Opposition-based learning memetic algorithm for the maximum intersection of k-subsets problem

Wen Sun [a,b], Xu Li [a], Jin-Kao Hao [c,*], Wenlong Li [a], Zhipeng Lü [d]

[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] *SMART, School of Computer Science and Technology, Huazhong University of Science and Technology, 430074 Wuhan, China*

Engineering Applications of Artificial Intelligence 164: 113271, 2026
https://doi.org/10.1016/j.engappai.2025.113271

Abstract

Given m elements and n subsets of elements, the maximum intersection of k-subsets (kMIS) problem is to select k subsets of elements to maximize the number of elements simultaneously covered by all of the selected subsets. As a general model, kMIS can be used to formulate some practical problems including data privacy control, community detection, and deoxyribonucleic acid microarray technology. This paper presents an opposition-based learning memetic algorithm that integrates opposition-based learning initialization, adaptive crossover, and solution-based tabu search. Experimental results on 608 instances show that the algorithm competes favorably with the state-of-the-art methods. The importance of the algorithmic components is experimentally validated.

Keywords: Opposition-based learning; Solution-based tabu search; Maximum intersection.

* Corresponding author.
Email addresses: `101012533@seu.edu.cn` (Wen Sun), `220224957@seu.edu.cn` (Xu Li), `hao@info.univ-angers.fr` (Jin-Kao Hao), `220215468@seu.edu.cn` (Wenlong Li), `zhipeng.lv@hust.edu.cn` (Zhipeng Lü).

1 Introduction

Given a set $E = \{e_1, e_2, ..., e_m\}$ of m elements, a collection $S = \{s_1, s_2, ..., s_n\}$ of n non-empty subsets of E, and an integer number k, the maximum intersection of k-subsets (kMIS) problem [1] is to select a collection X of k subsets from S to maximize the number of common elements from E. The kMIS problem can be expressed formally as follows [1].

$$(kMIS) \qquad \text{Maximize} \quad f_0(X) = |\bigcap_{s_i \in X} s_i| \tag{1}$$

$$\text{subject to} \quad |X| = k \tag{2}$$

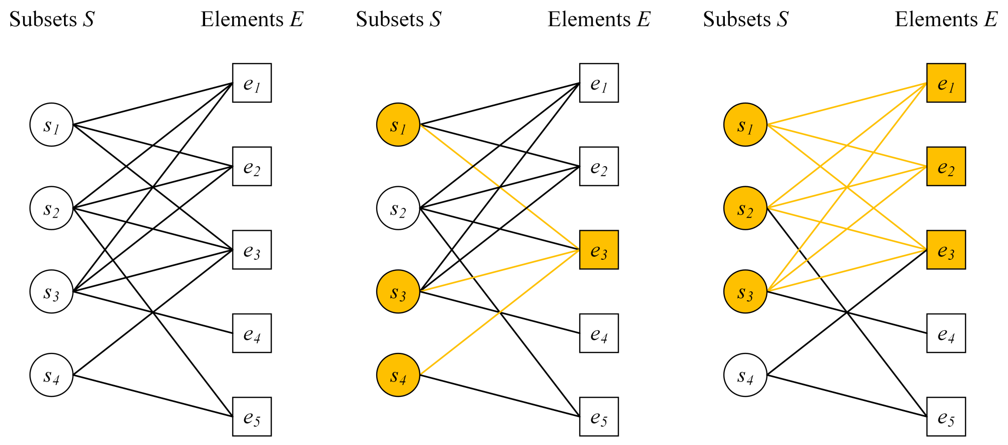
$$X \subseteq S \tag{3}$$

Equation (1) (objective function) aims to maximize the number of common elements covered by all of the selected subsets in X. Constraint (2) ensures that exactly k subsets are selected in solution X. Constraint (3) requires that the selected subsets must be from S.

The kMIS problem can also be modeled as a bipartite graph $G = (S \cup E, A)$, where E is the set of m elements, S is the collection of n non-empty subsets of E, A is the set of edge $\{s_i, e_j\}$ in which $e_j \in s_i$ ($i \in \{1, ..., n\}, j \in \{1, ..., m\}$). The density of G is the number of $|A|$ divided the number of $|S| \times |E|$ [2]. Figure 1(a) exemplifies a kMIS instance with $E = \{e_1, e_2, ..., e_5\}$, $S = \{s_1, s_2, ..., s_4\}$, $A = \{\{s_1, e_1\}, \{s_1, e_2\}, ..., \{s_4, e_5\}\}$, $k = 3$, and $density = |A|/(|S| \times |E|) = 0.65$. Figure 1(b) depicts a candidate solution $X = \{s_1, s_3, s_4\}$ with an objective value $f_0(X) = |s_1 \cap s_3 \cap s_4| = |\{e_3\}| = 1$, and highlights the selected subsets, their common elements, and the edges between the selected subsets and common elements in yellow. Figure 1(c) shows an optimal solution $X = \{s_1, s_2, s_3\}$ with the maximum objective function $f_0(X)$ of 3.

KMIS was proven to be NP-hard and is known to be difficult to approximate [3]. As a general model, kMIS can also be used to formulate some practical problems related to data privacy control [1], community detection [4], and DNA microarray technology [5]. For instance, a typical application of kMIS is in recommender systems for musician group formation [6], where each musical artist is related to several fans. Then the goal is to find k groups of musical artists maximizing the number of fans to match a musical event or playlist. By considering that E is the set of fans and S is composed of different groups of musical artists, this practical problem is conveniently modeled by kMIS.

In this paper, we present the first opposition-based learning memetic algorithm to solve kMIS with following contributions. First, the proposed algorithm uses an opposition-based learning (OBL) population initialization by



(a) An example of kMIS $G = (S \cup E, A)$ (b) A candidate solution $X = \{s_1, s_3, s_4\}$ (c) An optimal solution $X = \{s_1, s_2, s_3\}$

Fig. 1. An example of kMIS with a candidate solution and an optimal solution (the selected subsets, their common elements, and the edges between the selected subsets and common elements are in yellow)

simultaneously considering a candidate solution and its corresponding opposite solution. Second, it integrates an adaptive procedure for dynamically selecting a suitable crossover between two candidate crossovers. Third, it uses a solution-based tabu search with a multi-swap neighborhood to effectively examine the search space, a hash-based approach to effectively decide the tabu status of neighbor solutions and a fast neighborhood evaluation to shorten the computation time.

We present 26 record-breaking best solutions on the benchmark instances, which can be useful for future research on the problem.

The rest of the paper is structured as follows. Section 2 introduces an equivalent problem of kMIS and reviews the literature for these two problems. Section 3 illustrates the proposed algorithm. Section 4 presents the performance comparison. Section 5 justifies the significance of the main ingredients of the algorithm. In the last section summarizes this study and discusses the future research prospects.

2 Preliminaries

This section introduces the minimum k-union problem (MkUP) which is equivalent to kMIS and then provide a literature review.

2.1 Equivalent problem of kMIS

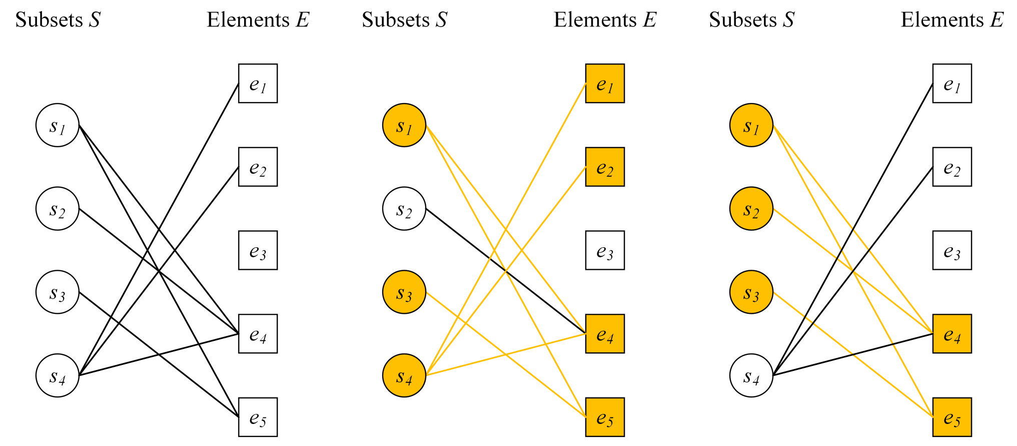
According to [1], kMIS and the minimum k-union problem (MkUP) are two "equivalent" problems. The MkUP is to select a collection X of k subsets from S to minimize the union of elements uncovered by each selected subset. The MkUP can be presented following [1].

$$(MkUP) \qquad \text{Minimize} \quad f(X) = |\bigcup_{s_i \in X} (E \setminus s_i)| \tag{4}$$

$$\text{subject to} \quad |X| = k \tag{5}$$

$$X \subseteq S \tag{6}$$

Equation (4) (objective function) is the minimization of the union of elements uncovered by all of the selected subsets in X. As $f_0(X) + f(X) = |\bigcap_{s_i \in X} s_i| + |\bigcup_{s_i \in X}(E \setminus s_i)| = m$, in which m is the sum amount of elements and fixed, minimizing f equals to maximizing f_0 (Equation (1)). Constraint (5) ensures that there are k subsets in solution X. Constraints (6) indicate that the selected subsets must be from the n candidate subsets of S.



(a) An example of MkUP $\bar{G} = (S \cup E, \bar{A})$ (b) A candidate solution $X = \{s_1, s_3, s_4\}$ (c) An optimal solution $X = \{s_1, s_2, s_3\}$

Fig. 2. An example of MkUP with a candidate solution and an optimal solution (the selected subsets, uncovered elements of the selected subsets, and the edges between the selected subsets and uncovered elements are in yellow)

The MkUP problem is modeled as a complement graph $\bar{G} = (S \cup E, \bar{A})$ to G, where E is the set of m elements and S is the collection of n non-empty subsets of E like for kMIS, and $\bar{A}$ is the set of edges $\{s_i, e_j\}$, where $e_j \notin s_i$ $(i \in \{1, ..., n\}, j \in \{1, ..., m\})$. Figure 2(a) is the corresponding complement graph $\bar{G} = (S \cup E, \bar{A})$ of G in Figure 1(a). Differently, $\bar{A} = \{\{s_1, e_4\}, ..., \{s_4, e_4\}\}$.Figure 2(b) shows a candidate solution $X = \{s_1, s_3, s_4\}$ with an objective value of 4, i.e., $f(X) = |(E \setminus s_1) \cup (E \setminus s_3) \cup (E \setminus s_4)| =$

$|\{e_1, e_2, e_4, e_5\}| = 4$, where the selected subsets, the uncovered elements, and the edges between the selected subsets and uncovered elements are highlighted in yellow. Figure 2(c) exemplifies an optimal solution $X = \{s_1, s_2, s_3\}$ with the minimum objective value $f(X)$ of 2.

As shown in [1], any solution to MkUP is also the solution to kMIS. However, the MkUP is more appealing than kMIS since we can evaluate the neighboring solutions using remove move operator, which is obtained by removing a subset from current solution, in constant time $O(1)$, while the neighborhood evaluation requires $O(k \times n)$ for kMIS with the same move operator.

2.2 Literature review of kMIS and MkUP

We classify currently available algorithms for kMIS into exact and heuristic algorithms.

Exact algorithms are known to guarantee the optimality of their obtained solutions. Nussbaum et al. [5] stated that kMIS is solvable in polynomial time for all convex bipartite graphs. Bogue et al. [6] presented two integer linear programming models (M_{CLIQUE} and M_{EDGE}) and generated 360 real-world instances ($50 \leq n \leq 150$ and $2 \leq k \leq 7$). These 360 real-world instances, which are also referred to as Lastfm instances in this paper, originated from a music recommendation dataset [1] . Computational results showed that when processed by CPLEX solvers, the M_{CLIQUE} model was more efficient than the model M_{EDGE}, and optimally solved all the instances with $n \leq 150$ and $k \leq 5$.

Because of their high complexity, exact algorithms are unpractical for large instances with $k > 5$. Specifically, as shown in [6], within the given time limit of 1800 seconds exact algorithms can only solve up to 96.6% of the instances with $k = 6$ and 70% of the instances with $k = 7$. Therefore, heuristic algorithms are adopted to obtain suboptimal solutions for large instances in acceptable computation time.

Bogue et al. [6] introduced a reactive greedy randomized adaptive search procedure (Reactive-GRASP). The proposed algorithm integrates a greedy randomized initialization phase, a local search phase with the swap operator, and a path-relinking phase. Their results showed that for the 360 real-world instances, their algorithm could obtain 94% optimal solutions in less than 7 seconds in average, which was much faster than the time required by CPLEX solvers to process the M_{CLIQUE} and M_{EDGE}.

Dias et al. [7] introduced a reactive variable neighborhood search (Reactive-

[1] https://www.last.fm/

VNS) that adopts an initial constructive phase and iterates between a local search phase and a shaking phase. The constructive phase adopts greedy remove and a path-relinking procedure to create an initial solution. The local search phase randomly removes $0.3 \times k$ subsets from the current solution and greedily inserts the same number of subsets to the solution. The shaking phase randomly removes some subsets, whose number is adaptively chosen from $[1, k]$, and then inserts the same number of subsets to the solution through a greedy randomized strategy. The authors generated 238 new random instances ($32 \leq n \leq 300$ and $7 \leq k \leq 266$). Experimental results on these random instances showed that compared with Reactive-GRASP, Reactive-VNS obtained 30 better, 196 equal and 12 worse results.

Casado et al. [8] presented a greedy randomized adaptive search procedure (GRASP) that incorporates a greedy randomized initialization phase and a tabu search phase. The initialization phase iteratively inserts the best subset from half of random unselected subsets to the current solution. Then, the tabu search phase deploys a swap operator and the first improvement approach to improve each solution generated by the constructive phase. Experiments showed that compared with Reactive-VNS, GRASP obtained 20 better, 208 equal and 10 worse results on the 238 random instances proposed in [7].

Therefore, Reactive-GRASP, Reactive-VNS and GRASP are the most advanced existing heuristic algorithms for tackling kMIS. They are used as comparative algorithms for this study.

For MkUP, the equivalent problem of kMIS, there are only three approximation algorithms with theoretically guaranteed performance, but without showing practical results. Chlamtác et al. [9] presented an approximation algorithm with a guaranteed approximation ratio of $O(\sqrt{n})$, in which n is the sum amount of subsets. Similarly, Yonatan [10] proposed an $O(\sqrt{n})$-approximation. Later, Chlamtác et al. [11] achieved an algorithm with an approximation guarantee of $O(n^{1/4})$ and proved that for some dense and random graphs there is no polynomial-time algorithm that can approximate MkUP better than $O(n^{1/4})$.

Table 1 sums up the algorithms studied in this section. Column 1 indicates the authors of the literature. Column 2 shows the algorithm name of the corresponding literature. Column 3 gives the move operators used by an algorithm when it is applicable, where "-" indicates that no search operator is used. Column 4 indicates the problems solved by the corresponding algorithms. Column 5 gives the category of instances on which an algorithm has been tested.

Table 1
Representative exact, heuristic and approximation algorithms for kMIS and MkUP.

Literature	Algorithm name	Operators	Problem solved	Instance
Exact algorithms				
Bogue et al. [6]	M_{CLIQUE} and M_{EDGE}	-	kMIS	360 real-world
Heuristic algorithms				
Bogue et al. [6]	Reactive-GRASP	swap	kMIS	360 real-world
Dias et al. [7]	Reactive-VNS	remove, insert	kMIS	238 random
Casado et al. [8]	GRASP	swap	kMIS	238 random
Approximation algorithms				
Chlamtác et al. [9]	$O(\sqrt{n})$-approximation	-	MkUP	-
Yonatan [10]	$O(\sqrt{n})$-approximation	-	MkUP	-
Chlamtác et al. [11]	$O(n^{1/4})$-approximation	-	MkUP	-

2.3 Literature review of OBL

OBL was originally proposed as a machine intelligence scheme and applied in reinforcement learning [12]. The main idea of OBL is to consider a candidate solution and its corresponding opposite solution simultaneously. According to [13] and [14], OBL is a rapidly growing research field in which various theoretical models and technical methods have been studied to solve complex optimization problems.

Zhou et al. [15] proposed an opposition-based memetic search to solve the maximum diversity problem, which uses opposite solutions during population initialization and local optimization. Ventresca et al. [16] proposed a population-based incremental learning algorithm for the traveling salesman problem, which uses opposite solutions in the population update procedure to improve the population diversification. Zhang et al. [17] proposed an opposition-based ant colony optimization algorithm for the traveling salesman problem, in which the opposite solution of the local optimal solution is used to update the pheromone in each iteration. Rahnamayan et al. [18] proposed an opposition-based differential evolution algorithm for solving constrained minimization problems, which constructs opposite solutions in the population initialization procedure. Lei et al. [19] proposed a quasi-opposition-based imperialist competition algorithm for solving the I-beam design problem, the speed reducer design problem, and the car side impact design problem, where opposite solutions are used for both population initialization and population update. Yildiz et al. [20] presented a dynamic opposition-based flow direction optimization method for solving several problems, in which opposite solutions are used to update the population. Mehta et al. [21] introduced the generalized normal distribution algorithm based on elite opposition-based learning for solving mechanical engineering design problems, where the opposite solution generated from the best elite solution is used to update the population.

In summary, the OBL method is widely applied to routing problems and engineering design problems. In this paper, we present an OBL method for 0-1 optimization problems, which can be adapted to related binary problems. On the other hand, existing OBL algorithms usually randomly generate opposite solutions in population initialization or update procedures, while our OBL strategy designs a greedy randomized approach to generate opposite solutions.

2.4 Research gaps and objectives

According to the reported results, there are few practical algorithms that can effectively solve kMIS and MkUP. The exact algorithms in [6] could not obtain optimal solutions on large instances due to the inevitable exponential time complexity. The heuristic algorithms in [7,8] lacked stability on large instances with $n \geq 160$ and $k \geq 91$.

Furthermore, the recent heuristic algorithms for kMIS are based on either GRASP or VNS, while the powerful population-based memetic framework has not been studied for solving the problem. In fact, memetic algorithms are known to have excellent performance for several related problems [22,23]. In addition, recent research on improving search algorithms through opposition-based learning memetic techniques has achieved top performances for several difficult binary optimization problems [15,24]. Finally, the local search procedures of existing algorithms are based on the classic swap operator, while ignoring other move operators. Therefore, we strive here to advance the state of the art for this problem by designing a new hybrid algorithm combining the opposition-based learning memetic algorithm framework using two adaptively applied crossover operators and an effective local search using an original multi-swap operator and its fast evaluation technique.

3 Opposition-based learning memetic algorithm

Algorithm 1 illustrates the opposition-based learning memetic algorithm (OBLMA) for solving MkUP. OBLMA begins with a collection of diversified elite solutions that are achieved by the OBL initialization procedure (line 1). During each generation, the adaptive crossover procedure (line 5) selects, according to the probability learning technique, a crossover operator to generate an offspring solution from two parent solutions. Subsequently, the solution-based tabu search procedure (line 6) is activated to improve the offspring. Finally, the diversity-based pool updating strategy (line 10) is used to update the population with the improved offspring. This process continues until the stopping condition (e.g., a time limit) is satisfied (lines 4-11).

Algorithm 1 The main framework of the OBLMA algorithm for solving MkUP

```
Input: Instance $\bar{G} = (S \cup E, \bar{A})$, population size $p$, greediness ratio $\epsilon$, depth of solution-based tabu search $\omega_1$, $\omega_2$ weighting coefficient $\beta$, hash vectors $H_k$ with a length of $L$ $(k = 1, 2, 3)$; hash functions $h_k$ $(k = 1, 2, 3)$
Output: The recorded best solution $X^*$
 1: $P = \{X^1, X^2, ..., X^p\} \leftarrow OBL\_Initialization(\bar{G}, p, \epsilon, \omega_1, H_k, h_k)$
                                                        /*Construct initial population*/
 2: $X^* \leftarrow argmin\{f(X^i)|i = 1, 2, ..., p\}$        /*Initialize the best solution $X^*$*/
 3: Initialize the crossover probability $\gamma$
 4: while Stopping condition is not satisfied do
 5:     $X \leftarrow Adaptive\_Crossover(\bar{G}, P, \gamma)$              /*Generate offspring $X$*/
 6:     $X \leftarrow Solution\_Based\_Tabu\_Search(\bar{G}, X, \omega_2, H_k, h_k)$ /*Improve offspring $X$*/
 7:     if $f(X) < f(X^*)$ then
 8:         $X^* \leftarrow X$                       /*Update the recorded best solution $X^*$*/
 9:     end if
10:     $(P, \gamma) \leftarrow DiversityBased\_Pool\_Updating(P, X, \beta, \gamma)$
                         /*Update the population $P$ and the crossover probability $\gamma$*/
11: end while
12: return $X^*$
```

3.1 OBL population initialization

The OBL population initialization constructs a population P with p diverse and high quality solutions. Different from the previous initialization in local search[6–8], our OBL initialization procedure design a novel greedy construction strategy and integrate an OBL strategy. The OBL strategy was firstly proposed as a machine intelligence scheme for reinforcement learning [12].

In detail, the OBL initialization procedure (Algorithm 2) firstly constructs a pair of solutions, that is, a greedy solution X and its opposite solution $\hat{X}$. These two solutions are then improved by the solution-based tabu search procedure described in Section 3.3. Finally, the better one of the two improved solutions X^i is inserted into the population P (lines 6-7). This procedure repeats until the population contains p distinct solutions (lines 2-8).

Our novel greedy construction strategy build a greedy solution X by iteratively inserting subsets from S. Specifically, each iteration inserts the subset $s_i \in S \setminus X$ with the minimum incremental objective value into X with a probability of ϵ; otherwise, inserts a random subset $s_i \in S \setminus X$ into X with a probability of $1 - \epsilon$. The construction procedure of X stops when there are k subsets in X.

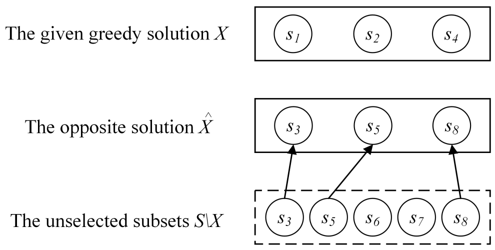
There are two cases for the construction of the opposite solution $\hat{X}$: (1) if $k < \frac{n}{2}$, $\hat{X}$ is constructed by iteratively inserting one subset $s_i \in S \setminus X$ into

Algorithm 2 OBL population initialization

```
Input: Instance $\bar{G} = (S \cup E, \bar{A})$, population size $p$, greediness ratio $\epsilon$, depth of
    solution-based tabu search $\omega_1$, hash vectors $H_k$ with a length of $L$ $(k = 1, 2, 3)$;
    hash functions $h_k$ $(k = 1, 2, 3)$
Output: Initial population $P$
1: $P \leftarrow \emptyset$
2: for $i \leftarrow 1$ to $p$ do
3:     Construct a candidate solution $X$ and its opposite solution $\hat{X}$ by greedy
       randomized construction procedure
4:     $X \leftarrow Solution\_Based\_Tabu\_Search(\bar{G}, X, \omega_1, H_k, h_k)$
5:     $\hat{X} \leftarrow Solution\_Based\_Tabu\_Search(\bar{G}, \hat{X}, \omega_1, H_k, h_k)$
6:     $X^i \leftarrow argmin\{f(X), f(\hat{X})\}$          /* Identify the better solution $X^i$ */
7:     $P \leftarrow P \cup \{X^i\}$                  /* Add $X^i$ into the population $P$ */
8: end for
9: return $P = \{X^1, X^2, ..., X^p\}$
```

$\hat{X}$ following the greedy randomized construction strategy until there are k subsets in $\hat{X}$; (2) if $k \geq \frac{n}{2}$, all subsets of $S \setminus X$ are firstly inserted into $\hat{X}$, and then subsets from X are iteratively inserted into $\hat{X}$ following the greedy randomized construction strategy until there are k subsets in $\hat{X}$.

Figure 3 illustrates these two cases on an instance with $S = \{s_1, s_2, ..., s_8\}$: 1) $k < \frac{n}{2}$, suppose $k = 3$ as shown in Figure 3(a); 2) $k \geq \frac{n}{2}$, suppose $k = 5$ as shown in Figure 3(b). In Figure 3(a), suppose that the greedy solution $X = \{s_1, s_2, s_4\}$ is given, the opposite solution $\hat{X}$ is generated by iteratively inserting three subsets of $S \setminus X$, for example $\{s_3, s_5, s_8\}$, according to the greedy randomized construction strategy. In Figure 3(b), suppose that the greedy solution $X = \{s_1, s_3, s_4, s_5, s_8\}$ is given, the opposite solution $\hat{X}$ is constructed by firstly inserting all subsets of $S \setminus X$, i.e., $\{s_2, s_6, s_7\}$, and then iteratively inserting subsets of X (in the example $\{s_1, s_4\}$) according to the greedy randomized construction strategy until the size of $\hat{X}$ equals k.


The given greedy solution X
s_1
s_2
s_4
The opposite solution $\hat{X}$
s_3
s_5
s_8
The unselected subsets $S\setminus X$
s_3
s_5
s_6
s_7
s_8
(a) The first case with $k < \frac{n}{2}$, $k = 3$

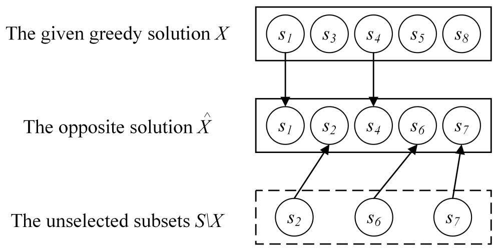

The given greedy solution X
s_1
s_3
s_4
s_5
s_8
The opposite solution $\hat{X}$
s_1
s_2
s_4
s_6
s_7
The unselected subsets $S\setminus X$
s_2
s_6
s_7
(b) The second case with $k \geq \frac{n}{2}$, $k = 5$

Fig. 3. The two cases of construction of the opposite solutions

3.2 Crossovers and applications

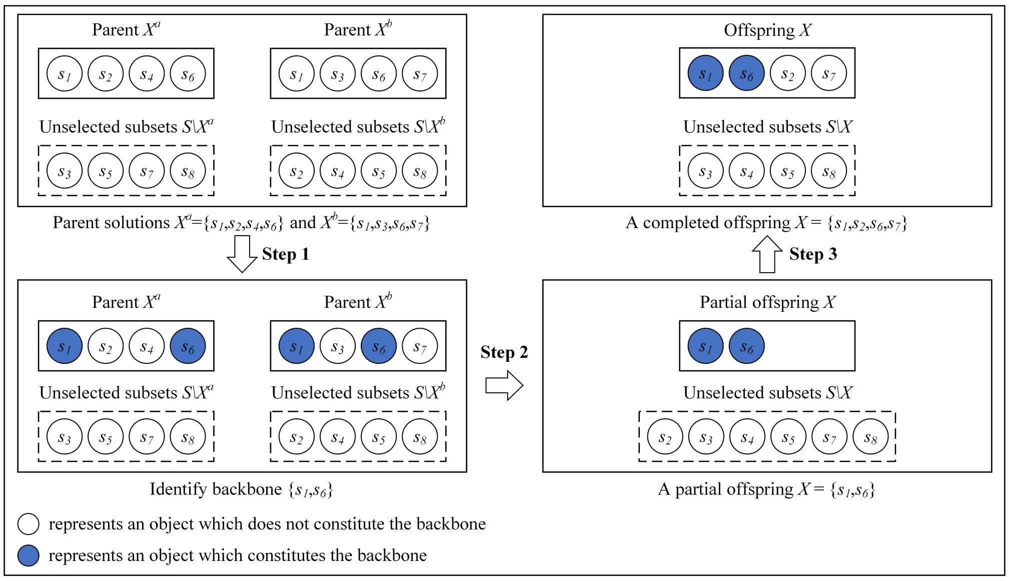
Crossover operators are crucial for population-based memetic approaches. They have to transmit meaningful characteristics from parents to offspring and guarantee offspring diversity at the same time [25]. Parents can pass on their different characteristics by means of different crossovers to maintain offspring performance and diversity. Moreover, particular characteristics may be needed by different problem instances, or one instance at different search stages. Consequently, there is no single crossover operator that can fit all instances or all search stages. Thus, adaptively applying an ensemble of crossover operators during the search is an interesting approach, as shown in [26–28].

3.2.1 Crossovers operators

The backbone approach has led to some successful crossovers for subset selection problems [29,15]. We employ the backbone idea in designing our crossovers for MkUP since MkUP is also a typical subset selection problem. In our case, both the tradition backbone-based crossover and a new OBL backbone crossover are used to generate diverse offspring, which are selected according to a probabilistic learning technique.

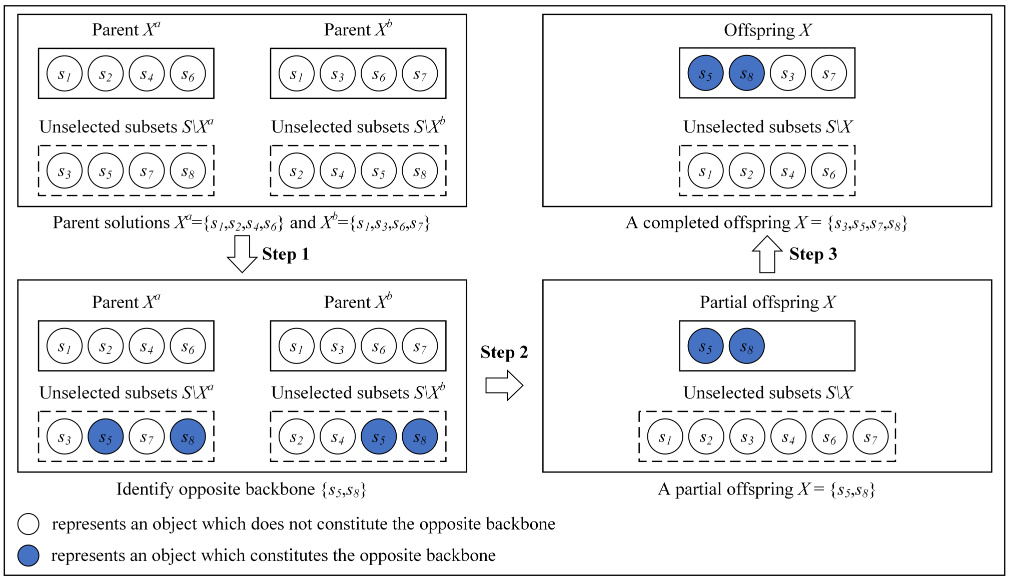
Backbone-based crossover [25,29]: This operator generates an offspring in three steps. First, we identify the backbone between two parents X^a and X^b, i.e., $X = X^a \cap X^b$. Second, the partial offspring is formed by inheriting the backbone. Finally, the partial offspring X is successively repaired by using a greedy approach, which alternatively inserts a subset $s_i \in S \setminus X$ with the minimum objective value $f(X \cup \{s_i\})$ until the size of X reaches k. Figure 4 exemplifies how the backbone-based crossover operator works. The example takes an instance of 8 subsets, and operates with two parent solutions X^a and X^b. Firstly, the backbone $\{s_1, s_6\}$ between X^a and X^b is identified. Secondly, the partial offspring X is generated by inheriting the backbone subsets. Finally, the partial solution is completed by alternatively inserting subsets with the minimum incremental objective value from $S \setminus X$ (subsets s_2 and s_7).

OBL backbone crossover: Similar to backbone-based crossover, the OBL backbone crossover also constructs its offspring in three steps. Firstly, the opposite backbone is identified, which consists of the subsets not in any parent, i.e., $S \setminus \{X^a \cup X^b\}$. Second, a partial solution X is obtained by inheriting the opposite backbone. Finally, if $k < \frac{n}{2}$, the subset $s_i \in S \setminus X$ with the minimum objective value $f(X \cup s_i)$ is iteratively inserted into the partial offspring X until the size of X equals to k; otherwise, subsets $s_i \in X$ with the minimum objective value $f(X \setminus \{s_i\})$ are iteratively removed from X until the size of X equals to k. Figure 5 illustrates the OBL backbone crossover. Firstly, the opposite backbone $\{s_5, s_8\}$ between X^a and X^b is identified. Secondly, the partial offspring X is built by conserving the opposite backbone subsets. Finally, the partial solution is completed by alternatively inserting subsets with the minimum incremental objective value from $S \setminus X$ (subsets s_3 and s_7).



Parent X^a
Parent X^b
Unselected subsets $S\setminus X^a$
Unselected subsets $S\setminus X^b$
Parent solutions $X^a=\{s_1,s_2,s_4,s_6\}$ and $X^b=\{s_1,s_3,s_6,s_7\}$
Step 1
Identify backbone $\{s_1,s_6\}$
Step 2
Partial offspring X
Unselected subsets $S\setminus X$
A partial offspring $X=\{s_1,s_6\}$
Step 3
Offspring X
A completed offspring $X=\{s_1,s_2,s_6,s_7\}$
represents an object which does not constitute the backbone
represents an object which constitutes the backbone

Fig. 4. Example of the backbone-based crossover with parents $X^a = \{s_1, s_2, s_4, s_6\}$, $X^b = \{s_1, s_3, s_6, s_7\}$, and $k = 4$



Parent X^a
Parent X^b
Unselected subsets $S\setminus X^a$
Unselected subsets $S\setminus X^b$
Parent solutions $X^a=\{s_1,s_2,s_4,s_6\}$ and $X^b=\{s_1,s_3,s_6,s_7\}$
Step 1
Identify opposite backbone $\{s_5,s_8\}$
Step 2
Partial offspring X
Unselected subsets $S\setminus X$
A partial offspring $X=\{s_5,s_8\}$
Step 3
Offspring X
A completed offspring $X=\{s_3,s_5,s_7,s_8\}$
represents an object which does not constitute the opposite backbone
represents an object which constitutes the opposite backbone

Fig. 5. Example of the OBL backbone crossover with parents $X^a = \{s_1, s_2, s_4, s_6\}$, $X^b = \{s_1, s_3, s_6, s_7\}$, and $k = 4$

3.2.2 Learning-based adaptive crossover

Each generation of the algorithm chooses one type of crossovers with a learning-based strategy [30]. Specifically, we generate *rand* which is a random number between 0 and 1. If *rand* less than the value of learning-based crossover probability γ, the backbone-based crossover is chosen (lines 2-3); otherwise, the OBL backbone crossover is chosen (lines 4-5).

Algorithm 3 Learning-based adaptive crossover

```
Input: Instance $\bar{G} = (S \cup E, \bar{A})$, the population $P$, the learning-based crossover probability $\gamma$
Output: Offspring $X$
1: Randomly select two parent solutions $X^a$ and $X^b$ from $P$
2: if $rand(0,1) < \gamma$ then
3:     $X \leftarrow BackboneBased\_Crossover(X^a, X^b)$
4: else
5:     $X \leftarrow OBL\_BackboneCrossover(X^a, X^b)$
6: end if
7: return $X$
```

The learning-based updating of γ is described in the Section 3.4. Specifically, the value of γ will adaptively rise if the backbone-based crossover operator is chosen and the resulting offspring is incorporated into the population. Otherwise, γ will adaptively reduce. It is expected to select more suitable crossover at each generation of the algorithm by using the adaptively learned probabilities to create promising offspring solutions.

3.3 Solution-based tabu search

Solution-based tabu search (SBTS) has recently shown great performances on challenging optimization problems including the multidemand multidimensional knapsack problem [31], the obnoxious p-median problem [32], and the set-union knapsack problem [33]. Compared to attribute-based tabu search, SBTS can eliminate the difficulty of tuning the tabu tenure and ensuring stronger search intensification.

Algorithm 4 shows the framework of our proposed SBTS procedure. Beginning with the initial solution X (as indicated at line 1), SBTS first sets up the hash vectors (as shown at line 4) It then proceeds into a loop that runs until no improved solution is found for a given number of iterations (encompassed by lines 6-16). In each loop iteration, SBTS identifies the best admissible neighboring solution X' within the multi-swap neighborhood (explained below) and adopts it as the new current solution (lines 7-8). It subsequently updates the hash vectors (line 9), potentially updates the best recorded solution X_{best}, and adjusts the counter for the number of consecutive non-improvement (ranging from lines 10-15). The search stops if there is no update to the best solution found for ω_2 consecutive iterations, where ω_2 represents the search depth of SBTS.

Algorithm 4 Solution-based tabu search

```
Input: Instance $\bar{G} = (S \cup E, \bar{A})$, input solution $X$, search depth $\omega_2$, hash vectors
    $H_r$ of length $L$, $r = 1, 2, 3$; hash functions $h_r$, $r = 1, 2, 3$
Output: The best solution $X_{best}$ found during the search
 1: $X_{best} \leftarrow X$    /* Initialize the best solution $X_{best}$ of solution-based tabu search */
 2: $non\_improve \leftarrow 0$    /* Indicate the continuous iterations where $X_{best}$ is not updated */
 3: for $i \leftarrow 0$ to $L - 1$ do
 4:     $H_1[i] \leftarrow 0, H_2[i] \leftarrow 0, H_3[i] \leftarrow 0$    /* Initialize the hash vectors */
 5: end for
 6: while $non\_improve \leq \omega_2$ do
 7:     Find a best neighbor solution $X'$ that is not simultaneously forbidden by the
        hash vectors (i.e., tabu lists) $H_1$, $H_2$, and $H_3$ from the current neighborhood
        $N(X)$
 8:     $X \leftarrow X'$
 9:     $H_1[h_1(X)] \leftarrow 1, H_2[h_2(X)] \leftarrow 1, H_3[h_3(X)] \leftarrow 1$    /* Update the hash vectors
        with $X$ */
10:     if $f(X) < f(X_{best})$ then
11:         $X_{best} \leftarrow X$
12:         $non\_improve \leftarrow 0$
13:     else
14:         $non\_improve \leftarrow non\_improve + 1$
15:     end if
16: end while
17: return $X_{best}$
```

Multi-Swap Operator: For local optimization, [6] and [8] adopt the traditional swap operator to examine the search space. The swap operator combines two consecutive moves, i.e., removing a selected subset from the solution and inserting another subset into the solution. However, according to the experimental results, it may happen that the remove phase does not decrease the objective value since the elements which are not covered by a removed subset are also not covered by other subsets in local optimal solutions. To overcome this obstacle, [7] tries to randomly remove a fixed number of subsets and then greedily insert the same number of subsets into the solution.

In this study, we present a new multi-swap operator that always considers the objective variation and determines the suitable number of swapped subsets according to the current solution. Specifically, the multi-swap operator consists two phases. The first phase iteratively removes subsets from X with the minimum incremental objective value (Equation (8)) until the current solution is improved. Then, subsets with the minimum incremental objective value are iteratively inserted to X (ties are broken randomly) until the number of the subsets in X equals k.

Neighborhood structure: The neighborhood N contains all possible solutions generated by adopting "multi-swap" to X.

$$N(X) = \{X \oplus multi\text{-}swap(R, I) : R \subset S, I \subset S, R \cap I = \emptyset, |R| = |I|\} \quad (7)$$

where R be the set of subsets to be removed, I the set of subsets to be inserted, and $multi\text{-}swap(R, I)$ designates the operation that removes the subsets of R from X and inserts the subsets of I to X. The size of N is bounded by $O(k \times n)$.

Fast Neighborhood Evaluation Technique: The multi-swap operator is a combination of several remove operations and several insert operations. The incremental objective value (also called move gain, denoted by Δf) after applying $multi\text{-}swap(R, I)$ can be calculated by the sum of incremental evaluation values of removing all subsets in R and incremental evaluation values of inserting all subsets in I. To efficiently compute the incremental evaluation value (Δf) of removing a subset or inserting a subset, we maintain a vector δ of size n in which $\delta[i]$ ($1 \leq i \leq n$) records Δf of removing or inserting s_i. The vector δ is initialized as below:

$$\delta[i] = \begin{cases} |\{e_j \in s_i : |U_j \cap X| = 0\}|, & s_i \notin X \\ -|\{e_j \in s_i : |U_j \cap X| = 1\}|, & s_i \in X \end{cases} \quad (8)$$

where e_j is an element of the subset s_i, and U_j is the set of the subsets not covering e_j.

Once a subset is removed or inserted, δ is updated as follows.

1) For the removed subset s_i (or the inserted subset s_i), $\delta[i] = -\delta[i]$.

2) If subset s_i is removed from X, the value $\delta[d]$ of each other subset s_d ($d \neq i$) is updated by,

$$\delta[d] = \begin{cases} \delta[d] + 1, \ if \ |X \cap U_j| = 1 \ and \ s_d \in U_j \\ \delta[d] - 1, \ if \ |X \cap U_j| = 2 \ and \ s_d \in X \cap U_j \\ \delta[d], \quad otherwise \end{cases} \quad (9)$$

where e_j is an element that does not belong to the subset s_i, and U_j is the set of the subsets not covering e_j.

3) If subset s_i is inserted to X, the value $\delta[d]$ of each other subset s_d ($d \neq i$)

is updated by,

$$\delta[d] = \begin{cases} \delta[d] - 1, if \ |X \cap U_j| = 0 \ and \ s_d \in U_j \\ \delta[d] + 1, if \ |X \cap U_j| = 1 \ and \ s_d \in X \cap U_j \\ \delta[d], \qquad otherwise \end{cases} \tag{10}$$

where e_j is an element that does not belong to the subset s_i, and U_j is the set of the subsets not covering e_j.

The worst time complexity of updating δ is $O(mn)$.

Tabu list management strategy using hash functions: SBTS utilizes a tabu list mechanism to avoid revisiting any previously encountered solutions during the search process. Specifically, we use the solution-based tabu strategy which adopts three binary hash vectors H_1, H_2, and H_3 of length L, where each hash vector H_r is associated to a hash function h_r. The hash functions serve to assign a solution to an index within H_r $(r = 1, 2, 3)$.

Based on H_r and h_r $(r = 1, 2, 3)$, we define the following rule to decide the status of a candidate solution X. If $H_1(h_1(X)) \vee H_2(h_2(X)) \vee H_3(h_3(X)) = 1$, then X is classified as already seen and will not be considered further. Otherwise, X is classified as a non-tabu solution.

Let $X = (x_1, x_2, ..., x_n)$ be a candidate solution such that the $x_i = 1$ if subset $s_i \in S$ is selected, $x_i = 0$ otherwise. Then the hash values h_r $(r = 1, 2, 3)$ are given by:

$$h_r(X) = (\sum_{i=1}^{n} \lfloor i^{\xi_r} \rfloor \times x_i) \ mod \ L \tag{11}$$

where ξ_r $(r = 1, 2, 3)$ is a parameter and L is the length of hash vectors which is determined to be 10^8 based on empirical experimentation.

Given a solution X, its status (i.e., tabu or non-tabu) can be determined in $O(1)$ according to Equation (11).

Figure 6 illustrates the hash-based tabu list management strategy. According to Equation (11), applying the hash functions to the given solution X gives the hash values $h_1(X) = 1$, $h_2(X) = 5$ and $h_3(X) = 2$. Checking these values implies that X is a prohibited solution since $H_1(h_1(X)) \wedge H_2(h_2(X)) \wedge H_3(h_3(X)) = H_1[1] \wedge H_2[5] \wedge H_3[2] = 1$.

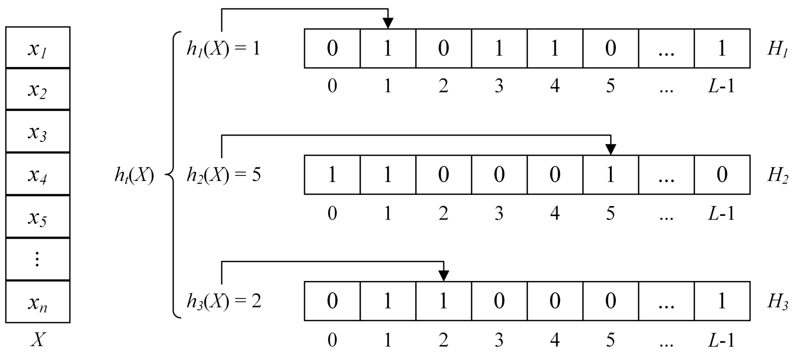


Fig. 6. An example of the hash-based tabu list management strategy

3.4 Diversity-based pool updating

We use a diversity-based population updating procedure (see Algorithm 5) to manage the insertion of each offspring solution X into the population P. Firstly, the offspring X is added to the population (line 2). Secondly, the goodness score of each solution in P is computed according to Equation (12) (line 3). Thirdly, the worst solution X_{worst} according to the goodness score is identified and removed from P (lines 4-5). Finally, the crossover probability γ is updated according to Equation (13) (line 7).

Algorithm 5 Diversity-based pool updating

```
Input: Population $P$, improved offspring $X$, weighting coefficient $\beta$, crossover probability $\gamma$
Output: Updated population $P$, updated crossover probability $\gamma$
1: /*Update the population $P$*/
2: $P \leftarrow P \cup \{X\}$
3: Calculate $Score(X^i)$ of each $X^i \in P$ according to Equation (12)
4: $X^{worst} \leftarrow argmin\{Score(X^i) | X^i \in P\}$
5: $P \leftarrow P \setminus \{X^{worst}\}$
6: /*Update the crossover probability $\gamma$*/
7: Update $\gamma$ by Equation (13)
8: return $(P, \gamma)$
```

Calculate goodness score $Score$**.** We calculate the goodness score $Score(X^i)$ of a solution X^i as below [34,35]:

$$Score(X^i) = \beta \times \frac{f_{max} - f(X^i)}{f_{max} - f_{min}} + (1 - \beta) \times \frac{D(X^i) - D_{min}}{D_{max} - D_{min}} \tag{12}$$

where f_{max} and f_{min} designate respectively the maximum and minimum objective value of the individuals in the population, $D(X^i) = Min\{distance(X^i, X^j) : X^j \in P, X^b \neq X^i\}$ is the distance between individual X^i and population, $distance(X^i, X^j)$ stands for the Hamming distance between X^i and X^j, D_{max}

and D_{min} are respectively the maximum and minimum Hamming distance between an individual and the population. The parameter β is tuned to 0.8 based on empirical test.

Update the crossover probability γ. When the population is updated with an offspring, the crossover probability γ is updated using a probability learning method (Equation (13)), inspired by [28,36].

$$\gamma = \frac{50 + q_1}{100 + q_1 + q_2} \tag{13}$$

where q_1 refers to the number of times the population is updated with an improved offspring from the backbone-based crossover and q_2 refers to the number of times the population is updated with improved offspring from the OBL backbone-based crossover. The increase of q_1 will equally increase both the numerator and denominator of Equation (13), which will increase the crossover probability γ and decrease the selection probability of the other (not applied) crossover.

3.5 Computational complexity

The complexity of the population initialization procedure is restricted by $O(k \times m \times n)$, where k is the given integer number of the selected subsets, m is the number of elements in E, and n is the number of subsets in S. The main loop of the OBLMA algorithm consists of three procedures: crossover, the solution-based tabu search, and pool updating procedure. The first requires time $O(k \times m \times n)$, the second needs $O(T \times k \times m \times n)$, and the last procedure can be accomplished in $O(p^2 \times n)$, where T is the number of iterations of tabu search, p is the population size. Therefore, one iteration of the main loop of the OBLMA algorithm requires $O(T \times k \times m \times n)$ time.

4 Computational assessment and comparative analysis

In this section, we present detailed experimental results and assessment of the proposed OBLMA algorithm on 360 real-world instances generated in this study, 238 random instances initially tested in [7] and 10 large real-world instances from the KONECT repository [37].

4.1 Benchmark instances

These instances are organized into three categories.

1. **Lastfm (360 real-world instances):** The instances of the first category come from a music recommendation dataset [2] , were firstly proposed in [6], but the source instances were not available. We generated the new instances named Lastfm** following the same method in [6] with different random seed. The optimal solutions of all new generated instances cannot be obtained by the CPLEX solver (as in Section 4.2). The k values are within $\{2, 3, 4, 5, 6, 7\}$ following [6]. Each instance has $m \in [965, 985]$ elements, $n \in [50, 150]$ subsets, and a density within $[0.355, 0.491]$.
2. **Random (238 random instances):** The second category introduced in [7] consists of 238 random instances generated following the graph models in [38], among which the optimal values of 180 instances are known. The names of these instances are in the form of classe**. These instances are featured by the number of elements m within $[32, 300]$, the number of subsets n within $[32, 300]$, the density within $[0.499, 0.991]$, and the value of k within $[7, 266]$.
3. **KONECT (10 large real-world instances):** The third category is composed of 10 instances from the KONECT repository [37], and their optimal solutions are unknown. The KONECT repository contains hundreds of networks from diversified real-life applications, such as rating networks, affiliation networks, and interaction networks. The selected instances are derived from movie rating networks, and with movielens** in their names. Each instance is characterized by the number of elements $m \in \{1682, 3706\}$, the number of subsets $n \in \{943, 6040\}$, the density within $[0.045, 0.063]$, and the value of k within $\{10, 20, 30, 40, 50\}$. For five instances, $m(n) = 1682(943)$, and for the other five instances, $m(n) = 3706(6040)$. Each k value corresponds to two different instances in this category, and m takes 1682 for five of them and 3706 for the others.

These instances can be accessed at https://github.com/sunseu2022/kMIS.

4.2 Experimental protocol

The OBLMA algorithm is coded in C++ [3] and compiled by the g++ compiler with the -O3 option.

[2] http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html
[3] The source code of the OBLMA algorithm will be publicly available at https://github.com/sunseu2022/kMIS with the publication of the paper.

Parameters: OBLMA requires 8 parameters, including the population size p, the greediness ratio ϵ, the search depth of OBL population initialization ω_1, the search depth of solution-based tabu search ω_2, the parameters ξ_1, ξ_2 and ξ_3 used in the hash function, and weighting coefficient β. The automatic parameter tuning package "irace" was employed to calibrate these parameters. For each parameter, a set of candidate values was first determined by preliminary experiments and presented in the fourth column of Table 2. For the tuning experiment, the maximum number of executions (tuning budget) used by Irace is set to 2,000 to choose one elite configuration as the final setting. We measured execution quality based on the number of the best objective values (f_{best}) achieved on 24 representative, difficult instances, for which the best objective values (f_{best}) cannot be consistently reached. The elite configuration returned by irace is shown in the last column (Final value) of Table 2. These final parameter values are used in all experiments reported in the following sections and can be considered to form the default parameter setting of our algorithm.

Table 2
Parameter settings of the OBLMA algorithm

Parameters	Section	Description	Candidate values	Final value
p	3.1	Population size	{5, 10, 15, 20, 25}	15
ϵ	3.1	Greediness ratio of initialization	{0.5, 0.6, 0.7, 0.8, 0.9}	0.7
ω_1	3.1	Search depth of OBL population initialization	{20, 40, 60, 80, 100}	40
ω_2	3.3	Search depth of local search	{100, 500, 1000, 1500, 2000}	1000
ξ_1	3.3	Parameter in the first hash function	{1.1, 1.2, 1.3, 1.4, 1.5}	1.2
ξ_2	3.3	Parameter in the second hash function	{1.3, 1.4, 1.5, 1.6, 1.7}	1.6
ξ_3	3.3	Parameter in the third hash function	{1.7, 1.8, 1.9, 2.0, 2.1}	2.0
β	3.4	Weighting coefficient of pool updating	{0.5, 0.6, 0.7, 0.8, 0.9}	0.8

Reference algorithms: Four state-of-the-art kMIS algorithms are taken as references to evaluate the OBLMA algorithm, including CPLEX solver (CPLEX), Reactive-GRASP [6], Reactive-VNS [7], and GRASP [8].

We run last three reference algorithms (named as Re-Reactive-GRASP, Re-Reactive-VNS and Re-GRASP) using the codes provided by [7] and [8], and followed the calibrated parameter settings provided in corresponding papers.

1) The CPLEX solver (CPLEX): We present an integer linear programming model (see Appendix A) and solve it based on the CPLEX solver (version 12.8) under the time limit of 1 hour for each instance.
2) Re-Reactive-GRASP [6]: The Reactive-GRASP algorithm was firstly proposed in [6], but the source code were not available. [7] re-implemented the Reactive-GRASP algorithm (Re-Reactive-GRASP), and we adopt its code in this paper. The code is written in C++ and compiled by the g++ compiler with the -O3 option.
3) Re-Reactive-VNS [7]: We run the source code of Reactive-VNS provided in

[7] (Re-Reactive-VNS). The code is also written in C++ and compiled by the g++ compiler with the -O3 option.
4) Re-GRASP [8]: We run the source code of GRASP provided in [7] (Re-GRASP). The code is written in Java.

Stopping conditions: All experiments are conducted on an Intel (R) Core (TM) 2 Duo CPU T7700 2.40GHz processor with 2GB of RAM. Following [6–8], we run all algorithms 10 times to solve each instance.

The reference algorithms in [6–8] have different stopping conditions (using a maximum allowed number of iterations, a maximum allowed time limit), and can obtain the best results for most instances (80%) within 60 seconds. Therefore, the stopping condition of all algorithms for one run is a time limit of 60 seconds for all instances.

4.3 Comparative study

This section shows summarized comparison results on the three set of benchmark instances. The detailed results for each instance are available at https://github.com/sunseu2022/kMIS, and the optimal values are indicated by asterisks "*".

4.3.1 Computational results on the 360 real-world instances

Table 3 shows the comparative results of OBLMA against the reference algorithms on the 360 real-world instances,in terms of the best result f_{best}, the worst result f_{worst}, and the average result f_{avg} over 10 runs. Recalled that no optimal solutions can be obtained by CPLEX. The first column indicates the name of the two compared algorithms and the second column states the performance indicators. Columns 3/4/5 respectively indicate the number of instances where OBLMA obtains better/equal/worse results compared to each reference algorithm. The last column shows the p-values from the non-parametric Friedman tests, where NA means that the same results were obtained by the compared algorithms. p-values smaller than 0.05 are highlighted in bold to remind significant differences.

Table 3 shows that judging from the lower bounds (LB), OBLMA obtained notably 24 better and 336 equal results against the exact algorithm (CPLEX) (row 1). Besides, OBLMA achieved equal results for all 360 instances against Re-Reactive-GRASP, Re-Reactive-VNS and Re-GRASP in terms of f_{best}, f_{worst}, and f_{avg}.

Table 3
Summary of the comparison between OBLMA and the four reference algorithms on the real-world instances (360 instances)

Algorithm pair	Indicator	Better	Equal	Worse	p-value
OBLMA vs. CPLEX	f_{best}/LB	24	336	0	**9.63E-07**
OBLMA vs. Re-Reactive-GRASP	f_{best}	0	360	0	NA
	f_{worst}	0	360	0	NA
	f_{avg}	0	360	0	NA
OBLMA vs. Re-Reactive-VNS	f_{best}	0	360	0	NA
	f_{worst}	0	360	0	NA
	f_{avg}	0	360	0	NA
OBLMA vs. Re-GRASP	f_{best}	0	360	0	NA
	f_{worst}	0	360	0	NA
	f_{avg}	0	360	0	NA

4.3.2 Computational results on the 238 random instances

The comparative outcomes between OBLMA and the reference algorithms on the 238 random instances are summarized in Table 4 with the same information as in the last section.

Table 4
Summary of the comparison between OBLMA and four reference algorithms on 238 random instances

Algorithm pair	Indicator	Better	Equal	Worse	p-value
OBLMA vs. CPLEX	f_{best}/LB	41	197	0	**1.52E-10**
OBLMA vs. Re-Reactive-GRASP	f_{best}	61	177	0	**5.71E-15**
	f_{worst}	93	145	0	**5.23E-22**
	f_{avg}	93	145	0	**5.23E-22**
OBLMA vs. Re-Reactive-VNS	f_{best}	38	200	0	**7.07E-10**
	f_{worst}	47	191	0	**7.10E-12**
	f_{avg}	50	188	0	**1.54E-12**
OBLMA vs. Re-GRASP	f_{best}	28	210	0	**1.21E-07**
	f_{worst}	39	199	0	**4.24E-10**
	f_{avg}	42	196	0	**9.12E-11**

According to Table 4 and detailed results on the website [4], OBLMA achieved 41 much better and 197 equal lower bounds (including 180 known optimal results) against CPLEX solver. Furthermore, OBLMA outperformed Re-Reactive-GRASP, Re-Reactive-VNS and Re-GRASP. Specifically, in terms of f_{best}, OBLMA dominated Re-Reactive-GRASP (with 61 better and 177 equal results), Re-Reactive-VNS (with 38 better and 200 equal results) and Re-GRASP (with 28 better and 210 equal results). In terms of f_{worst}, OBLMA outperformed reference algorithms Re-Reactive-GRASP/Re-Reactive-VNS/Re-GRASP by obtaining respectively 93/47/39 better results, and 145/191/199

[4] https://github.com/sunseu2022/kMIS

equal results. In terms of f_{avg}, the dominance of OBLMA was even more evident by obtaining respectively 93/50/42 better results, and 145/188/196 equal results against Re-Reactive-GRASP/Re-Reactive-VNS/Re-GRASP. According to the non-parametric Friedman tests, small p-values ($<< 0.05$) implies statistically significant differences between OBLMA and its competitors.

4.3.3 Computational results on the 10 large real-world instances

Table 5 summarized the comparative outcomes of OBLMA against the reference algorithms on the 10 large real-world instances.

Table 5
Summary of the comparison between OBLMA and four reference algorithms on the 10 large real-world instances

Algorithm pair	Indicator	Better	Equal	Worse	p-value
OBLMA vs. CPLEX	f_{best}/LB	9	1	0	**2.70E-03**
OBLMA vs. Re-Reactive-GRASP	f_{best}	9	1	0	**2.70E-03**
	f_{worst}	10	0	0	**1.60E-03**
	f_{avg}	10	0	0	**1.60E-03**
OBLMA vs. Re-Reactive-VNS	f_{best}	3	7	0	8.33E-02
	f_{worst}	5	5	0	**2.53E-02**
	f_{avg}	5	5	0	**2.53E-02**
OBLMA vs. Re-GRASP	f_{best}	3	7	0	8.33E-02
	f_{worst}	8	2	0	**4.70E-03**
	f_{avg}	8	2	0	**4.70E-03**

Table 5 reveals that OBLMA performed remarkably on these very large instances, for which no optimal solutions can be obtained by CPLEX. Compared to CPLEX, OBLMA achieved 9 better and 1 equal lower bounds. Besides, in terms of f_{best}, OBLMA dominated Re-Reactive-GRASP (with 9 better and 1 equal results), Re-Reactive-VNS (with 3 better and 7 equal results), and Re-GRASP (with 3 better and 7 equal results). In terms of f_{worst}, OBLMA outperformed reference algorithms Re-Reactive-GRASP/Re-Reactive-VNS/Re-GRASP by obtaining respectively 10/5/8 better results and 0/5/2 equal results. The comparative results in terms of f_{avg} are same to those in terms of f_{worst}. The p-values of 2.70E-03 for OBLMA vs. CPLEX and OBLMA vs. Re-Reactive-GRASP validate the dominance of OBLMA over these methods in terms of f_{best}. Besides, the small p-values (< 0.05) also imply that our worst and average results differ significantly from those of the reference algorithms.

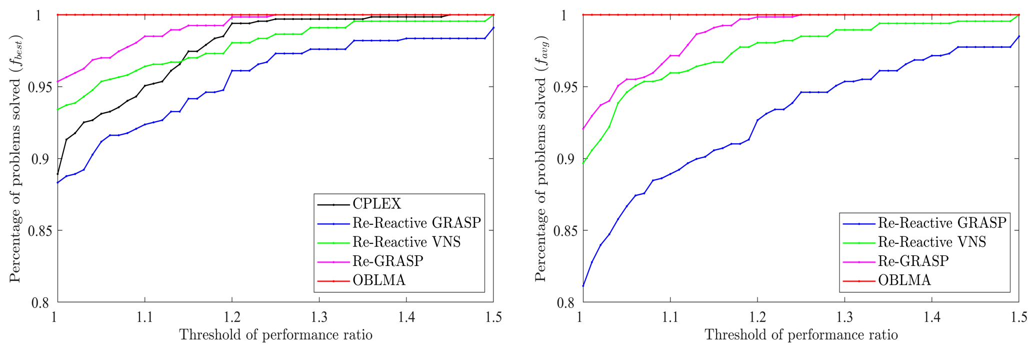
Specifically for this section, OBLMA matched the 360 best results among the 360 real-world instances, hold 24 best results among the 238 random instances, obtained 2 best results among the 10 large real-world instances, and matched the remaining best results, including 180 known optimal results.

4.3.4 The comparison with performance profiles

We present performance profiles [39] to visually illustrate how each algorithm performs, based on f_{best} and f_{avg}. To compare the performance of a set of algorithms W on a set of instances Q, we use $Q_w(\tau)$ as the performance function of each algorithm w to indicate the percentage of instances whose performance ratios are less than τ as follows,

$$Q_w(\tau) = \frac{|\{q \in Q | \alpha_{w,q} \leq \tau\}|}{|Q|} \tag{14}$$

where performance ratio $\alpha_{w,q} = \frac{max\{f_{w,q:w \in W}\}}{f_{w,q}}$, $f_{w,q}$ is the f_{best} or f_{avg} value of instance q of Q got by algorithm w of W, and τ is a threshold set for the performance ratio.



(a) Performance in terms of f_{best} (b) Performance in terms of f_{avg}

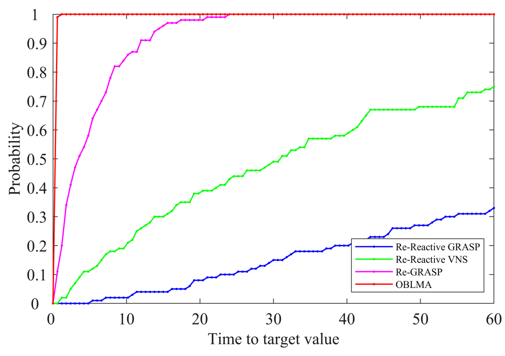
Fig. 7. The performance of OBLMA and four reference algorithms on all 608 benchmark instances.

Figure 7 shows the performance of OBLMA and four reference algorithms on all 608 benchmark instances, where the *X-axis* means the threshold τ and the *Y-axis* means the performance function $Q_w(\tau)$. The value $Q_w(\tau)$ is the proportion of best known results that algorithm w can achieve by scaling its results by τ times. When $\tau = 1$, the value on the *Y-axis* corresponds to the proportion of instances where algorithm w achieves the best value of the set W of compared algorithms.

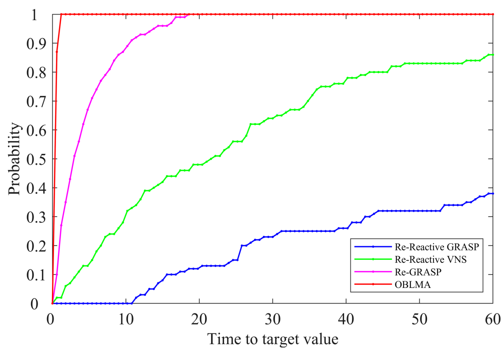
Figure 7 demonstrates the much better performance of OBLMA on the 608 instances against the reference algorithms. Specifically, when $\tau = 1$, in terms of f_{best}, OBLMA could reach 100% best f_{best} values on all these instances, but CPLEX and Re-Reactive-GRASP failed on around 10%, and Re-Reactive-VNS and Re-GRASP failed on around 5%. In terms of f_{avg}, OBLMA reached 100% best values, while reference algorithms failed on at least 8% of the instances. When $\tau = 1.5$, all algorithms could reach at least 98% best f_{best} values and best f_{avg} values.

4.3.5 Time-to-target analysis

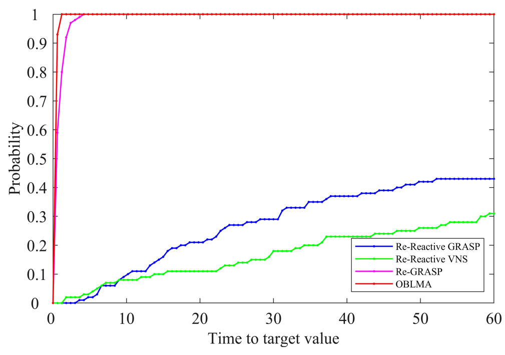
The time-to-target analysis [40,41] aims to assess the computing capability of OBLMA and the reference algorithms. We compare the required time for OBLMA and the reference algorithms to reach a given objective value. This experiment is tested on four out of the 24 representative instances (classe_5_140_140, classe_7_240_240, classe_7_280_224, and classe_8_280_224). The given objective values of these four instances were set to 10, 38, 54 and 14, respectively. OBLMA and the reference algorithms are run 100 times independently with the default experimental settings. Termination of each run occurred immediately upon achieving the specified objective values. Corresponding running time was recorded as the output of the test. Figure 8 illustrates the time-to-target analysis results of OBLMA and the reference algorithms. The *X-axis* and *Y-axis* represent the running time and the cumulative probability of reaching the designated objectives, respectively.



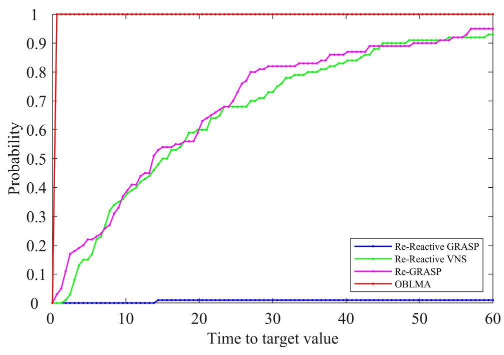
(a) classe_5_140_140



(b) classe_7_240_240



(c) classe_7_280_224



(d) classe_8_280_224

Fig. 8. Time-to-target analysis of OBLMA and the reference algorithms

Figure 8 reveals that the time-to-target lines of OBLMA are almost strictly above those of the reference algorithms, indicating that OBLMA can reach the given objective values with a much higher cumulative probability. Besides, the analysis again validated the competitiveness of OBLMA in computing capability.

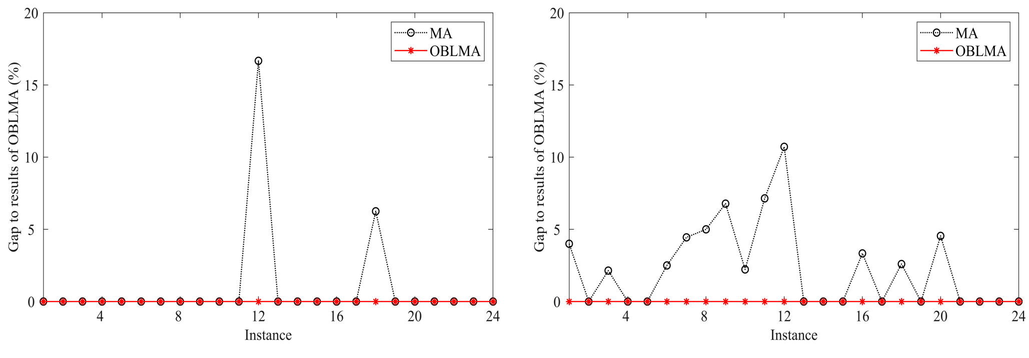
5 Analysis

The core elements of the OBLMA consist of the OBL strategy, the adaptive crossover mechanism, and the multi-swap neighborhood technique. In this analysis, we examine their impact on the OBLMA's effectiveness. The subsequent evaluation adhered to the same experimental setup and time constraints as detailed in Section 4.2. The detailed results of these experiments are available at https://github.com/sunseu2022/kMIS.

5.1 Impacts of OBL strategy

To assess the influences of OBL mechanism on population diversity, we compare OBLMA with its variant MA on the minimum and average distances of individuals in the population. MA stands for the OBLMA without OBL strategy in the population initialization and crossover procedure, which generates two greedy solutions with the greedy randomized construction strategy and employs only the backbone-based crossover, as opposed to OBLMA which generates the greedy solution along with its opposite solution and incorporates the OBL backbone crossover.

Figure 9 shows the results of the comparison between OBLMA and its variant MA in terms of the best results f_{best} and the average results f_{avg}. The *X-axis* indicates the instances which are numbered following the alphabetic order. The *Y-axis* presents the gap between f_{best} (f_{avg}) and the best value (the average value) of new lower bound the NLB_{best} (NLB_{avg}) from OBLMA, as shown in Section 4.3, i.e., $gap = \frac{NLB_{best}-f_{best}}{NLB_{best}} \times 100\%$ for Figure 9(a) and $gap = \frac{NLB_{avg}-f_{avg}}{NLB_{avg}} \times 100\%$ for Figure 9(b).



(a) Comparison in terms of f_{best} (b) Comparison in terms of f_{avg}

Fig. 9. Comparison of OBLMA with its variant MA.

We observe from Figure 9 that OBLMA exhibits a superior performance in terms of the best results and the average results. Specifically, compared with the variant MA, OBLMA obtains 2(12) better results and 22(12) equal results in terms of $f_{best}(f_{avg})$. This indicates the usefulness of the OBL strategy for solving the MkUP under the population-based search.

To complete this experiment, Figure 10 shows the percentage of opposite solutions kept in the population during initialization. The X-$axis$ presents the instances, which are numbered in alphabetic order and the Y-$axis$ indicates the percentage of the opposite solutions kept in the population. The results show that, in 18 out of the 24 instances, about 5% of the opposite solutions are retained in the population. This indicates that only the most promising opposite solutions are preserved, favoring a balance between exploration and exploitation.

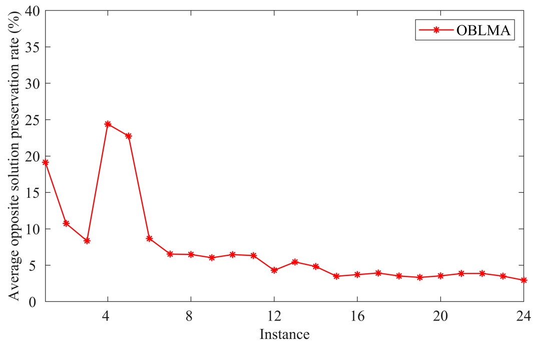


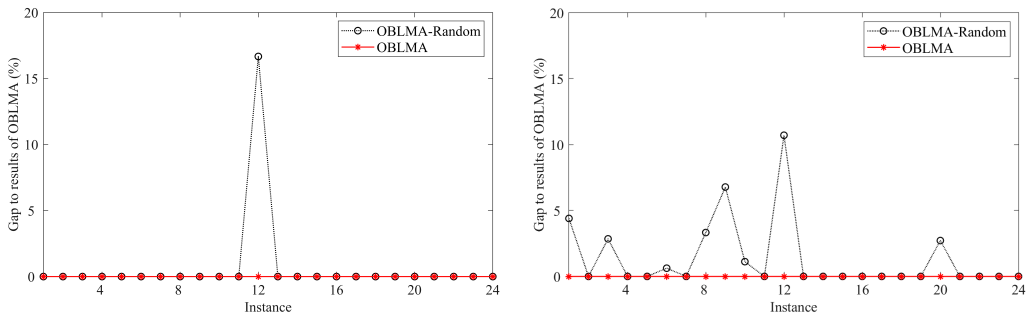
Fig. 10. The proportions of opposite solutions kept in initialization procedure.

5.2 Impacts of the adaptive crossover mechanism

To analyze the effectiveness of the adaptive crossover mechanism, we implement a variant OBLMA-Random, where the learning-based adaptive crossover is replaced by a random crossover.

Figure 11 shows the comparison results between OBLMA and its variant OBLMA-Random using 30 seconds as the stopping condition. Specifically, compared with OBLMA-Random, OBLMA obtains better f_{best} values for one instance out of the 24 instances, better f_{avg} values for 8 instances, and equal results for the remaining instances respectively. The reason for the small difference in terms of the f_{best} in the comparison is that the multi-swap operator can significantly improve the value of f_{best}. Since OBLM-Random adopts the multi-swap operator in the SBTS procedure, it can also obtain an excellent result in terms of f_{best}. However, OBLMA dominates OBLMA-Random in terms of f_{avg}, which means the adaptive crossover mechanism improves the stability and robustness of the algorithm.

To understand the changes of crossover rate parameter γ during the search, we present the variation of γ within a running time of 30 seconds on 24 representative instances in Figure 12. Each colored curve represents the variation of γ of one instance. The X-*axis* shows the running time in seconds and the Y-*axis* indicates the value of γ. The results of Figure 12 show that the values of γ display some large fluctuations within the first 15 seconds, and tend to achieve stable convergence afterward except for two instances. That means that the adaptive crossover procedure of OBLMA dynamically adjusted the γ such that a suitable diversification-intensification balance can be reached for most of the instances.



(a) Comparison in terms of f_{best} (b) Comparison in terms of f_{avg}

Fig. 11. Comparison of OBLMA with its variant OBLMA-Random.

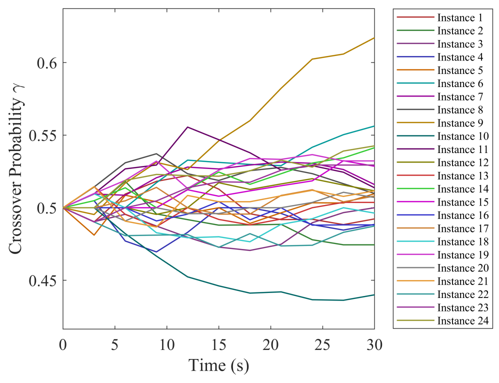
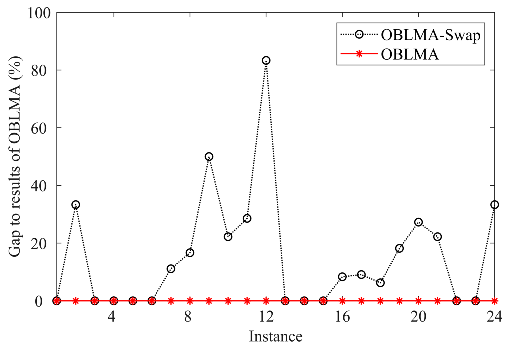


Fig. 12. Variation of the crossover probability γ of OBLMA on 24 representative instances as a function of running time.

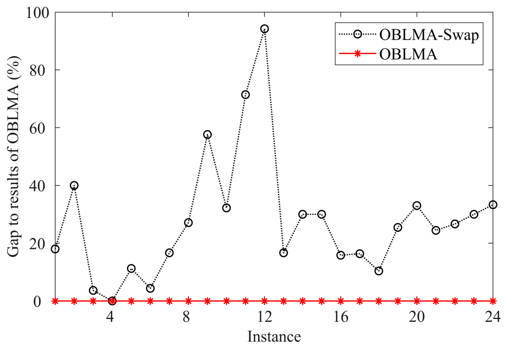
5.3 Impacts of the multi-swap neighborhood

As explained in Section 3.3, OBLMA applies the multi-swap operator in the solution-based tabu search. To justify its merits, a variant of OBLMA, called OBLMA-Swap, is created for comparison. OBLMA-Swap employs the swap operator that exchanges a subset from X with a subset from $S \setminus X$ to generate neighbor solutions.

Figure 13 exhibits the comparitive results of OBLMA and OBLMA-Swap in terms of best and average results (over 10 runs). Specifically, in terms of



(a) Comparison in terms of f_{best}



(b) Comparison in terms of f_{avg}

Fig. 13. Comparison of OBLMA with its variants OBLMA-Swap

f_{best}, OBLMA dominates OBLMA-Swap by obtaining 14 better and 10 equal results. As for f_{avg}, OBLMA performs better by obtaining 23 better and 1 equal results. This indicates that the multi-swap neighborhood guarantees the high performance of OBLMA.

6 Conclusions

The maximum intersection of k-subsets problem (kMIS) is widely applicable in many fields, but difficult to solve due to its NP-hard nature. For this end, we presented in this work an opposition-based learning memetic algorithm that features several complementary search components including an OBL population initialization to construct promising starting population, an adaptive crossover procedure to produce promising offspring, a solution-based tabu search to effectively inspect candidate solution exploration, and a diversity-based pool updating to maintain a healthy population.

We conducted a performance evaluation of our proposed algorithm on a set of 608 instances, comparing it with the state-of-the-art algorithms. The comparative results demonstrated that our algorithm excelled, identifying 26 new lower bounds and achieving the current best results for the remaining instances. Specifically, OBLMA obtains 74 better lower bounds than CPLEX, 70 better results than Re-Reactive-GRASP, 41 better results than Re-Reactive-VNS, and 31 better results than Re-GRASP. Besides, the benefits of the learning-driven strategy and the solution-based tabu search were also investigated.

Several directions can be followed by coming research. First, investigating exact and approximate approaches would help to narrow the gap between the stated and the optimal solutions. Second, different operators from those in this study can be explored to improve the search performance further. Finally, it could be interesting to test the proposed approach on other related problems, such as the knapsack problem [42], the maximum diversity problem [43], and the non-unicost set covering problem [44].

Acknowledgments

We are grateful to the reviewers for their valuable comments and suggestions, which helped us to improve the paper. 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 [Grant No. 2242022K60005].

References

[1] S. A. Vinterbo, A note on the hardness of the k-ambiguity problem, Technical Report, Harvard Medical School, Boston (2002).

[2] J. L. Guillaume, M. Latapy, Bipartite graphs as models of complex networks, Physica A: Statistical Mechanics and its Applications 371 (2) (2006) 795–813.

[3] E. C. Xavier, A note on a maximum k-subset intersection problem, Information Processing Letters 112 (12) (2012) 471–472.

[4] S. Fortunato, Community detection in graphs, Physics Reports 486 (3-5) (2010) 75–174.

[5] D. Nussbaum, S. Pu, J. Sack, T. Uno, H. Zarrabi-Zadeh, Finding maximum edge bicliques in convex bipartite graphs, Algorithmica 64 (2) (2012) 311–325.

[6] E. T. Bogue, C. C. de Souza, E. C. Xavier, A. S. Freire, An integer programming formulation for the maximum k-subset intersection problem, in: P. Fouilhoux, L. E. N. Gouveia, A. R. Mahjoub, V. T. Paschos (Eds.), Combinatorial Optimization - Third International Symposium, Springer, 2014, pp. 87–99.

[7] F. C. S. Dias, W. A. Tavares, J. R. de Freitas Costa, Reactive VNS algorithm for the maximum k-subset intersection problem, Journal of Heuristics 26 (6) (2020) 913–941.

[8] A. Casado, S. Pérez-Peló, J. Sánchez-Oro, A. Duarte, A GRASP algorithm with tabu search improvement for solving the maximum intersection of k-subsets problem, Journal of Heuristics 28 (1) (2022) 121–146.

[9] E. Chlamtác, M. Dinitz, C. Konrad, G. Kortsarz, G. Rabanca, The densest k-subhypergraph problem, SIAM Journal on Discrete Mathematics 32 (2) (2018) 1458–1477.

[10] Y. Naamad, Hardness from Densest Subgraph Conjectures, Princeton University, Princeton, 2017, pp. 96–98.

[11] E. Chlamtác, M. Dinitz, Y. Makarychev, Minimizing the union: Tight approximations for small set bipartite vertex expansion, in: P. N. Klein (Ed.), Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms, 2017, pp. 881–899.

[12] H. R. Tizhoosh, Opposition-based learning: A new scheme for machine intelligence, in: Proceedings of International Conference on Computational Intelligence for Modelling, IEEE Computer Society, 2005, pp. 695–701.

[13] F. S. Al-Qunaieer, H. R. Tizhoosh, S. Rahnamayan, Opposition based computing—a survey, in: The 2010 International Joint Conference on Neural Networks, IEEE, 2010, pp. 1–7.

[14] Q. Xu, L. Wang, N. Wang, X. Hei, L. Zhao, A review of opposition-based learning from 2005 to 2012, Engineering Applications of Artificial Intelligence 29 (2014) 1–12.

[15] Y. M. Zhou, J. K. Hao, B. Duval, Opposition-based memetic search for the maximum diversity problem, IEEE Transactions on Evolutionary Computation 21 (5) (2017) 731–745.

[16] M. Ventresca, H. R. Tizhoosh, A diversity maintaining population-based incremental learning algorithm, Information Sciences 178 (21) (2008) 4038–4056.

[17] Z. Zhang, Z. Xu, S. Luan, X. Li, Y. Sun, Opposition-based ant colony optimization algorithm for the traveling salesman problem, Mathematics 8 (10) (2020) 1650.

[18] S. Rahnamayan, H. R. Tizhoosh, M. M. Salama, Opposition-based differential evolution, IEEE Transactions on Evolutionary computation 12 (1) (2008) 64–79.

[19] D. Lei, L. Cai, F. Wu, Imperialist competition algorithm with quasi-opposition-based learning for function optimization and engineering design problems, Automatika: Journal of Control, Measurement, Electronics, Computing and Communications 65 (4) (2024) 1640–1665.

[20] B. S. Yildiz, N. Pholdee, P. Mehta, S. M. Sait, S. Kumar, S. Bureerat, A. R. Yildiz, A novel hybrid flow direction optimizer-dynamic oppositional based learning algorithm for solving complex constrained mechanical design problems, Materials Testing 65 (1) (2023) 134–143.

[21] P. Mehta, B. Sultan Yildiz, N. Pholdee, S. Kumar, A. Riza Yildiz, S. M. Sait, S. Bureerat, A novel generalized normal distribution optimizer with elite oppositional based learning for optimization of mechanical engineering problems, Materials Testing 65 (2) (2023) 210–223.

[22] M. Pinard, L. Moalic, M. Brévilliers, J. Lepagnot, L. Idoumghar, A memetic approach for the unicost set covering problem, in: I. S. Kotsireas, P. M. Pardalos (Eds.), Learning and Intelligent Optimization - 14th International Conference, Springer, 2020, pp. 233–248.

[23] F. Xu, J. L. Li, A hybrid encoded memetic algorithm for set covering problem, in: Tenth International Conference on Advanced Computational Intelligence, IEEE, 2018, pp. 552–557.

[24] Q. Zhou, U. Benlic, Q. H. Wu, An opposition-based memetic algorithm for the maximum quasi-clique problem, European Journal of Operational Research 286 (1) (2020) 63–83.

[25] J. K. Hao, Memetic algorithms in discrete optimization, in: F. Neri, C. Cotta, P. Moscato (Eds.), Handbook of Memetic Algorithms, Springer, 2012, pp. 73–94.

[26] Y. L. Lu, U. Benlic, Q. H. Wu, A hybrid dynamic programming and memetic algorithm to the traveling salesman problem with hotel selection, Computers & Operations Research 90 (2018) 193–207.

[27] I. Sghir, J. K. Hao, I. B. Jaâfar, K. Ghédira, A multi-agent based optimization method applied to the quadratic assignment problem, Expert Systems with Applications 42 (23) (2015) 9252–9262.

[28] Y. L. Lu, J. K. Hao, Q. H. Wu, Hybrid evolutionary search for the traveling repairman problem with profits, Information Sciences 502 (2019) 91–108.

[29] Q. H. Wu, J. K. Hao, A hybrid metaheuristic method for the maximum diversity problem, European Journal of Operational Research 231 (2) (2013) 452–464.

[30] R. Martí, A. Duarte, M. Laguna, Advanced scatter search for the max-cut problem, INFORMS Journal on Computing 21 (1) (2009) 26–38.

[31] X. J. Lai, J. K. Hao, D. Yue, Two-stage solution-based tabu search for the multidemand multidimensional knapsack problem, European Journal of Operational Research 274 (1) (2019) 35–48.

[32] J. Chang, L. F. Wang, J. K. Hao, Y. Wang, Parallel iterative solution-based tabu search for the obnoxious p-median problem, Computers & Operations Research 127 (2021) 105155.

[33] Z. Q. Wei, J. K. Hao, Multistart solution-based tabu search for the set-union knapsack problem, Applied Soft Computing 105 (2021) 107260.

[34] Z. Q. Wei, J. K. Hao, A threshold search based memetic algorithm for the disjunctively constrained knapsack problem, Computers & Operations Research 136 (2021) 105447.

[35] X. J. Lai, J. K. Hao, F. W. Glover, Z. P. Lü, A two-phase tabu-evolutionary algorithm for the 0-1 multidimensional knapsack problem, Information Sciences 436-437 (2018) 282–301.

[36] W. Sun, J. K. Hao, W. L. Li, Q. H. Wu, An adaptive memetic algorithm for the bidirectional loop layout problem, Knowledge-Based Systems 258 (2022) 110002.

[37] J. Kunegis, KONECT – The Koblenz Network Collection, in: Proceedings of the 22nd International Conference on World Wide Web, 2013, pp. 1343–1350.

[38] E. N. Gilbert, Random graphs, The Annals of Mathematical Statistics 30 (4) (1959) 1141–1144.

[39] E. D. Dolan, J. J. Moré, Benchmarking optimization software with performance profiles, Mathematical Programming 91 (2002) 201–213.

[40] R. M. Aiex, M. G. Resende, C. C. Ribeiro, Ttt plots: a perl program to create time-to-target plots, Optimization Letters 1 (2007) 355–366.

[41] C. C. Ribeiro, I. Rosseti, R. Vallejos, Exploiting run time distributions to compare sequential and parallel stochastic local search algorithms, Journal of Global Optimization 54 (2012) 405–429.

[42] Y. Chen, J. K. Hao, Memetic search for the generalized quadratic multiple knapsack problem, IEEE Transactions on Evolutionary Computation 20 (6) (2016) 908–923.

[43] R. Martí, M. Gallego, A. Duarte, E. G. Pardo, Heuristics and metaheuristics for the maximum diversity problem, Journal of Heuristics 19 (4) (2013) 591–615.

[44] J. E. Beasley, A heuristic for the non-unicost set covering problem using local branching, International Transactions in Operational Research 31 (5) (2024) 2807–2825.

A ILP (integer linear programming) for MkUP

We utilized the 0-1 ILP model in Section 4 for the computations in MkUP, and here are details about it. The model covers following variables:

- $X = \{x_1, x_2, ..., x_n\}$ is taken as a candidate solution.
- $x_i \in \{0, 1\}$ as a binary decision variable that equal to 1 only when subset s_i is selected; 0 otherwise.
- $y_j \in \{0, 1\}$ as a binary logical variable that equal to 1 only when element e_j is uncovered by one of the selected subsets; 0 otherwise.
- V as a binary matrix specifying the relationship between the elements and subsets so that $v_{ij} = 1$ if subset s_i uncovers element e_j; otherwise, $v_{ij} = 0$.

Then, the following ILP model for the MkUP is formed:

$$f(X) = min \sum_{j=1}^{m} y_j \tag{A.1}$$

$$\sum_{i=1}^{n} x_i = k \tag{A.2}$$

$$\sum_{i=1}^{n} v_{ij} x_i \leq y_j, \forall j \in \{1, ..., m\} \tag{A.3}$$

$$x_i, y_j \in \{0, 1\}, \forall i \in \{1, ..., n\}, \forall j \in \{1, ..., m\} \tag{A.4}$$

The objective function (A.1) is to minimize the union of elements uncovered by each selected subset in X. Constraint (A.2) ensures that there are k selected subsets. Constraint (A.3) enforces that an element is in the union if one of the subsets that uncovers it is selected. Constraint (A.4) stipulates the binary nature for variables x_i and y_j.