European Journal of Operational Research 2026, https://doi.org/10.1016/j.ejor.2026.09.026

Learning-driven Multi-Start Local Search for the Multiple Row Equal Facility Layout Problem

Wen Sun [a,b], Qi Yue [a], Jin-Kao Hao [c,*], Qinghua Wu [d], Fengshun Wang [a]

[a] *School of Cyber Science and Engineering, Southeast University, 2 Road Southeast University, 211189 Nanjing, China*

[b] *Frontiers Science Center for Mobile Information Communication and Security, Southeast University, 2 Road Southeast University, 211189 Nanjing, China*

[c] *LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

[d] *School of Management, Huazhong University of Science and Technology, 1037 Road Luoyu, 430074 Wuhan, China*

Abstract

Given a set of facilities, integer numbers k and r, a facility flow matrix, and a group distance matrix, the multiple row equal facility layout problem (MREFLP) aims to determine a facility-to-group assignment with the minimum aggregate of the products of the flow cost and the facility distances, such that the number of the facilities assigned to each group does not exceed r. The MREFLP is a general model for various applications including operating theater layout in hospitals. However, due to its NP-hardness, solving this problem is computationally difficult. This paper proposes an effective learning-driven multi-start local search algorithm for this problem, which features a learning-driven solution construction procedure and a dedicated two-phase local search. Experiments on 572 benchmark instances demonstrate that the algorithm surpasses the state-of-the-art algorithms. Specifically, it achieves 165 new upper bounds and 407 equal results (including 289 known optima). An ablation study is performed to show the influence of each algorithm component.

Keywords: Multi-start local search; Learning-driven construction; Fast neighborhood evaluation; Multiple row equal facility layout; Heuristics.

1 Introduction

Given a set of facilities $\pi = \{1, \ldots, n\}$, an integer number k (number of groups), an integer number r (maximum number of facilities per group), a facility flow w_{uv} from facility u to v for all $u, v \in \pi$, and a group distance D_{ij} between group i and j for all $i, j \in \{1, \ldots, k\}$, the multiple row equal facility layout problem aims to identify a minimum total cost assignment of n facilities to k groups, with the constraint that the number of facilities in each group does not exceed r (Anjos et al., 2018). The total cost of a facility to group assignment is the aggregate of the products of the facility flows and the facility distances between all facility pairs, where the center-to-center facility distance d_{uv} between facilities u and v for all $u, v \in \pi$ is defined as the distance between their groups i and j. The group distance D_{ij} between the groups i and j is defined as the absolute value of the difference between the two groups. Let $S = \{V_1, V_2, \ldots, V_k\}$ represent a candidate solution, where V_i $(1 \leq i \leq k)$ is the set of facilities assigned to group i. Then, the MREFLP can be mathematically defined as follows (Uribe et al., 2021).

$$(MREFLP) \quad \text{Minimize} \quad f(S) = \sum_{u=1}^{n-1} \sum_{v=u+1}^{n} w_{uv} d_{uv} \tag{1}$$

subject to

$$0 \leq |V_i| \leq r, \ 1 \leq i \leq k \tag{2}$$

$$|V_i \cap V_j| = 0, \ 1 \leq i \neq j \leq k \tag{3}$$

$$\sum_{i=1}^{k} |V_i| = n \tag{4}$$

* Corresponding author.
Email addresses: 101012533@seu.edu.cn (Wen Sun), 220245689@seu.edu.cn (Qi Yue), jin-kao.hao@univ-angers.fr (Jin-Kao Hao), qinghuawu1005@gmail.com (Qinghua Wu), 220224925@seu.edu.cn (Fengshun Wang).

$$d_{uv} = D_{ij},\ u \in V_i, v \in V_j \tag{5}$$

$$D_{ij} = |i - j|,\ 1 \leq i, j \leq k \tag{6}$$

where w, d, D, and n are the facility flow matrix, the facility distance matrix, the group distance matrix, and the number of facilities, respectively.

Equation (1) (objective function) aims to minimize the total cost. Constraint (2) ensures that no group exceeds r facilities. Constraints (3) and (4) guarantee unique group assignments for each facility. Constraint (5) states that d_{uv} is the distance between two facilities u and v. Constraint (6) indicates that the distance D_{ij} between groups i and j is the absolute value of the difference between the two groups.

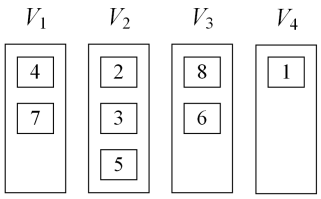
Figure 1 illustrates an instance of MREFLP with $\pi = \{1, 2, \ldots, 8\}$, $r = 3$, and $k = 4$, along with the facility flow matrix (a) and the group distance matrix (b). Figure 1(c) shows a candidate solution $S = \{\{4, 7\}, \{2, 3, 5\}, \{8, 6\}, \{1\}\}$ with an objective value of 328. Figure 1(d) shows an optimal solution $S = \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}\}$ with an objective value of 40.

$$w = \begin{bmatrix} 0 & 50 & 1 & 1 & 1 & 1 & 1 & 1 \\ 50 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 50 & 1 & 1 & 1 & 1 \\ 1 & 1 & 50 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 50 & 1 & 1 \\ 1 & 1 & 1 & 1 & 50 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 50 \\ 1 & 1 & 1 & 1 & 1 & 1 & 50 & 0 \end{bmatrix}$$

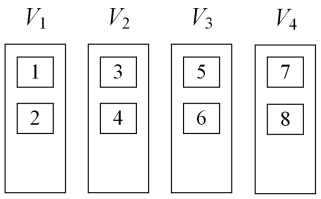
(a) A facility flow matrix w

$$D = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix}$$

(b) A group distance matrix D



(c) A candidate solution $S = \{\{4, 7\}, \{2, 3, 5\}, \{8, 6\}, \{1\}\}$



(d) An optimal solution $S = \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}\}$

Figure 1. An example of MREFLP with its facility flow matrix w, group distance matrix D, a candidate solution and an optimal solution

The MREFLP is inspired by the operating theater layout problem in hospitals (Lin et al., 2015). This involves assigning a set of theaters to multiple floors and rooms in a hospital. There are different interaction frequencies between each

pair of theaters for patient transfers, shared staff, or equipment. These theaters are located on different floors and rooms, with transportation between floors incurring movement costs. The goal is the minimization of transportation costs between theaters. The transportation cost is calculated as the aggregate of the product of interaction frequency and movement cost for each pair of theaters. By modeling this scenario with π as the set of operating theaters, w as the interaction frequency, D as the movement cost, k as the number of floors, and r as the limitation of the number of rooms on each floor, the MREFLP addresses the optimization of theater assignments to reduce overall costs.

As a general model, the MREFLP can also be used to formulate other practical problems related to machine layout in flexible manufacturing systems (Heragu and Kusiak, 1988), data memory layout (Wess and Zeitlhofer, 2004), and keyboard arrangement (Eggers et al., 2003). The MREFLP is an NP-hard problem (Anjos et al., 2018) and therefore solving this problem is computationally challenging.

Recent studies have shown that few practical algorithms have been proposed for the MREFLP (see Section 2). In particular, considering the computational complexity of the MREFLP, exact algorithms are limited to small instances (typically with fewer than 13 facilities) (Anjos et al., 2018). For larger instances, heuristic algorithms are employed for sub-optimal solutions. However, the state-of-the-art heuristic algorithm for the MREFLP, i.e., the GRASP approach (Uribe et al., 2021), struggles with instances of $n \geq 20$, in some instances achieving worse results than exact algorithms. This highlights the need for more efficient solution techniques.

It is found that the integration of learning-driven techniques and conventional heuristic approaches can achieve significant performance for many combinatorial optimization problems (Li et al., 2022; Zhou et al., 2018, 2016; Lu et al., 2024). These studies motivate us to design a novel approach which combines the powerful local optimization and the learning-driven strategies to solve the MREFLP. Thus, this work proposes an effective learning-driven multi-start local search algorithm (LMLS) to enhance the solution approaches of MREFLP. Our key contributions are as follows:

From an algorithm design perspective, the LMLS algorithm combines a learning-

driven construction procedure to create promising initial solutions and a dedicated two-phase local search procedure to strengthen the intensification search capability. In particular, the learning-driven construction generates promising initial solutions by utilizing the history information of the local optima previously found. The two-phase local search adopts two move operators for effective examination of the search space and a new fast neighborhood evaluation to reduce the computation time.

In terms of computational results, we report 148 new best results (improved upper bounds) and 404 best-known results (including 289 known optimal results) on a total of 552 commonly used benchmark instances. We also obtain 17 new best solutions on the 20 real-world instances. We will release our LMLS code to support future research on the MREFLP and its applications.

The paper proceeds as follows: Section 2 presents a literature review and identifies research gaps. Section 3 explains the proposed algorithm. Section 4 presents comparative results with the latest algorithms. Section 5 discusses the impact of the proposed algorithm's main components, and the final section concludes with future directions.

2 Literature review and the research gaps

This section overviews relevant algorithms and also addresses current research gaps for the MREFLP.

2.1 Literature review

Several exact algorithms have been developed to address the MREFLP in the literature. Anjos et al. (2018) introduced an integer linear programming (ILP) model and re-implemented a semidefinite programming (SDP) model, and three mixed integer linear programming (MILP) models: TAN (Chung and Tanchoco, 2010), AMA (Amaral, 2013), and AMA2 (Amaral, 2011). In addition, they generated 412 instances ($10 \leq n \leq 60$), which were created by extending instances from the single-row (Hungerlaender, 2014) and double-row facility layout problems (Amaral, 2013). Computational results show that ILP, SDP, and AMA2 dominate TAN and AMA by achieving no worse results on all instances. However, ILP, SDP, and AMA2 cannot dominate each other

because they achieve better solutions on different instances. In particular, ILP achieves 3 better results, SDP reports 106 better results, and AMA2 obtains 9 better results than the other models on 412 instances.

Due to the computational challenge of the MREFLP, efficient heuristic algorithms have been developed to obtain high-quality solutions in reasonable computation time. Uribe et al. (2021) proposed a greedy randomized adaptive search procedure (GRASP) for the MREFLP, which integrates a greedy randomized solution construction, a probabilistic mechanism, a local search, and a perturbation technique. The construction phase initializes a solution by randomly assigning r facilities to the first group. In each subsequent construction step, the remaining $k-1$ groups are considered in turn, and the unassigned vertices are greedily assigned to the groups one at a time. A probabilistic mechanism decides whether to accept the resulting solution or to generate another solution through the construction phase. The local search uses the exchange operator to find a local optimum solution. The perturbation adopts the insert operator to produce new starting solutions. GRASP obtains 19 better, 381 equal and 12 worse results compared to the best results in (Anjos et al., 2018) on 412 instances. Additionally, Uribe et al. (2021) generated 140 new large instances ($64 \leq n \leq 100$) and presented the first results on these instances.

According to the reported results, ILP, SDP, AMA2 and GRASP are the most advanced existing approaches currently available for addressing the problem. These approaches are thus used as reference algorithms in this study.

2.2 Research gaps

Although the existing algorithms have helped solve the challenging MREFLP problem, there is still room for improvement. First, recent approaches combining learning-driven methods with optimization techniques have significantly improved the performance of various algorithms for solving combinatorial optimization problems (Li et al., 2022; Zhou et al., 2018, 2016; Lu et al., 2024). However, this learning-driven optimization approach has yet to be used to solve the MREFLP. Second, it is known that two-phase local search with different well-designed neighborhoods can boost the search capacity of local optimization (Wei et al., 2023; Wei and Hao, 2019). However, this approach has not yet been applied to the MREFLP. Finally, the success of local search

depends on the use of fast incremental evaluation mechanisms for neighborhood examination, which play a critical role in the computation efficiency of each iteration (Fleurent and Ferland, 1996). Nevertheless, these techniques remain unexplored for the MREFLP. Even the most effective evaluation techniques in (Uribe et al., 2021) both require $O(n^3)$ time, to examine the entire neighborhood of the exchange and insert move operators.

To fill these gaps, this study presents the first learning-driven multi-start local search for the MREFLP. The main objective of this study is as follows: (1) enrich the existing literature by proposing a novel approach able to achieving high-quality solutions with reduced computation complexity. (2) make methodological contributions to integrate learning-driven techniques with optimization methods and to improve the popular multi-start local search method for general combinatorial optimization.

3 The learning-driven multi-start local search

Algorithm 1 The architecture of LMLS algorithm for the MREFLP

Input: Problem instance I (facilities set π, facility flow matrix w, group size limitation r, number of groups k, group distance matrix D), greediness ratio ϵ, depth of tabu search $maxIter$
Output: The recorded global best solution S^*

```
Initialize the learning-driven probability matrix η
S* ← ∅                                   /* Initialize the global solution S* */
f(S*) ← +∞                               /* Record the objective value of S* */
while stopping condition has not been verified do
    S0 ← Learning_Driven_Construction(I, η, ε)
            /* Create a new initial solution S0 using the matrix η, section 3.1 */
    S ← Two_Phase_LocalSearch(I, S0, maxIter)
                                         /* Improve S0, section 3.2 */
    if f(S) < f(S*) then
        S* ← S                           /* Update S* */
        f(S*) ← f(S)
    end if
    Update matrix η                      /* Section 3.3 */
end while
return S*
```

The proposed LMLS algorithm for the MREFLP derives from the multi-start local search framework (Martí et al., 2010) and iterates the learning-driven construction procedure and the two-phase local search procedure. Algorithm 1 and Figure 2 show the general architecture of the LMLS algorithm.

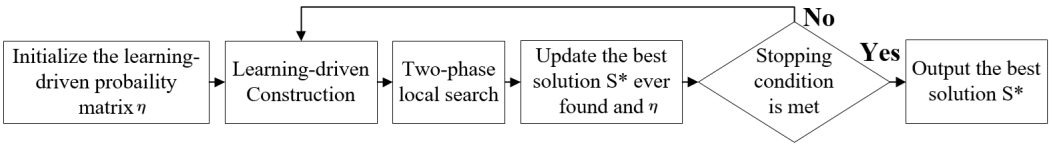


Figure 2. Flowchart of the proposed LMLS algorithm.

The proposed LMLS algorithm uses a probability matrix η to generate informed initial solutions (Section 3.1). It also uses several parameters: the greediness ratio ϵ for balancing quality and randomness of initial solutions (Section 3.1); the depth of tabu search $maxIter$ for balancing solution quality with computation time (Section 3.2.1).

3.1 Learning-driven construction

Algorithm 2 Learning-driven construction

Input: Problem instance I (facilities set π, facility flow matrix w, group size limitation r, number of groups k, group distance matrix D), probability matrix η, greediness ratio ϵ
Output: The initial solution $S_0 = \{V_1, V_2, \ldots, V_k\}$

for $i \leftarrow 1$ **to** k **do**
  $V_i \leftarrow \emptyset$
end for
$S_0 \leftarrow \{V_1, V_2, \ldots, V_k\}$
$\pi' \leftarrow \pi$ /* Initialize the set of unselected facilities π' */
$CG \leftarrow \{1, 2, \ldots, k\}$ /* Initialize the set of the candidate groups CG */
while $\pi' \neq \emptyset$ **do**
  Randomly select a facility $v \in \pi'$
  /* Choose a group i */
  if $rand(0, 1) < \epsilon$ **then**
    Select a group $i \in CG$ with the highest value η_{vi} in the v-th row of matrix η
  else
    Randomly choose a group $i \in CG$
  end if
  $V_i \leftarrow V_i \cup \{v\}$ /* Assign facility v to group i */
  $\pi' \leftarrow \pi' \backslash \{v\}$ /* Remove the chosen facility v from π' */
  if $|V_i| = r$ **then**
    $CG \leftarrow CG \backslash \{i\}$ /* Remove group i from CG */
  end if
end while
return $S_0 = \{V_1, V_2, ..., V_k\}$

The learning-driven construction aims to create promising initial solutions by taking advantage of historical information (Algorithm 2). It follows the idea of probability learning for grouping problems (Zhou et al., 2018, 2016). Un-

like conventional adaptive or self-adaptive methods that adjust search strategies and parameters according to search feedback (Eiben et al., 1999), our frequency-based reward/punishment mechanism learns from the internal structure of historical high-quality solutions. After each iteration, we assign rewards when facilities fall into the same group and impose penalties otherwise. Rather than tuning how the algorithm performs the search, our mechanism identifies promising facility position assignment patterns inherent to the problem itself.

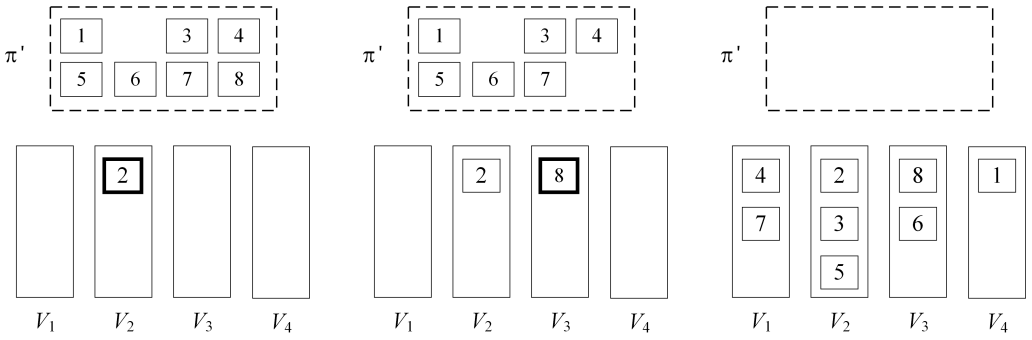
The method initializes the current solution $S_0 = \{V_1, V_2, \ldots, V_k\}$ with k empty groups, and iteratively assigns one facility to one group according to the probability matrix η of size $n \times k$ (lines 7-20). η_{vi} records the probability of facility v being assigned to group i (see Figure 3 for an example). η_{vi} is initialized to $\frac{1}{k}$, $\forall v \in \{1, 2, \ldots, n\}$, $\forall i \in \{1, 2, \ldots, k\}$ (line 1, Algorithm 1), and updated once the local search ends (Section 3.3). Specifically, to assign a facility to a group: 1) a random facility $v \in \pi'$ is selected (line 8), where π' represents the set of facilities that have not been chosen; 2) the group $i \in CG$ with the highest value in the v-th row of the matrix η is selected according to a probability of ϵ and a random group $i \in CG$ is chosen according to a probability of 1-ϵ (lines 10-14), where CG represents the set of candidate groups, which includes groups where the number of facilities assigned to the group is less than r; 3) facility v is assigned to group i and removed from π' (lines 15-16); 4) group i is removed from CG if the number of facilities assigned to group i equals to r (lines 17-19). The construction procedure ends when π' is empty.

$$\eta_{n\times k} = \begin{bmatrix} \eta_{11} & \eta_{12} & \cdots & \eta_{1k} \\ \eta_{21} & \eta_{22} & \cdots & \eta_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \eta_{n1} & \eta_{n2} & \cdots & \eta_{nk} \end{bmatrix}$$

Figure 3. An example of the probability matrix η

To illustrate this procedure, we analyze the instance in Figure 1 with 8 facilities $\pi = \{1, 2, \ldots, 8\}$, $r = 3$, and $k = 4$. Figure 4 provides an example where $S_0 = \{V_1, V_2, \ldots, V_k\}$, $V_i = \emptyset$ $(1 \leq i \leq k)$ and $\pi' = \pi = \{1, 2, \ldots, 8\}$ initially. First, facility 2 is randomly selected and the group 2 with the largest value η_{22} in the 2nd row of η is chosen in accordance with the greediness ratio ϵ and the probability matrix η. Then, facility 2 is assigned to group 2 and removed from

π'. Second, facility 8 is randomly selected and assigned to a random group 3 according to the greediness ratio ϵ. Then, facility 8 is removed from π'. This procedure continues until each facility is assigned to a group. In the end, a solution $S_0 = \{\{4, 7\}, \{2, 3, 5\}, \{8, 6\}, \{1\}\}$ is obtained.



(a) Facility 2 is assigned to group 2 according to the greediness ratio ϵ and the probability matrix η

(b) Facility 8 is assigned to group 3 according to the greediness ratio ϵ

(c) A new initial solution is obtained when all facilities are assigned to a group

Figure 4. An example of the learning-driven solution construction procedure

3.2 Two-phase local search

Algorithm 3 Two-phase local search

Input: Problem instance I (facilities set π, facility flow matrix w, group size limitation r, number of groups k, group distance matrix D), initial solution S, depth of tabu search $maxIter$

Output: The best solution S obtained throughout two-phase local phase

1: $S \leftarrow OneMove_TabuSearch(I, S, maxIter)$
/* Identify a local optimal solution S */
2: $S \leftarrow Swap_DescentSearch(I, S)$
/* Further improve S */
3: **return** S

The two-phase local search method has been used to tackle complex optimization problems like the multi-objective facility layout problem (Aurich et al., 2021), the colored traveling salesman problem (He and Hao, 2021), and the multidimensional knapsack problem (Lai et al., 2019). Our two-phase local search for the MREFLP (Algorithm 3) seeks to optimize the initial solution from the learning-driven construction procedure. The first phase ($OneMove$ tabu search) conducts a rapid and extensive search. The second phase (swap-based descent search) carries out a targeted and intensive search.

3.2.1 First Phase - OneMove tabu search

Our OneMove-based tabu search (Algorithm 4) follows the tabu search method (Glover and Laguna, 1998) to iteratively replace the current solution S with a best admissible neighbor solution S' until the recorded best solution in this search is not improved for a specified number of iterations (the depth of tabu search). Neighbor solutions are generated by applying the OneMove operator to S. Here, an admissible solution is a solution such that its corresponding move is not marked as tabu or if it surpasses the recorded best solution.

Algorithm 4 OneMove-based tabu search

Input: Problem instance I (facilities set π, facility flow matrix w, group size limitation r, number of groups k, group distance matrix D), input solution S, depth of tabu search $maxIter$

Output: The recorded best solution S^{best} during the tabu process

$S^{best} \leftarrow S$ /* Set S^{best} as the input solution S */
$NoImprove \leftarrow 0$ /* Initialize the number of consecutive iterations without improvement */
$Iter \leftarrow 0$
while $NoImprove < maxIter$ **do**
  Identify a best admissible neighbor solution $S' \in N_1(S)$
  $S \leftarrow S'$
  $Iter \leftarrow Iter + 1$
  Update the tabu list
  if $f(S) < f(S^{best})$ **then**
    $NoImprove \leftarrow 0$
    $S^{best} \leftarrow S$ /* Update S^{best} */
  else
    $NoImprove \leftarrow NoImprove + 1$
  end if
end while
return S^{best}

OneMove Operator: Given a solution S, the $OneMove(v, V_i, V_j)$ moves a facility v from its current group i to a distinct group j ($i \neq j$) satisfying that the group j has fewer facilities than r. Let $S \oplus OneMove(v, V_i, V_j)$ be the neighbor solution given by performing $OneMove(v, V_i, V_j)$ to S. The neighborhood N_1 induced by the $OneMove(v, V_i, V_j)$ operator is defined as:

$$N_1(S) = \{S \oplus OneMove(v, V_i, V_j) : v \in V_i, |V_j| < r, 1 \leq i \neq j \leq k\} \quad (7)$$

The size of N_1 is bounded by $O(n \times k)$.

Fast Neighborhood Evaluation Mechanism: To ensure high computational efficiency, we develop a fast incremental evaluation mechanism to calculate the incremental value (Δf) of the objective function f (Equation (1)) after the $OneMove(v, V_i, V_j)$ operator has been applied. This technique uses an $n \times k$ matrix δ, where δ_{vj} $(1 \leq v \leq n, 1 \leq j \leq k)$ stores the incremental value of moving facility v from its current group i to a distinct group j. The matrix δ is initialized as follows:

$$\delta_{vj} = \sum_{u=1}^{n} (|j - g| - |i - g|) w_{uv} \tag{8}$$

where g is the current group of facility u and w_{uv} is the flow from u to v.

Each time an $OneMove(v, V_i, V_j)$ involving facility v is performed, the move gain $\Delta f(OneMove(v, V_i, V_j))$ is obtained by

$$\Delta f(OneMove(v, V_i, V_j)) = f(S \oplus OneMove(v, V_i, V_j)) - f(S) = \delta_{vj} \tag{9}$$

When $OneMove(v, V_i, V_j)$ is performed, the matrix δ is updated accordingly.

1) For the moved facility v, the distance between facility v and other facilities needs to be computed from scratch, thus the value of the v-th row of matrix δ is calculated according to Equation (8).

2) For the facility $u \in \pi \backslash \{v\}$, we only need to update the distance between facility u and v affected by this move, then δ_{uh} $(1 \leq h \leq k)$ is updated by,

$$\delta_{uh} = \delta_{uh} + (|h - j| - |h - i| + |g - i| - |g - j|) w_{uv} \tag{10}$$

where the terms in Equation (10) have the following meanings. h and g denote the destination group and current group of facility u respectively. i and j denote the original group and destination group of the moved facility v. w_{uv} is the flow from facility u to facility v. $|h - j| - |h - i|$ is the distance change between facilities u and v when u is assigned to group h and v is moved from group i to group j. $|g-j|-|g-i|$ denotes the distance change between facilities u and v when u is assigned to group g and v is moved from group i to group j. The calculation of δ takes $O(kn)$ time.

Figure 5 presents the $OneMove$ operator's effect on a given solution $S =$

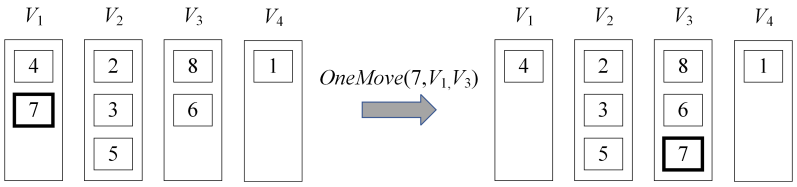


Figure 5. An example of the $OneMove$ operator

$\{\{4,7\},\{2,3,5\},\{8,6\},\{1\}\}$ with the objective value of 328. When moving facility 7 from V_1 to V_3, an improved neighboring solution $S \oplus OneMove(7, V_1, V_3)$ is obtained with the incremental objective value $\Delta f(OneMove(7, V_1, V_3)) = -102$, leading to the new objective value of 226.

Fast evaluation methods for the MREFLP are currently absent from the literature. For example, Uribe et al. (2021) compute the objective value of each candidate produced by the insert operator by recomputing distances between every facility and the moved facilities; consequently, exploring the entire insert neighborhood incurs $O(n^3)$ time complexity, severely slowing the search on large instances. The above technique fills thus this gap.

Tabu List Management Strategy: To avoid revisiting recent solutions, we use an n-dimensional vector T as a tabu list to track facilities involved in the $OneMove$ operator, with $T[v]$ $(1 \leq v \leq n)$ initialized to 0 and updated whenever facility v is moved,

$$T[v] = Iter + rand(A). \tag{11}$$

where $Iter$ is the iteration count, and $rand(A)$ represents a random integer from 1 to A. A is the threshold for the tabu tenure, which is set to 3 based on experiments (see Section 4.2.1). In subsequent iterations, if $Iter < T[v]$, facility v is prohibited from moving, unless the move involving v results in a solution superior to the recorded best solution S^{best} in the tabu process.

3.2.2 Second phase - Swap-based descent search

The swap-based descent search (Algorithm 5) seeks better solutions by exploring the neighborhood N_2 induced by the swap operator. During each iteration, the current solution S is replaced by the best neighbor solution S' in $N_2(S)$ if S' is superior (lines 3-8). This process halts upon reaching a local optimal.

Algorithm 5 Swap-based descent search

Input: Problem instance I (facilities set π, facility flow matrix w, group size limitation r, number of groups k, group distance matrix D), input solution S
Output: The updated solution S

```
NoImprove ← 0                          /* Denote whether S has been replaced */
while NoImprove = 0 do
    NoImprove ← 1
    Select the best neighbor solution S' ∈ N2(S)
    if f(S') < f(S) then
        S ← S'                                              /* Replace S */
        NoImprove ← 0
    end if
end while
return S
```

Swap Move Operator: The swap operator is the same as the exchange operator used in (Uribe et al., 2021). The $swap(u, v)$ operator exchanges facility u in group i with facility v in group j ($i \neq j$), leading to the following N_2 neighborhood of size of $O(n^2)$:

$$N_2(S) = \{S \oplus swap(u, v) : u \in V_i, v \in V_j, 1 \leq i \neq j \leq k\} \tag{12}$$

Fast Neighborhood Evaluation Mechanism: In (Uribe et al., 2021), each neighbor solution is evaluated by recalculating the distances between each facility and the moved facilities. Thus, examining the whole neighborhood is time-consuming with a complexity of $O(n^3)$. To solve this problem, we also design a fast incremental evaluation mechanism to quickly calculate the incremental value. Specifically, the $swap(u, v)$ move can be executed by combining two consecutive $OneMove$ operator, i.e., $S \oplus swap(u, v) = (S \oplus OneMove(u, V_i, V_j)) \oplus OneMove(v, V_j, V_i)$. As a result, the incremental value $\Delta f(swap(u, v))$ is obtained by,

$$\Delta f(swap(u, v)) = f(S \oplus swap(u, v)) - f(S) = \delta_{uj} + \delta_{vi} + 2w_{uv}|i - j| \tag{13}$$

where δ_{uj} is the incremental objective value for moving facility u from group i to j, δ_{vi} is the incremental objective value for moving facility v from group j to i. Both of them are initialized by Equation (8). Moreover, $w_{uv}|i - j|$ is the current cost between facilities u and v.

After performing a swap move, the matrix δ is updated twice using the up-

dating rules of the $OneMove$ operator (see Section 3.2.1). The computation of δ takes $O(kn)$ time.

Figure 6 shows the impact of the swap operator on a solution. We start with the recorded best solution during the $OneMove$ tabu search, i.e., $S = \{\{4\}, \{2, 3, 5\}, \{8, 6, 7\}, \{1\}\}$ with an objective value of 226. Swapping facility 5 and 1 yields an improved solution $S \oplus swap(5, 1)$. The incremental objective value $\Delta f(swap(5, 1)) = -98$ and the improved objective value is 128.

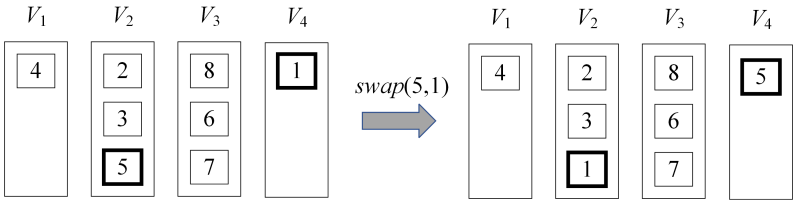


Figure 6. An example of the swap operator

3.3 Learning-driven probability matrix updating and smoothing

The probability updating strategy follows the probability learning approach of (Zhou et al., 2018, 2016), which is motivated by learning automata (Narendra and Thathachar, 1989). This strategy updates the probability matrix η through a comparison between the initial solution S_0 and the optimized solution S after each round of the two phase local search procedure. The update is performed according to the following two situations:

1) If facility $v \in \pi$ remains in the same group i, a reward is given to group i and penalize the remaining groups. Then, the v-th row of η is updated accordingly:

$$\eta_{vp} = \begin{cases} \alpha + (1-\alpha)\eta_{vp}, & p = i \\ (1-\alpha)\eta_{vp}, & otherwise. \end{cases} \quad (1 \leq p \leq k) \tag{14}$$

where the reward factor α $(0 < \alpha < 1)$ is set to 0.1 (see Section 4.2.1).

2) If the facility v is reassigned to group j from its original group i $(i \neq j)$ in the optimized solution S, we apply a penalty to its original group i, and a

compensation to the group j. The v-th row of η is updated as follows:

$$\eta_{vp} = \begin{cases} (1-\beta)(1-\gamma)\eta_{vp}, & p = i \\ \gamma + (1-\gamma)\frac{\beta}{k-1} + (1-\beta)(1-\gamma)\eta_{vp}, & p = j \\ (1-\gamma)\frac{\beta}{k-1} + (1-\beta)(1-\gamma)\eta_{vp}, & otherwise. \end{cases} \quad (1 \le p \le k) \quad (15)$$

where the penalization factor β $(0 < \beta < 1)$ and the compensation factor γ $(0 < \gamma < 1)$ are set to 0.2 and 0.3, respectively (see Section 4.2.1).

Figure 7 shows an example of the probability updating procedure with the input solution $S_0 = \{\{4,7\},\{2,3,5\},\{8,6\},\{1\}\}$ and the improved solution $S = \{\{4\},\{2,3,1\},\{8,6,7\},\{5\}\}$. Here facilities 2, 3, 4, 6, and 8 remain in their groups (the first situation). Thus, we update (in red) their respective row value in matrix η according to Equation (14). In contrast, facilities 1, 5, and 7 have changed groups (the second situation). Therefore, we update (in blue) the corresponding row values in the matrix using Equation (15).

$$\eta_{8\times4} = \begin{bmatrix} 0.220 & 0.420 & 0.140 & 0.220 \\ 0.225 & 0.325 & 0.225 & 0.225 \\ 0.225 & 0.325 & 0.225 & 0.225 \\ 0.325 & 0.225 & 0.225 & 0.225 \\ 0.220 & 0.140 & 0.420 & 0.220 \\ 0.225 & 0.225 & 0.325 & 0.225 \\ 0.140 & 0.220 & 0.220 & 0.420 \\ 0.225 & 0.225 & 0.325 & 0.250 \end{bmatrix}$$

(a) Probability matrix before learning-driven update

$$\eta_{8\times4} = \begin{bmatrix} 0.170\downarrow & \mathbf{0.581}\uparrow & \underline{0.078}\downarrow & 0.170\downarrow \\ 0.203\downarrow & \underline{\mathbf{0.393}}\uparrow & 0.203\downarrow & 0.203\downarrow \\ 0.203\downarrow & \underline{\mathbf{0.393}}\uparrow & 0.203\downarrow & 0.203\downarrow \\ \underline{\mathbf{0.393}}\uparrow & 0.203\downarrow & 0.203\downarrow & 0.203\downarrow \\ 0.170\downarrow & \underline{0.078}\downarrow & \mathbf{0.581}\uparrow & 0.170\downarrow \\ 0.203\downarrow & 0.203\downarrow & \underline{\mathbf{0.393}}\uparrow & 0.203\downarrow \\ \underline{0.078}\downarrow & 0.170\downarrow & 0.170\downarrow & \mathbf{0.581}\uparrow \\ 0.203\downarrow & 0.203\downarrow & \underline{\mathbf{0.393}}\uparrow & 0.203\downarrow \end{bmatrix}$$

(b) Probability matrix after learning-driven update

Figure 7. An example of learning-driven probability matrix updating procedure (rows highlighted in red indicate facilities that remain in their groups, and rows highlighted in blue indicate facilities that change groups, with the bold font representing the facility's group after the change, the underline representing the facility's group before the change, and an upward arrow indicates that the corresponding value has increased after the update, while a downward arrow signifies a decrease)

Finally, a probability smoothing strategy is used to avoid adverse effects of historical information on the current search. Once an element η_{vp} $(1 \le v \le n, 1 \le p \le k)$ in the learning-driven probability matrix reaches a specified

threshold, it is adjusted by a smoothing factor (i.e., ρ, $0 < \rho < 1$) to downplay outdated information as follows.

$$\eta_{vp} = \begin{cases} \rho + (1-\rho)\eta_{vp}, \; if \;\; \eta_{vp} < 0.05 \\ (1-\rho)\eta_{vp}, \qquad if \;\; \eta_{vp} > 0.95. \end{cases} \tag{16}$$

3.4 Computation complexity

The learning-driven construction procedure requires $O(k \times n)$ time. The two-phase local search takes $O((K_1 \times k + K_2 \times n + k \times n) \times n)$ time, with K_1 and K_2 representing the tabu search and the descent search iterations, respectively. The probability updating and smoothing procedure requires $O(k \times n)$ time. Thus, one iteration of the LMLS costs $O((K_1 \times k + K_2 \times n + k \times n) \times n)$ time.

4 Computational evaluation and comparative study

We evaluated the LMLS algorithm on 552 conventional benchmark instances from (Anjos et al., 2018; Uribe et al., 2021), and 20 real-world instances (Eschermann and Wunderlich, 1990; Taillard, 1995) introduced in this work. Among the conventional benchmarks, the optimal objective values for 289 instances are given in (Anjos et al., 2018). The best results (best upper bounds) for the rest instances are from (Anjos et al., 2018) and (Uribe et al., 2021).

4.1 Benchmark instances

The 572 instances are divided into four categories. These instances are available at https://github.com/QYseu/MREFLP.

1. **Small (216 instances):** The first category, introduced in (Anjos et al., 2018), consists of 216 small instances named with prefixes A*, Y*, O* and S*, among which the optimal values of 210 instances are already known. These instances have the number of facilities n within $[10, 14]$, the flow between each facility within $[0, 37]$, the number of groups k within $[3, 9]$, and the group size limit $r \in \{2, 3, 4, 5\}$.

2. **Medium (196 instances):** The second category, also introduced in (Anjos et al., 2018), comprises 196 medium instances named with A*, Y*, O*, S* and N*, with 79 known optimal values. These instances have the number of facilities n within $[15, 60]$, the flow between each facility within $[0, 26]$, the number of groups k within $[5, 39]$, and the group size limit $r \in \{2, 3, 4, 5\}$.
3. **Large (140 instances):** The third category, introduced in (Uribe et al., 2021), contains 140 large instances named with AKV* or sko*, where the AKV* instances were randomly generated in (Anjos et al., 2005), and the sko* instances are from the unified testing platform QAPlib (Skorin-Kapov, 1990). Each instance has an n within $[64, 100]$ facilities, the flow between each facility within $[0, 16]$, the number of groups k within $[21, 66]$, and group size limit $r \in \{2, 3, 4, 5\}$. The optimal solutions are unknown.
4. **Real-World (20 instances):** The fourth category, introduced in this paper, comprises 20 real-world instances named with esc* or tai*, where the esc* instances are from the problem of self-testable sequential circuits (Eschermann and Wunderlich, 1990), and the tai* instances are from the quadratic assignment problem (Taillard, 1995). These instances have n within $[32, 256]$, the flow between facilities within $[0, 100000]$, the number of groups k within $[10, 170]$, and the group size limit $r \in \{2, 3, 4, 5\}$. The optimal solutions for these instances are unknown.

4.2 Experimental Protocol

The LMLS algorithm was implemented in C++ and compiled using GNU g++ 4.1.2 with the -O3 option. The experiments were performed on a 2.40GHz Intel Core 2 Duo T7700 processor with 2GB RAM, running Linux. The algorithm code will be released publicly along with the paper's publication.

4.2.1 Parameters

Our LMLS algorithm relies on 7 parameters: the greediness ratio ϵ in construction phase; the depth of $OneMove$ tabu search $maxIter$, the tabu tenure threshold A in the two-phase local search phase, the reward factor α, the penalization factor β, the compensation factor γ and the smoothing coefficient ρ in the learning-driven probability matrix updating and smoothing phase. We

used the automatic parameter configuration tool "irace" to calibrate these parameters, testing on 30 challenging instances (as indicated in Table A.9 - A.16 at https://github.com/QYseu/MREFLP) with a limit of 200 runs. Table 1 displays the candidate alongside the "irace"-determined final parameter values, which were employed in all subsequent experiments. Note that fine-tuning the parameters could yield still better results.

Table 1
Parameter settings of the LMLS algorithm

Parameters	Section	Description	Candidate values	Final value
ϵ	3.1	The greediness ratio of learning-driven solution construction	{0.5, 0.6, 0.7, 0.8, 0.9}	0.6
$maxIter$	3.2.1	The search depth of $OneMove$ tabu search	{10, 20, 50, 100, 200}	20
A	3.2.1	The tabu tenure threshold of $OneMove$ tabu search	{3, 5, 7, 8, 9}	3
α	3.3	The reward factor of learning-driven probability matrix η	{0.1, 0.2, 0.3, 0.4, 0.5}	0.1
β	3.3	The penalization factor of learning-driven probability matrix η	{0.1, 0.2, 0.3, 0.4, 0.5}	0.2
γ	3.3	The compensation factor of learning-driven probability matrix η	{0.1, 0.2, 0.3, 0.4, 0.5}	0.3
ρ	3.3	The smoothing coefficient of learning-driven probability matrix η	{0.1, 0.2, 0.3, 0.4, 0.5}	0.3

4.2.2 Reference algorithms

To evaluate the algorithm LMLS, we used 4 latest algorithms for comparison.

1. The CPLEX solver (ILP) (Anjos et al., 2018) (run on a 3.4GHz Intel i7 4770 processor with 32GB RAM);
2. The semidefinite programming (SDP) (Anjos et al., 2018) (run on a 3.4GHz Intel i7 4770 processor with 32GB RAM);
3. The CPLEX solver (AMA2) (Anjos et al., 2018) (run on a 3.4GHz Intel i7 4770 processor with 32GB RAM);
4. The greedy randomized adaptive search procedure (GRASP) (Uribe et al., 2021) (run on a 2.6GHz Intel i7 6700HQ processor with 16GB RAM).

We re-implemented the ILP model (re-ILP), and used it to solve all instances on our computer, since the results for some instances of small and medium categories for ILP are not available in (Anjos et al., 2018). For instances with known optimal results, both re-ILP and ILP can obtain the same results. Furthermore, for the other instances that results are available, re-ILP dominated ILP by achieving no worse results under the same time limit. Moreover, We ran the GRASP code from https://zenodo.org/records/14717425 on our computer to solve real-world instances, whereas the results for small, medium, and large categories instances are cited from (Uribe et al., 2021).

4.2.3 Stopping conditions

Cutoff time of each run: Our comparative study concentrates on the solution quality. Nevertheless, we also present the information about the computation time. Following (Usberti et al., 2013; Chen and Hao, 2017), we normalized the computation time based on our CPU frequency (Intel Core T7700 2.40GHz) (scaling factors in Table 2).

Table 2
Scaling factors for the computers used by the reference algorithms. Our computer (Intel Core T7700) serves as the basis.

Algorithm	Reference	Processor type	Frequency (GHz)	Factor
LMLS	-	Intel Core T7700	2.4	1.0
ILP	(Anjos et al., 2018)	Intel i7 4770	3.4	1.42
AMA2	(Anjos et al., 2018)	Intel i7 4770	3.4	1.42
SDP	(Anjos et al., 2018)	Intel i7 4770	3.4	1.42
GRASP	(Uribe et al., 2021)	Intel i7 6700HQ	2.6	1.08

We configured our algorithm's stopping criteria based on the scaling factors in Table 2 to ensure comparability with the strictest conditions used by the reference algorithms. Following the 600-second cutoff time for all instances of the GRASP algorithm in (Uribe et al., 2021), we set 600-second cutoff time for all instances in our algorithm. This is a more restrictive stopping condition since GRASP was run on a 1.08-times faster machine.

Single run. According to (Uribe et al., 2021), in our first experiment (called the normal test), we ran our LMLS algorithm once for each instance with the 600 seconds cutoff time.

Multiple runs. Considering the stochastic nature of our algorithm, in our second experiment (called the relaxed test), we ran our LMLS algorithm 10 times for each instance with different random seeds, each run using the same cutoff time limit with normal test, i.e., 600 seconds.

4.3 Comparative study

Table 3 summarizes the comparative results under the normal test condition. Column 1 lists the algorithms compared, and column 2 names the instance categories and their sizes. Columns 3-5 report the number of instances where LMLS outperforms, matches, or underperforms re-ILP, SDP, AMA2, GRASP and best known values (BKV) from the literature in terms of the objective

value (f) and computation time ($time$). Column 6 presents the Friedman test p-values to check significant difference in terms of the objective value and time. The "NA" means that the same results were achieved by the algorithms compared. The p-values below 0.05 are shown in bold to highlight a significant difference between the two compared algorithms. Note that the results for some instances of small/medium categories for SDP and AMA2 are not available in (Anjos et al., 2018). Detailed results for four categories of instances are available at https://github.com/QYseu/MREFLP, and the optimal values are indicated by asterisks "*".

Table 3
Summary of comparative results under the normal test condition between LMLS, re-ILP, SDP, AMA2 and BKV in terms of objective value and computation time in parenthesis.

Algorithm pair	Category/Number	Better	Equal	Worse	p-value
LMLS vs re-ILP	Small/216	93 (216)	123 (0)	0 (0)	**5.23E-23** (**6.74E-49**)
	Medium/196	171 (196)	25 (0)	0 (0)	**3.35E-38** (**1.16E-43**)
	Large/140	140 (140)	0 (0)	0 (0)	**2.66E-32** (**2.66E-32**)
	Real-World/20	20 (20)	0 (0)	0 (0)	**7.74E-06** (**7.74E-06**)
LMLS vs SDP	Small/208	0 (204)	208 (4)	0 (0)	NA (**3.04E-48**)
	Medium/138	21 (138)	117 (0)	0 (0)	**3.00E-04** (**1.16E-43**)
LMLS vs AMA2	Small/189	0 (124)	189 (47)	0 (18)	NA (**2.66E-14**)
	Medium/39	20 (37)	19 (0)	0 (2)	**4.59E-06** (**2.33E-21**)
LMLS vs GRASP	Small/216	0 (165)	216 (2)	0 (49)	NA (**1.47E-15**)
	Medium/196	20 (142)	176 (0)	0 (54)	**4.59E-06** (**1.34E-09**)
	Large/140	138 (120)	2 (0)	0 (20)	**7.81E-18** (**2.87E-17**)
	Real-World/20	17 (18)	3 (0)	0 (2)	**3.74E-05** (**3.47E-04**)
LMLS vs BKV	Small/216	0 (140)	216 (55)	0 (21)	NA (**4.17E-21**)
	Medium/196	10 (145)	176 (0)	0 (51)	**1.6E-3** (**8.29E-11**)
	Large/140	138 (120)	2 (0)	0 (20)	**7.81E-18** (**2.87E-17**)
	Real-World/20	17 (18)	3 (0)	0 (2)	**3.74E-05** (**3.47E-04**)

The results of Table 3 indicate that LMLS dominates all reference algorithms on all benchmark instances under the normal test condition. Specifically, 1) compared to re-ILP, LMLS easily improves 424 best results and finds 148 known optimal results; 2) LMLS surpasses SDP with 21 better and 325 equal results out of 346 instances; 3) compared to AMA2, LMLS obtains 20 better results, and matches the remaining 208 known optimal results; 4) LMLS dominates GRASP with 175 better and 397 equal results on

572 instances, with significant differences indicated by p-values of 4.59E-06 (Medium), 7.81E-18 (Large), and 3.74E-05 (Real-World); 5) LMLS also outperforms BKV by finding 165 better results and 407 equal results (including 289 known optimal results) on 572 instances, with a p-value of 1.6E-03 (Medium), 7.81E-18 (Large), and 3.74E-05 (Real-World), confirming significant differences. Besides, it consumes less time on 572/342/161/446/424 instances than re-ILP/SDP/AMA2/GRASP/BKV, despite their use of more powerful computers.

Finally, we tested the performance of our LMLS algorithm under the relaxed test with 10 runs with the 600 seconds cutoff time. Table 4 presents a comparative analysis of LMLS's performance under 10 runs (denoted as LMLS_relaxed) versus the normal test with one run (LMLS), along with the corresponding p-values derived from the Friedman signed-rank test (detailed results are provided in the Appendix). These results reveal that LMLS_relaxed achieves 21 better results and matches 551 results.

Table 4
Summary of comparative results between LMLS using the relaxed test (LMLS_relaxed) and LMLS using the normal test (LMLS)

Algorithm pair	Category/Number	Indicator	Better	Equal	Worse	p-value
LMLS_relaxed vs LMLS	Small/216	f	0	216	0	NA
	Medium/196	f	0	196	0	NA
	Large/140	f	18	122	0	**2.21E-05**
	Real-World/20	f	3	17	0	8.33E-02

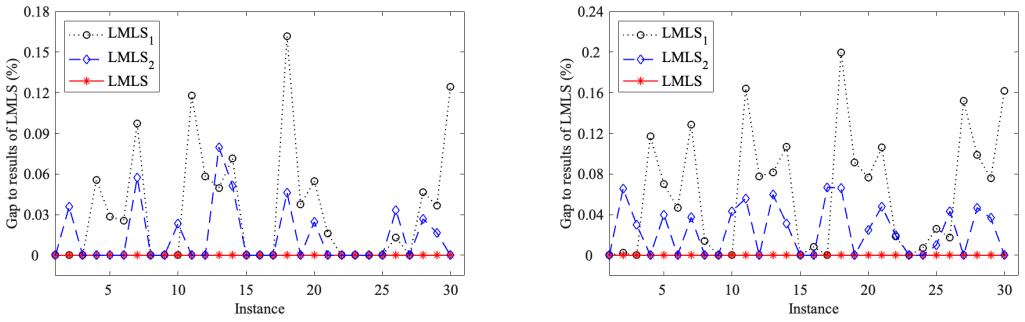
5 Analysis

The main components of LMLS include the learning-driven construction, the two-phase local search, and the incremental evaluation mechanism. We investigate their impact on the performance of LMLS. In this study, experiments are performed on 30 most difficult instances. Each investigated algorithm variant was run 10 times per instance, with the cutoff time of 600 seconds, where f_{best} is the best results and f_{avg} is the average results of 10 runs.

5.1 Impact of learning-driven mechanism

As described in Sections 3.1, LMLS applies a learning mechanism in the solution construction procedure. To illustrate its merit, two variants of LMLS

(LMLS$_1$ and LMLS$_2$) are designed for comparison. LMLS$_1$ and LMLS$_2$ remove the learning mechanism while keeping the local search procedure. LMLS$_1$ randomly constructs a solution while LMLS$_2$ greedily constructs a solution in the construction procedure without using the learning-driven probability matrix η. Figure 8 compares their f_{best} and f_{avg} of 10 runs. The X-axis lists the instances numbered as per Table A.9 - A.16 on the website [1], while the Y-axis depicts the percentage gap between f_{best} or f_{avg} and the new upper bound (NUB) from LMLS, calculated as $gap = \frac{f_{best}-NUB}{NUB} \times 100\%$ for Figure 8(a) and $gap = \frac{f_{avg}-NUB}{NUB} \times 100\%$ for Figure 8(b). Figure 8 shows that LMLS has



(a) Best results (f_{best}) (b) Average results (f_{avg})

Figure 8. Comparisons of LMLS (in red) with its variant LMLS$_1$ (in black) and LMLS$_2$ (in blue)

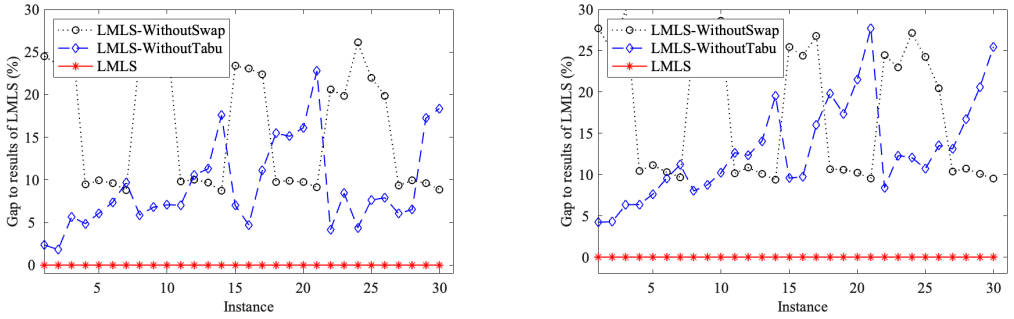
a superior performance regarding f_{best} and f_{avg}. Specifically, in terms of f_{best} (f_{avg}), LMLS achieves 16 (23) better and 14 (7) equal results compared to the variant LMLS$_1$. LMLS dominates LMLS$_2$ by reporting 10 (17) better and 20 (13) equal results in terms of f_{best} (f_{avg}). Moreover, the small p-values (4.23E-10 and 4.32E-08) highlight that the differences in performance between the compared algorithms are statistically significant.

5.2 Impact of the two-phase local search

To investigate the impact of the two-phase local search, we design two variants LMLS-WithoutTabu (without the tabu search) and LMLS-WithoutSwap (without the descent search).

Figure 9 compares f_{best} and f_{avg} among LMLS, LMLS-WithoutTabu and LMLS-WithoutSwap. LMLS outperforms both variants because it can achieve 30 better results in terms of f_{best} and f_{avg} respectively. Besides, the small p-

[1] https://github.com/QYseu/MREFLP



(a) Best results (f_{best}) (b) Average results (f_{avg})

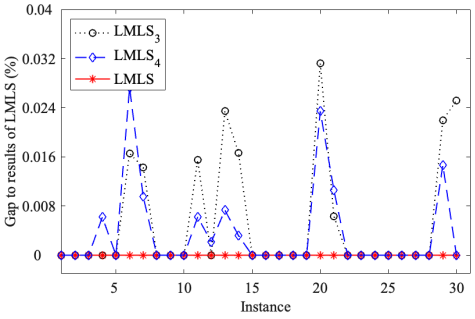
Figure 9. Comparisons of LMLS (in red) with its variant LMLS-WithoutTabu (in blue) and LMLS-WithoutSwap (in black)

values (5.71E-15 and 5.71E-15) demonstrate the statistically significant differences between the compared results. The experiment confirms that the two-phase local search strategy is useful for solving the MREFLP.

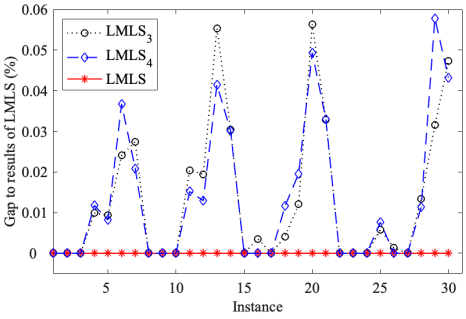
5.3 Impact of the incremental evaluation mechanism

To evaluate the impact of the fast incremental evaluation mechanism of the $OneMove$ and swap operators, we compared LMLS with two variants: LMLS_3 (without fast evaluation mechanism for the $OneMove$ operator) and LMLS_4 (without fast evaluation mechanism for the swap operator).

Figure 10 shows f_{best} and f_{avg} of LMLS, LMLS_3 and LMLS_4. In terms of f_{best}, LMLS dominates LMLS_3 with 9 better, 21 equal results and dominates LMLS_4 with 10 better, 20 equal results. As for f_{avg}, LMLS outperforms LMLS_3 for 18 instances and LMLS_4 for 16 instances. This confirms that the fast evaluation strategies are crucial for the high performance of LMLS, as their removal greatly deteriorates the effectiveness of the algorithm. Moreover, the statistically significant differences between LMLS and the two variants are confirmed by the p-values (3.41E-07 and 2.03E-07).

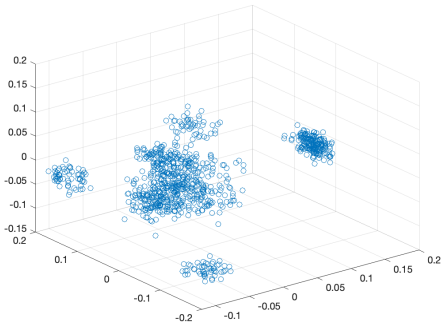


(a) Best results (f_{best})

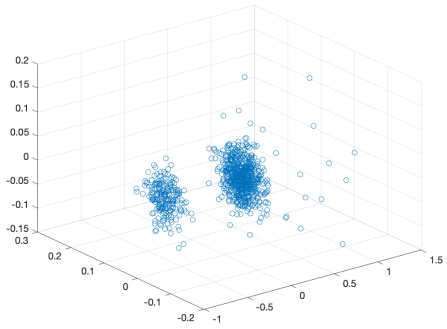


(b) Average results (f_{avg})

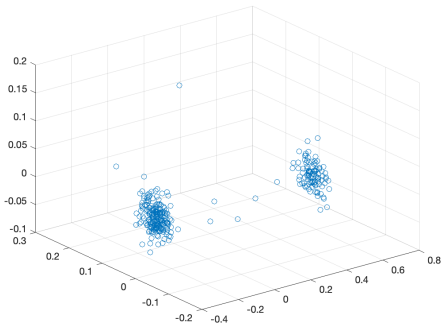
Figure 10. Comparisons of LMLS (in red) with its variants $LMLS_3$ (in black) and $LMLS_4$ (in blue)



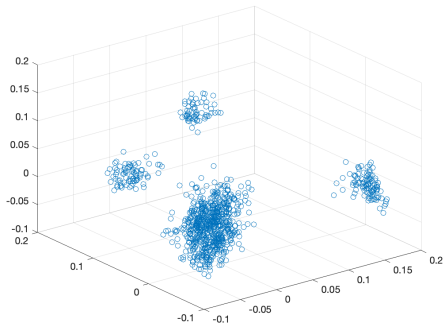
(a) Distribution of 900 local optima for sko81 with r=3



(b) Distribution of 924 local optima for sko81 with r=5



(c) Distribution of 855 local optima for sko100 with r=3



(d) Distribution of 887 local optima for sko100 with r=5

Figure 11. Distribution of the high-quality local optima for four instances

5.4 Search space cartography

To understand the importance of the learning-driven construction procedure (Section 3.1), we analyzed the spatial distribution of high-quality solutions,

following (Lai and Hao, 2016; Porumbel et al., 2010). We performed LMLS algorithm 10 runs on 4 instances (sko81 with r=3,5 and sko100 with r=3,5), collected all high-quality local optima encountered by LMLS to analyze their spatial distribution. A solution S is deemed high-quality when its objective value $f(S)$ does not exceed $1.001 \times NUB$ ($f(S) < 1.001 \times NUB$). We used the multidimensional scaling to illustrate the spatial distribution of high-quality solutions in the Euclidean space R^3 (Lai and Hao, 2016; Porumbel et al., 2010). We began by computing the distance matrix $SD_{l\times l}$, where l denotes the number of local optima, and $sd_{ab} \in SD_{l\times l}$ is the distance between solutions $S_a = \{y^a[1], y^a[2], ..., y^a[n]\}$ and $S_b = \{y^b[1], y^b[2], ..., y^b[n]\}$, in which $y^{a(b)}[u] = i$ if facility u belongs to group i in solution $S_a(S_b)$. sd_{ab} is defined as follows:

$$sd_{ab} = \frac{|(u,v) : (y^a[u] = y^a[v] \wedge y^b[u] \neq y^b[v]) \vee (y^a[u] \neq y^a[v] \wedge y^b[u] = y^b[v])|}{\frac{n^2}{k}} \tag{17}$$

Then, using *cmdscale*, we derived l coordinates in R^3 to minimize the distance distortion. Lastly, we draw scatter plot of these points in R^3.

Figure 11 displays the spatial distribution of local optima found by LMLS for each selected instance. These solutions form clusters with small-radius spheres, indicating that high-quality neighbor solutions can be obtained close to the high-quality solutions found. The clusters are also well separated, suggesting the need for significant diversification to move between promising regions, which LMLS achieves through its learning-driven solution construction. This experiment provides valuable insights for the design of our algorithm.

6 Conclusion

The multiple row equal facility layout problem is a versatile model for diverse applications. Nevertheless, solving the problem remains challenging. This paper presents a learning-driven multi-start local search algorithm to address this computationally difficult problem. Our algorithm integrates two techniques: a learning-driven solution construction procedure to create promising initial solutions, and a two-phase local search procedure with new incremental evaluation techniques to effectively optimize the constructed solutions.

We assessed the performance of LMLS on 572 benchmark instances versus the

latest algorithms. The comparisons revealed that LMLS performs exceptionally well by obtaining 165 new records (new upper bounds) and finding the best-known solutions of the 407 rest instances (including 289 known optima). In addition, the advantages of the learning-driven mechanism, the two-phase local search, and the incremental evaluation mechanism were demonstrated.

For future research, we can advance two directions. First, other machine learning-based approaches would be investigated, where the proposed algorithm can be advantageously used as a critical intensification component. Second, given its excellent performance for the layout problem, it would be interesting to adapt the proposed algorithm to other variants of layout problems, such as the T-row and the multi-bay facility layout problem (Dahlbeck, 2021), the space-free multi-row facility layout problem (Wu et al., 2024), and the combined cell layout problem (Dahlbeck et al., 2023).

7 Acknowledgments

We are grateful to the reviewers for their useful comments. This work was partially supported by the National Natural Science Foundation of China [Grant No. 62101125, No. 72122006], and Fundamental Research Funds for the Central Universities, China [Grant No. 2242022K60005].

8 Appendix

Detailed comparative results for the 572 instances tested: https://ars.els-cdn.com/content/image/1-s2.0-S0377221726007654-mmc1.xlsx

References

Amaral, A. R. (2011). On duplex arrangement of vertices. Technical report, Departamento de Informatica, Universidade Federal do Espirito Santo, Brazil.

Amaral, A. R. (2013). Optimal solutions for the double row layout problem. *Optimization Letters*, 7(2):407–413. https://doi.org/10.1007/s11590-011-0426-8.

Anjos, M. F., Fischer, A., and Hungerländer, P. (2018). Improved exact approaches for row layout problems with departments of equal length. *Euro-*

pean Journal of Operational Research, 270(2):514–529. https://doi.org/10.1016/j.ejor.2018.04.008.

Anjos, M. F., Kennings, A., and Vannelli, A. (2005). A semidefinite optimization approach for the single-row layout problem with unequal dimensions. *Discrete Optimization*, 2(2):113–122. https://doi.org/10.1016/j.disopt.2005.03.001.

Aurich, P., Speckmann, M., Böning, C., and Stonis, M. (2021). A two-stage tabu search for multi-objective facility layout problem. In *Proceedings of the Conference on Production Systems and Logistics*, pages 524–534. Hannover Publish-Ing.

Chen, Y. N. and Hao, J. K. (2017). An iterated "hyperplane exploration" approach for the quadratic knapsack problem. *Computers and Operations Research*, 77:226–239. https://doi.org/10.1016/j.cor.2016.08.006.

Chung, J. and Tanchoco, J. (2010). The double row layout problem. *International Journal of Production Research*, 48(3):709–727. https://doi.org/10.1080/00207540802192126.

Dahlbeck, M. (2021). A mixed-integer linear programming approach for the t-row and the multi-bay facility layout problem. *European journal of operational research*, 295(2):443–462. https://doi.org/10.1016/j.ejor.2021.02.044.

Dahlbeck, M., Fischer, A., Fischer, F., Hungerländer, P., and Maier, K. (2023). Exact approaches for the combined cell layout problem. *European Journal of Operational Research*, 305(2):530–546. https://doi.org/10.1016/j.ejor.2022.04.031.

Eggers, J., Feillet, D., Kehl, S., Wagner, M. O., and Yannou, B. (2003). Optimization of the keyboard arrangement problem using an ant colony algorithm. *European Journal of Operational Research*, 148(3):672–686. https://doi.org/10.1016/S0377-2217(02)00489-7.

Eiben, A., Hinterding, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141. https://doi.org/10.1109/4235.771166.

Eschermann, B. and Wunderlich, H. (1990). Optimized synthesis of self-testable finite state machines. In *Proceedings of the 20th International Symposium on Fault-Tolerant Computing*, pages 390–397. IEEE Computer Society. https://doi.org/10.1109/FTCS.1990.89393.

Fleurent, C. and Ferland, J. A. (1996). Genetic and hybrid algorithms for graph coloring. *Annal of Operations Research*, 63(3):437–461. https://doi.

org/10.1007/BF02125407.

Glover, F. and Laguna, M. (1998). Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. https://doi.org/10.1007/978-1-4615-6089-0.

He, P. F. and Hao, J. K. (2021). Iterated two-phase local search for the colored traveling salesmen problem. *Engineering Applications of Artificial Intelligence*, 97:104018. https://doi.org/10.1016/j.engappai.2020.104018.

Heragu, S. S. and Kusiak, A. (1988). Machine layout problem in flexible manufacturing systems. *Operations Research*, 36(2):258–268. https://doi.org/10.1287/opre.36.2.258.

Hungerlaender, P. (2014). Single-row equidistant facility layout as a special case of single-row facility layout. *International Journal of Production Research*, 52(5):1257–1268. https://doi.org/10.1080/00207543.2013.828163.

Lai, X. J. and Hao, J. K. (2016). Iterated maxima search for the maximally diverse grouping problem. *European Journal of Operational Research*, 254(3):780–800. https://doi.org/10.1016/j.ejor.2016.05.018.

Lai, X. J., Hao, J. K., and Yue, D. (2019). Two-stage solution-based tabu search for the multidemand multidimensional knapsack problem. *European Journal of Operational Research*, 274(1):35–48. https://doi.org/10.1016/j.ejor.2018.10.001.

Li, M. J., Hao, J. K., and Wu, Q. H. (2022). Learning-driven feasible and infeasible tabu search for airport gate assignment. *European Journal of Operational Research*, 302(1):172–186. https://doi.org/10.1016/j.ejor.2021.12.019.

Lin, Q. L., Liu, H. C., Wang, D. J., and Liu, L. (2015). Integrating systematic layout planning with fuzzy constraint theory to design and optimize the facility layout for operating theatre in hospitals. *Journal of Intelligent Manufacturing*, 26(1):87–95. https://doi.org/10.1007/s10845-013-0764-8.

Lu, Z., Gao, J., Hao, J. K., Yang, P. L., and Zhou, L. X. (2024). Learning driven three-phase search for the maximum independent union of cliques problem. *Computers and Operations Research*, 164:106549. https://doi.org/10.1016/j.cor.2024.106549.

Martí, R., Moreno-Vega, J. M., and Duarte, A. (2010). Advanced multi-start methods. In *Handbook of metaheuristics*, pages 265–281. https://doi.org/10.1007/978-1-4419-1665-5_9.

Narendra, K. S. and Thathachar, M. A. L. (1989). *Learning automata: an introduction*, pages 35–40. Prentice Hall. https://doi.org/10.1007/

978-1-4613-0947-5.

Porumbel, D. C., Hao, J. K., and Kuntz, P. (2010). A search space "cartography" for guiding graph coloring heuristics. *Computers and Operations Research*, 37(4):769–778. https://doi.org/10.1016/j.cor.2009.06.024.

Skorin-Kapov, J. (1990). Tabu search applied to the quadratic assignment problem. *INFORMS Journal on Computing*, 2(1):33–45. https://doi.org/10.1287/ijoc.2.1.33.

Taillard, E. D. (1995). Comparison of iterative searches for the quadratic assignment problem. *Location Science*, 3(2):87–105. https://doi.org/10.1016/0966-8349(95)00008-6.

Uribe, N. R., Herrán, A., Colmenar, J. M., and Duarte, A. (2021). An improved grasp method for the multiple row equal facility layout problem. *Expert Systems with Applications*, 182:115184. https://doi.org/10.1016/j.eswa.2021.115184.

Usberti, F. L., França, P. M., and França, A. L. M. (2013). Grasp with evolutionary path-relinking for the capacitated arc routing problem. *Computers and Operations Research*, 40(12):3206–3217. https://doi.org/10.1016/j.cor.2011.10.014.

Wei, Z. Q. and Hao, J. K. (2019). Iterated two-phase local search for the set-union knapsack problem. *Future Generation Computer Systems*, 101:1005–1017. https://doi.org/10.1016/j.future.2019.07.062.

Wei, Z. Q., Hao, J. K., Ren, J. T., and Glover, F. (2023). Responsive strategic oscillation for solving the disjunctively constrained knapsack problem. *European Journal of Operational Research*, 309(3):993–1009. https://doi.org/10.1016/j.ejor.2023.02.009.

Wess, B. and Zeitlhofer, T. (2004). On the phase coupling problem between data memory layout generation and address pointer assignment. In *Software and Compilers for Embedded Systems*, pages 152–166, Berlin, Heidelberg. Springer. https://doi.org/10.1007/978-3-540-30113-4_12.

Wu, S., Yang, W., Hanafi, S., Wilbaut, C., and Wang, Y. (2024). Iterated local search with ejection chains for the space-free multi-row facility layout problem. *European Journal of Operational Research*, 316(3):873–886. https://doi.org/10.1016/j.ejor.2024.03.012.

Zhou, Y., Duval, B., and Hao, J. K. (2018). Improving probability learning based local search for graph coloring. *Applied Soft Computing*, 65:542–553. https://doi.org/10.1016/j.asoc.2018.01.027.

Zhou, Y. M., Hao, J. K., and Duval, B. (2016). Reinforcement learning based local search for grouping problems: A case study on graph coloring. *Expert Systems with Applications*, 64:412–422. https://doi.org/10.1016/j.eswa.2016.07.047.