constraints.	Advanced-PTS	$[\frac{k}{10c}, \frac{k}{c}]$	$\frac{k}{3c}$
$MaxNI$	Max non-improving iterations.	Advanced-PTS	{500, 1000, 1500, 2000}	500
$MaxCR$	Max consecutive infeasible iterations.	Advanced-PTS	[0.1k, 0.9k]	0.1k
τ	Perturbation scoring parameter.	Advanced-PTS	[0.1, 0.9]	0.2
η	Number of elements perturbed.	Advanced-PTS	[0.1k, 0.9k]	0.1k

Table 3
Comparative results for FMMD-E, FMMD-S, and Advanced-PTS.

V	k	#ins	FMMD-E				FMMD-S				Advanced-PTS			
			#opt	%dev	t		#best	%dev	t		#best	%dev _b	%dev _a	t
100	10–50	20	20	0.00	0.29		0	9.74	0.12		20	0.00	0.00	0.02
500	10–50	20	20	0.00	10.67		0	13.67	0.33		20	0.00	0.00	0.47
1000	10–50	20	20	0.00	111.32		0	15.53	0.56		20	0.00	0.02	9.62
10 000	10–50	20	0	10.22	11 599.08		0	17.11	6.52		19	0.00	1.13	29.56
48 842	10–50	15	–	–	–		0	10.78	78.85		13	0.04	1.27	333.64
Sum		95	60	–	–		0	–	–		92	–	–	–
Avg			–	2.55	2 930.34		–	13.37	17.28		–	0.01	0.48	74.66

0.00 means less than 0.001.

Table 4
Comparative results on all tested benchmark instances for PTS1, PTS2, and GRASP.

V	k	#ins	PTS1				PTS2				GRASP			
			#best	%dev _b	%dev _a	t	#best	%dev _b	%dev _a	t	#best	%dev _b	%dev _a	t
100	10–50	20	20	0.00	0.01	0.41	20	0.00	0.21	1.08	19	0.13	0.22	1.64
500	10–50	20	17	0.10	0.19	3.94	15	0.25	0.67	8.59	11	1.04	1.84	11.70
1000	10–50	20	12	0.46	0.79	12.71	7	1.52	2.00	22.28	6	2.58	3.25	22.95
10 000	10–50	20	6	0.87	1.77	27.84	2	3.90	5.15	31.03	1	4.78	6.22	28.70
48 842	10–50	15	7	0.42	1.56	314.13	2	3.04	4.24	293.08	3	3.42	5.29	289.78
Summary		95	62	0.37	0.86	71.81	46	1.74	2.46	71.21	40	2.39	3.36	70.95
p-value				1.07e–10				2.54e–06				–		

0.00 means less than 0.001.

7.3. Detailed comparison between GRASP, PTS1 and PTS2

This section presents a detailed comparison of the three metaheuristics GRASP, and our probabilistic tabu search methods PTS1 and PTS2. We begin by summarizing their overall performance across all benchmark instances using the quantitative indicators previously introduced. To complement these, we additionally report the p -value from the Wilcoxon signed-rank test in Table 4, assessing the statistical significance of the performance differences when each method is compared against GRASP as a baseline. Beyond aggregate comparisons, we further explore the dynamic behavior of the algorithms during the search process in Fig. 2.

As shown in Table 4, across all instance sizes, PTS1 consistently outperforms both GRASP and PTS2 in terms of solution quality. Specifically, PTS1 achieved BKS in 62 out of 95 instances, while PTS2 and GRASP achieved 46 and 40 best results, respectively. In terms of relative deviation from the best solution found ($\%dev_b$), PTS1 achieves the lowest average value of 0.37%, followed by PTS2 at 1.74%, and GRASP at 2.39%. Similarly, with respect to the average deviation from the BKS ($\%dev_a$), PTS1 again reports the lowest deviation of 0.86%, whereas PTS2 and GRASP present 2.46% and 3.36%, respectively. This demonstrates the robustness and effectiveness of PTS1 algorithm. Although PTS1 generally takes slightly more computation time than GRASP on smaller instances, it remains competitive overall. For example, on the largest instance set ($|V| = 48\,842$), PTS1 and PTS2 required

314 s and 293 s on average, respectively, while GRASP ran slightly faster at 290 s. However, this marginal time difference is compensated by the significantly better solution quality of PTS1. Moreover, the statistical significance of the results is confirmed by p -values from the Wilcoxon signed-rank test. The comparison between PTS1 and GRASP yields a p -value of $1.07e-10$, and between PTS2 and GRASP a p -value of $2.54e-06$, indicating that the performance differences are statistically significant at the 0.05 level.

To further illustrate the behavior of the metaheuristics, we present the convergence results on a set of more challenging instances, where $n = 1000$ and $k = 30$. Fig. 2 contains four subgraphs, each representing the convergence behavior with 2, 4, 6, and 8 groups, respectively. The vertical axis represents the average objective value over 10 runs, while the horizontal axis indicates the running time. Each point in the graph corresponds to an update at that time.

The experimental results reveal that PTS1 converges rapidly and consistently yields high-quality solutions. In contrast, the multi-start methods, PTS2 and GRASP, exhibit slower convergence but continue to improve their solutions during the later stages of the search. Among them, PTS2 outperforms GRASP in both convergence speed and solution quality. Within the given time frame, GRASP fails to reach the solution quality achieved by the memory-based methods, PTS1 and PTS2. Notably, PTS1 delivers superior results compared to PTS2. Moreover, as the number of groups increases, PTS1 maintains its performance advantage and robustness, whereas PTS2 and GRASP experience a significant decline in solution quality.

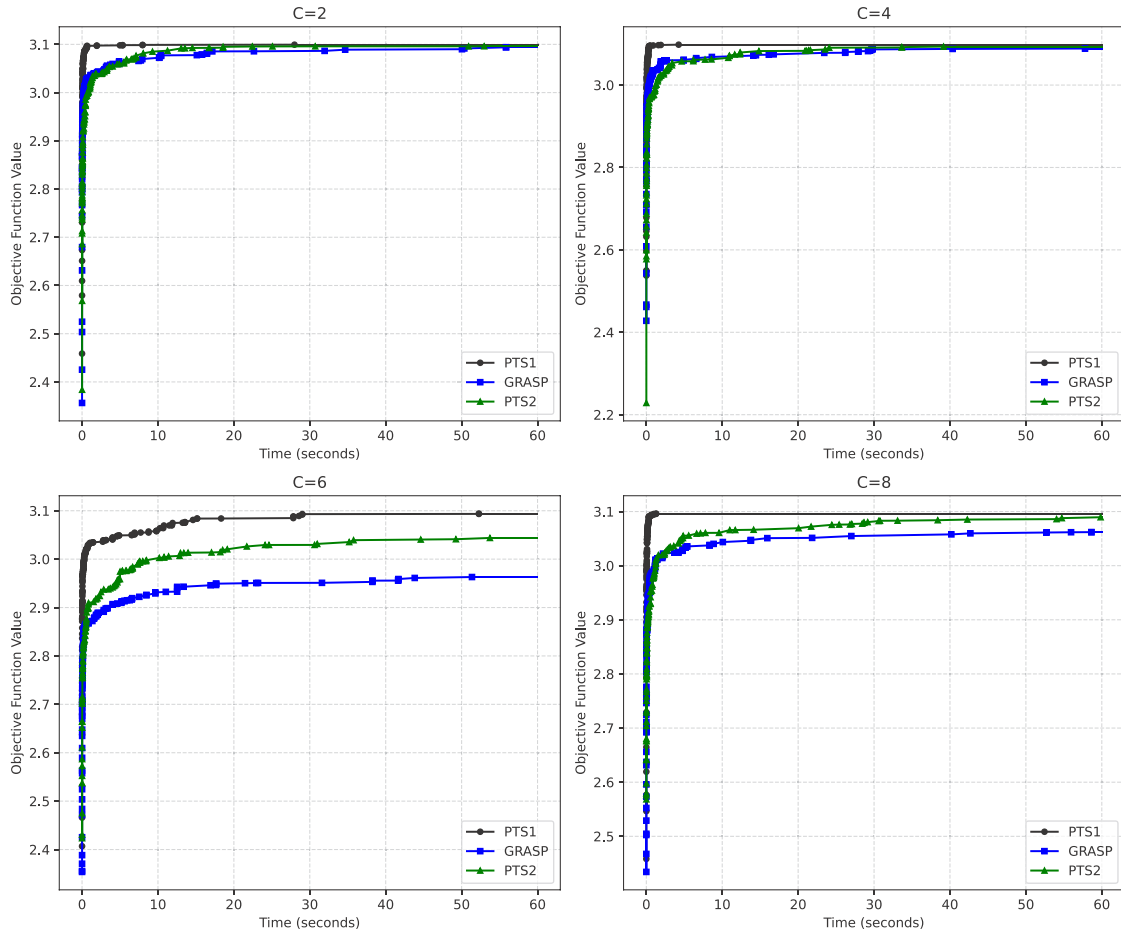


Fig. 2. Convergence analysis of PTS1, PTS2 and GRASP over time in selecting 30 elements from 1000.

These findings highlight the effectiveness of incorporating memory-based mechanisms, particularly during the improvement phase, as implemented in our PTS1 method. Similar trends have been observed in other combinatorial problems such as the QAP (Fleurent and Glover, 1999). This method proves especially effective for solving the FMMD problem, delivering stable performance even on challenging instances.

7.4. Impact of components in the advanced-PTS

To investigate the contribution of each component in the proposed Advanced-PTS framework, we constructed two improved variants based on the baseline PTS1. The first variant, Advanced-PTS1, integrates the memory-based perturbation mechanism into PTS1. The second variant, Advanced-PTS2, incorporates the relax-and-repair strategic oscillation into the same baseline.

Table 5 presents a comparative summary of the performance of these two variants and our Advanced-PTS against PTS1. For each method, we report the average CPU time (t) required to reach a local optimum over 10 independent runs for each instance. We also include the percentage deviation from PTS1's results, where $\%gap_b$, $\%gap_a$, and $\%gap_w$ respectively indicate the deviation based on the best, average, and worst solutions obtained in 10 runs. In addition, we include the number of successful runs ($\#succ$), which reflects how often the method attains the best solution among the 10 trials for each instance. Finally, we provide the p -value to indicate whether the observed performance improvement over PTS1 is statistically significant, based on the best results of each algorithm.

The results indicate that both memory-based perturbation and strategic oscillation contribute positively to improving the effectiveness of

the baseline PTS1, with the combination of both yielding the most consistent improvements. In terms of solution quality, Advanced-PTS achieves the highest average performance across all $\%gap$ metrics. Specifically, it attains an average $\%gap_b$, $\%gap_a$, and $\%gap_w$ of 0.37%, 0.39%, and 0.42%, respectively, which are all higher than those of the other variants. Notably, while Advanced-PTS1 and Advanced-PTS2 also demonstrate improvements over PTS1, the benefits are more pronounced when both mechanisms are used together. Regarding computational time, Advanced-PTS2 records the lowest average CPU time (66.70 s), suggesting that the relax-and-repair strategy helps the algorithm converge faster to high-quality local optima. Although Advanced-PTS shows slightly higher average runtime than PTS1, the trade-off is justified by the significant improvements in solution quality and success rate. The $\#succ$ values further support these observations. Advanced-PTS achieves the highest overall success rate (6.60), indicating that it consistently produces the best solutions more frequently across different runs. This is followed by Advanced-PTS2 (6.21), Advanced-PTS1 (6.14), and PTS1 (5.78), confirming the incremental effectiveness of each enhancement. Statistical significance is also validated by the p -values, all of which are below 0.05 for the variants, indicating that the observed improvements are not due to random variations. Overall, these results confirm that both memory-based perturbation and strategic oscillation are effective components that enhance the performance of PTS1, with their combination in Advanced-PTS providing the most substantial benefits in terms of both solution quality and robustness.

7.5. Sensitivity analysis

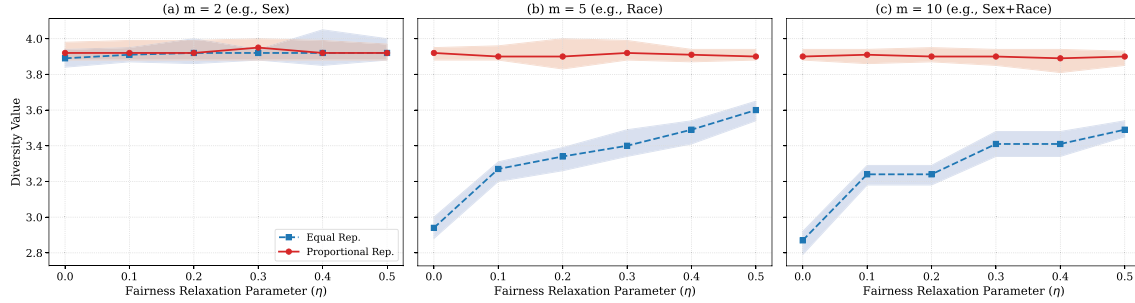
To further evaluate the impact of different fairness definitions and constraint tightness, we conduct a sensitivity analysis on the Adult

Table 5

Comparative results on all benchmark instances for PTS variants.

V	k	PTS1		Advanced-PTS1			Advanced-PTS2		Advanced-PTS2			Advanced-PTS		Advanced-PTS			Advanced-PTS		Advanced-PTS		
		t	#succ	%gap _b	%gap _a	%gap _w	t	#succ	%gap _b	%gap _a	%gap _w	t	#succ	%gap _b	%gap _a	%gap _w	t	#succ	%gap _b	%gap _a	%gap _w
100	10-50	0.41	9.95	0.00	0.01	0.13	0.05	10.00	0.00	0.01	0.13	0.18	10.00	0.00	0.01	0.13	0.02	10.00	0.00	0.01	0.13
500	10-50	3.94	8.65	0.10	0.16	0.31	2.75	9.35	0.10	0.19	0.53	0.70	9.95	0.10	0.19	0.53	0.47	10.00	0.10	0.19	0.53
1000	10-50	12.71	5.30	0.43	0.69	0.91	12.52	6.45	0.42	0.73	1.05	7.50	7.10	0.47	0.79	1.15	9.62	8.75	0.47	0.79	1.15
10 000	10-50	27.84	2.45	0.30	0.20	-0.12	34.15	2.50	0.79	0.53	0.06	32.13	1.85	0.88	0.67	0.23	29.56	2.10	0.88	0.67	0.23
48 842	10-50	314.13	2.53	0.15	-0.05	-0.17	331.66	2.40	0.11	0.10	-0.07	292.98	2.13	0.38	0.30	0.06	333.64	2.13	0.38	0.30	0.06
Avg		71.81	5.78	0.20	0.20	0.21	76.23	6.14	0.28	0.31	0.34	66.70	6.21	0.37	0.39	0.42	74.66	6.60	0.37	0.39	0.42
p-value		-	-	-	-	5.60e-04	-	-	-	-	5.40e-05	-	-	-	-	1.30e-08	-	-	-	-	-

0.00 means less than 0.001.

**Fig. 3.** Sensitivity of the objective value to different fairness constraint tightness levels under proportional and equal representation on the Adult dataset ($k = 50$).

dataset. Consistent with the main experiments, each parameter configuration is executed 10 times independently under the same stopping criteria. In this experiment, the subset size is fixed to $k = 50$, and we consider two common fairness definitions: proportional representation and equal representation.

For each group c , we first compute its baseline bounds, denoted as the base lower bound l_c^{base} and upper bound p_c^{base} . Under equal representation, the base bounds are defined as $l_c^{\text{base}} = \lfloor k \cdot \frac{1}{C} \rfloor$ and $p_c^{\text{base}} = \lceil k \cdot \frac{1}{C} \rceil$. Under proportional representation, the base bounds are determined according to the group size ratio, i.e., $l_c^{\text{base}} = \lfloor k \cdot \frac{|V_c|}{|V|} \rfloor$ and $p_c^{\text{base}} = \lceil k \cdot \frac{|V_c|}{|V|} \rceil$. The fairness tightness is controlled by a parameter $\eta \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$, where $\eta = 0$ corresponds to the strictest fairness constraint. Specifically, given base lower and upper bounds for each group, the actual bounds are set as $l_c = \lceil (1 - \eta) l_c^{\text{base}} \rceil$, $p_c = \lfloor (1 + \eta) p_c^{\text{base}} \rfloor$, allowing the algorithm to explore progressively larger feasible regions as η increases.

The results are illustrated in Fig. 3, showing the sensitivity of the objective value to the fairness relaxation parameter η under both equal and proportional representation settings. The lines represent the average objective value over 10 independent runs, while the shaded regions indicate the range between the best and worst solutions obtained. Under the proportional representation setting, the objective values remain relatively stable as η varies. This is because proportional representation allocates the number of selected elements according to the group sizes, which introduces relatively mild restrictions on the feasible region. In contrast, the equal representation setting is more sensitive to the relaxation parameter. When $\eta = 0$, the strict fairness constraint requires each group to contribute approximately the same number of selected elements. This restriction can significantly reduce the achievable diversity when the group sizes are highly imbalanced. In the Adult dataset, some groups contain only a small number of elements, which makes it difficult to select sufficiently diverse elements from these groups. As the relaxation parameter η increases, the feasible region gradually expands, allowing the algorithm to achieve higher objective values. Finally, the relatively narrow shaded regions indicate that the algorithm exhibits stable performance across the 10 runs for most parameter settings.

8. Conclusion

This study contributes to the growing field of diversity optimization under fairness constraints by addressing the FMMD problem. We presented exact optimization models for the FMMD that can solve small- and medium-sized instances to optimality and provide a reliable benchmark for evaluating heuristic performance. Recognizing the scalability challenges of exact methods, we designed several heuristic algorithms, including GRASP, and two probabilistic tabu search algorithms PTS1 and PTS2. Our results demonstrate that memory-based mechanisms, whether used in the construction or local search phase, consistently improve performance, with particularly significant gains observed when applied during the local search phase. To further enhance solution quality, we proposed an Advanced-PTS algorithm that incorporates memory-based perturbation and strategic oscillation. Computational experiments on a wide range of instances show that our Advanced-PTS algorithm delivers consistently superior solutions compared to the basic metaheuristics and achieves significantly superior performance relative to the exact approaches for larger problems. Component analysis further confirms the effectiveness of the enhancements introduced.

Future research on the FMMD problem could focus on further enhancing the quality and scalability of the solution. In particular, integrating population-based strategies or path-relinking mechanisms into the proposed PTS framework offers a promising direction to balance intensification and diversification more effectively. Another potential direction is to investigate hybrid approaches that combine metaheuristic methods with learning-based techniques, including those proposed in Glover and Laguna (1998) and Glover (2000), to adaptively guide the search process.

CRedit authorship contribution statement

He Zheng: Writing – original draft, Validation, Software, Methodology, Investigation, Data curation, Conceptualization. **Anna Martínez-Gavara:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Conceptualization. **Rafael Martí:** Writing – review & editing, Supervision, Methodology, Investigation, Conceptualization. **Fred Glover:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Acknowledgments

This research was partially supported by the Ministerio de Ciencia e Innovación of Spain (Grant No. PID2024-160226OB-C21), funded by MCIN/AEI/10.13039/501100011033 and “ERDF A way of making Europe”, and by the Generalitat Valenciana, Spain (Grant No. CIAICO/2026/100).

Data availability

Data will be made available on request.

References

- Abbassi, Z., Mirrokni, V.S., Thakur, M., 2013. Diversity maximization under matroid constraints. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '13, Association for Computing Machinery, New York, NY, USA, pp. 32–40.
- Absi, N., Cattaruzza, D., Feillet, D., Housseman, S., 2017. A relax-and-repair heuristic for the Swap-Body Vehicle Routing Problem. *Ann. Oper. Res.* 253, 957–978.
- Addanki, R., McGregor, A., Meliou, A., Mousmoulidou, Z., 2022. Improved approximation and scalability for fair max-min diversification. In: Olteanu, D., Vortmeier, N. (Eds.), 25th International Conference on Database Theory. ICDT 2022, In: Leibniz International Proceedings in Informatics (LIPIcs), vol. 220, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 7:1–7:21.
- Argyris, N., Karsu, Ö., Yavuz, M., 2022. Fair resource allocation: Using welfare-based dominance constraints. *European J. Oper. Res.* 297 (2), 560–578.
- Benlic, U., Hao, J.-K., 2013. Breakout local search for the vertex separator problem. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence. AAAI Press, pp. 461–467.
- Bera, S.K., Das, S., Galhotra, S., Kale, S.S., 2022. Fair k-center clustering in MapReduce and streaming settings. In: Proceedings of the ACM Web Conference 2022. WWW '22, Association for Computing Machinery, New York, NY, USA, pp. 1414–1422.
- Blanco, V., Gázquez, R., 2023. Fairness in maximal covering location problems. *Comput. Oper. Res.* 157, 106287.
- Brimberg, J., Mladenović, N., Todosijević, R., Urošević, D., 2019. Solving the capacitated clustering problem with variable neighborhood search. *Ann. Oper. Res.* 272, 289–321.
- Brimberg, J., Mladenović, N., Urošević, D., 2015. Solving the maximally diverse grouping problem by skewed general variable neighborhood search. *Inform. Sci.* 295, 650–675.
- Celis, E., Keswani, V., Straszak, D., Deshpande, A., Kathuria, T., Vishnoi, N., 2018. Fair and diverse DPP-based data summarization. In: Dy, J., Krause, A. (Eds.), Proceedings of the 35th International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 80, PMLR, Stockholm, Sweden, pp. 716–725.
- Chiang, W.-C., Kouvelis, P., 1996. An improved tabu search heuristic for solving facility layout design problems. *Int. J. Prod. Res.* 34 (9), 2565–2585.
- Chierichetti, F., Kumar, R., Lattanzi, S., Vassilvitskii, S., 2017. Fair clustering through fairlets. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. Vol. 30, Curran Associates, Inc., Long Beach, CA, USA, pp. 5029–5037.
- El Halabi, M., Mitrović, S., Norouzi-Fard, A., Tardos, J., Tarnawski, J.M., 2020. Fairness in streaming submodular maximization: Algorithms and hardness. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (Eds.), Advances in Neural Information Processing Systems. Vol. 33, Curran Associates, Inc., Vancouver, Canada, pp. 13609–13622.
- Feo, T.A., Resende, M.G., Smith, S.H., 1994. A greedy randomized adaptive search procedure for maximum independent set. *Oper. Res.* 42 (5), 860–878.
- Festa, P., Resende, M.G.C., 2018. GRASP. In: Handbook of Heuristics. Springer International Publishing, Cham, pp. 465–488.
- Filippi, C., Guastaroba, G., Huerta-Muñoz, D., Speranza, M., 2021. A kernel search heuristic for a fair facility location problem. *Comput. Oper. Res.* 132, 105292.
- Fleurent, C., Glover, F., 1999. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS J. Comput.* 11 (2), 198–204.
- Gallego, M., Laguna, M., Martí, R., Duarte, A., 2013. Tabu search with strategic oscillation for the maximally diverse grouping problem. *J. Oper. Res. Soc.* 64 (5), 724–734.
- Glover, F., 1989. Tabu search—Part I. *ORSA J. Comput.* 1 (3), 190–206.
- Glover, F., 1994. Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Appl. Math.* 49 (1), 231–255.
- Glover, F., 2000. Multi-start and strategic oscillation methods — Principles to exploit adaptive memory. In: Laguna, M., González Velarde, J.L. (Eds.), Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research. Kluwer Academic Publishers, Springer, Boston, MA, pp. 1–24.
- Glover, F., 2016. Pseudo-centroid clustering. *Soft Comput.* (21(22)), 6571–6592.
- Glover, F., Laguna, M., 1993. Tabu search. In: Reeves, C. (Ed.), Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Publishing, New York, pp. 71–140.
- Glover, F., Laguna, M., 1998. Tabu search. In: Du, D.-Z., Pardalos, P.M. (Eds.), Handbook of Combinatorial Optimization: Volume 1–3. Springer US, Boston, MA, pp. 2093–2229.
- Glover, F., Lü, Z., Hao, J.-K., 2010. Diversification-driven tabu search for unconstrained binary quadratic problems. *4OR* 8, 239–253.
- Guan, J., Lin, G., Feng, H.-B., 2018. A learning-based probabilistic tabu search for the uncapacitated single allocation hub location problem. *Comput. Oper. Res.* 98, 1–12.
- Guemri, O., Nduwayo, P., Todosijević, R., Hanafi, S., Glover, F., 2019. Probabilistic tabu search for the cross-docking assignment problem. *European J. Oper. Res.* 277 (3), 875–885.
- Kleindessner, M., Awasthi, P., Morgenstern, J., 2019. Fair k-center clustering for data summarization. In: Chaudhuri, K., Salakhutdinov, R. (Eds.), Proceedings of the 36th International Conference on Machine Learning. In: Proceedings of Machine Learning Research, vol. 97, PMLR, Long Beach, California, pp. 3448–3457.
- Kulturel-Konak, S., 2012. A linear programming embedded probabilistic tabu search for the unequal-area facility layout problem with flexible bays. *European J. Oper. Res.* 223 (3), 614–625.
- Kurkure, Y., Shamo, M., Wiseman, J., Galhotra, S., Sintos, S., 2024. Faster algorithms for fair max-min diversification in Rd. *Proc. ACM Manag. Data* 2 (3).
- Lai, X., Hao, J.-K., 2016. Iterated maxima search for the maximally diverse grouping problem. *European J. Oper. Res.* 254 (3), 780–800.
- Lai, X., Hao, J.-K., Fu, Z.-H., Yue, D., 2021. Neighborhood decomposition based variable neighborhood search and tabu search for maximally diverse grouping. *European J. Oper. Res.* 289 (3), 1067–1086.
- Li, G., Liu, H., 2006. Resource allocation for OFDMA relay networks with fairness constraints. *IEEE J. Sel. Areas Commun.* 24 (11), 2061–2069.
- López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. *Oper. Res. Perspect.* 3, 43–58.
- Lü, Z., Hao, J.-K., 2009. A critical element-guided perturbation strategy for iterated local search. In: Cotta, C., Cowling, P. (Eds.), Evolutionary Computation in Combinatorial Optimization. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–12.
- Lu, Z., Martínez-Gavara, A., Hao, J.-K., Lai, X., 2023. Solution-based tabu search for the capacitated dispersion problem. *Expert Syst. Appl.* 223, 119856.
- Machado, L., Stefanidis, K., 2019. Fair team recommendations for multidisciplinary projects. In: IEEE/WIC/ACM International Conference on Web Intelligence. WI '19, Association for Computing Machinery, New York, NY, USA, pp. 293–297.
- Marsh, M.T., Schilling, D.A., 1994. Equity measurement in facility location analysis: A review and framework. *European J. Oper. Res.* 74 (1), 1–17.
- Martí, R., Martínez-Gavara, A., 2023. Discrete Diversity and Dispersion Maximization. In: Springer Optimization and its Applications, vol. 204, Springer Switzerland AG.
- Martí, R., Martínez-Gavara, A., Pérez-Peló, S., Sánchez-Oro, J., 2022. A review on discrete diversity and dispersion maximization from an OR perspective. *European J. Oper. Res.* 299 (3), 795–813.
- Martínez-Gavara, A., Corberán, T., Martí, R., 2021. GRASP and tabu search for the generalized dispersion problem. *Expert Syst. Appl.* 173, 114703.
- Martínez-Gavara, A., Landa-Silva, D., Campos, V., Martí, R., 2017. Randomized heuristics for the Capacitated Clustering Problem. *Inform. Sci.* 417, 154–168.
- Mladenović, N., Todosijević, R., Urošević, D., Ratli, M., 2022. Solving the Capacitated Dispersion Problem with variable neighborhood search approaches: From basic to skewed VNS. *Comput. Oper. Res.* 139, 105622.
- Mori, H., Muroi, T., 2011. Application of probabilistic Tabu Search to distribution system service restoration. In: 2011 IEEE International Symposium of Circuits and Systems. ISCAS, pp. 1037–1040.
- Mousmoulidou, Z., McGregor, A., Meliou, A., 2021. Diverse data selection under fairness constraints. In: Yi, K., Wei, Z. (Eds.), 24th International Conference on Database Theory. ICDT 2021, In: Leibniz International Proceedings in Informatics (LIPIcs), vol. 186, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 13:1–13:25.
- Patel, D., Khan, A., Louis, A., 2021. Group fairness for knapsack problems. URL: <https://arxiv.org/abs/2006.07832>, arXiv:2006.07832.
- Prais, M., Ribeiro, C.C., 2000. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS J. Comput.* 12 (3), 164–176.
- Radanović, L., Mijović, A., Urošević, D., Davidović, T., Jovanovic, R., 2025. General variable neighborhood search for maximum diversity problem with capacity and budget constraints. *Expert Syst. Appl.* 267, 126188.
- Resende, M., Martí, R., Gallego, M., Duarte, A., 2010. GRASP and path relinking for the max-min diversity problem. *Comput. Oper. Res.* 37 (3), 498–508, Hybrid Metaheuristics.
- Ribeiro, C.C., Uchoa, E., Werneck, R.F., 2002. A Hybrid GRASP with perturbations for the Steiner problem in graphs. *INFORMS J. Comput.* 14 (3), 228–246.
- Sánchez-Oro, J., Martínez-Gavara, A., López-Sánchez, A.D., Martí, R., Duarte, A., 2025. GRASP with evolutionary path relinking for the conditional p-dispersion problem. *Appl. Soft Comput.* 168, 112494.

- Sayyady, F., Fathi, Y., 2016. An integer programming approach for solving the p-dispersion problem. *European J. Oper. Res.* 253 (1), 216–225.
- Wang, Y., Fabbri, F., Mathioudakis, M., 2021. Fair and representative subset selection from data streams. In: *Proceedings of the Web Conference 2021. WWW '21*, Association for Computing Machinery, New York, NY, USA, pp. 1340–1350.
- Wang, Y., Fabbri, F., Mathioudakis, M., Li, J., 2023a. Fair max-min diversity maximization in streaming and sliding-window models. *Entropy* 25 (7), 1066.
- Wang, Y., Mathioudakis, M., Li, J., Fabbri, F., 2023b. Max-min diversification with fairness constraints: Exact and approximation algorithms. In: *Proceedings of the 2023 SIAM International Conference on Data Mining. SDM, SIAM*, pp. 91–99.
- Wang, S., Zhao, Q., 2022. Probabilistic tabu search algorithm for container liner shipping problem with speed optimisation. *Int. J. Prod. Res.* 60 (12), 3651–3668.
- Xu, J., Chiu, S.Y., Glover, F., 1996. Probabilistic tabu search for telecommunications network design. *Comb. Optim.: Theory Pract.* 1 (1), 69–94.
- Yang, X., Cai, Z., Jin, T., Tang, Z., Gao, S., 2022. A three-phase search approach with dynamic population size for solving the maximally diverse grouping problem. *European J. Oper. Res.* 302 (3), 925–953.
- Zheng, H., Martínez-Gavara, A., Martí, R., 2025. Heuristic algorithms for the fair max-min diversity problem. In: *XVI Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*.
- Zhou, Q., Benlic, U., Wu, Q., Hao, J.-K., 2019. Heuristic search to the capacitated clustering problem. *European J. Oper. Res.* 273 (2), 464–487.