General swap-based multiple neighborhood adaptive search for the maximum balanced biclique problem

Mingjie Li [a], Jin-Kao Hao [b,c] and Qinghua Wu [a,*]

[a] *School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China, email: lmj@hust.edu.cn; qinghuawu1005@gmail.com*

[b] *LERIA, Université d'Angers, 2 bd Lavoisier, 49045 Angers Cedex 01, France, email: jin-kao.hao@univ-angers.fr*

[c] *Institut Universitaire de France, 1 rue Descartes, 75231 Paris, France*

Computers & Operations Research, 2020.
https://doi.org/10.1016/j.cor.2020.104922

Abstract

The maximum balanced biclique problem (MBBP) is to find the largest complete bipartite subgraph induced by two equal-sized subsets of vertices in a bipartite graph. MBBP is an NP-hard problem with a number of relevant applications. In this work, we propose a general swap-based multiple neighborhood adaptive search (SBMNAS) for MBBP. This algorithm combines a general *k-SWAP* operator which is used in local searches for MBBP for the first time, an adaptive rule for neighborhood exploration and a frequency-based perturbation strategy to ensure a global diversification. SBMNAS is evaluated on 60 random dense instances and 25 real-life large sparse instances from the popular Koblenz Network Collection. Computational results show that our proposed algorithm attains all but one best-known solutions, and finds improved best-known results for 19 instances (new lower bounds).

Keywords: Bipartite graph; balanced biclique; multiple neighborhood; heuristics; *k-SWAP*.

1 Introduction

Given an undirected bipartite graph $G = (U, V, E)$ with disjoint vertex sets U, V and edge set $E \subseteq U \times V$, let X be a subset of U, Y be a subset of

* Corresponding author.

V, and $G' = (X, Y, E(X \cup Y))$ be the subgraph induced by $X \cup Y$. $G' = (X, Y, E(X \cup Y))$ is a biclique if G' is a complete bipartite subgraph, i.e., $\forall x \in X, \forall y \in Y, \{x, y\} \in E$ holds. If additionally, $|X| = |Y|$ holds, G' is a balanced biclique and $|X|$ is its size. A maximum balanced biclique of G has the largest biclique size among the balanced bicliques in G. Then the maximum balanced biclique problem (MBBP) is to find a maximum balanced biclique of a general graph. For simplify, we use (X, Y) to denote the biclique $G' = (X, Y, E(X \cup Y))$.

MBBP is known to be NP-hard [5,7] and is related to the classical maximum clique problem (MCP) [17]. Like MCP which is an useful model for many relevant problems, MBBP has a wide range of practical applications in various domains such as biclustering on gene expression data [4,18], nanoelectronic system design [1,13] and array folding in VLSI [11]. A typical example concerns the problem of defecting tolerance for nanotechnology switches which is to find the largest $n \times n$ defect-free crossbar [1]. A crossbar is composed of two sets of orthogonal nanowires and a set of switches which are located at the intersection of two nanowires (including both functional ones and defective ones, see Figure 1 (a) for an example). Due to the presence of the defects, some switches may be unusable. Thus, one important task is to identify the largest subset of $n \times n$ defect-free nanowires such that the switches between any two orthogonal nanowires are functional. The problem can be formulated by building a graph where the vertex sets U and V correspond to the input nanowires and output nanowires, and the edge set E corresponds to the functional switches. Then the problem of identifying the largest $n \times n$ defect-free crossbar is equivalent to finding the maximum balanced biclique in the corresponding graph (see Figure 1 (b)).

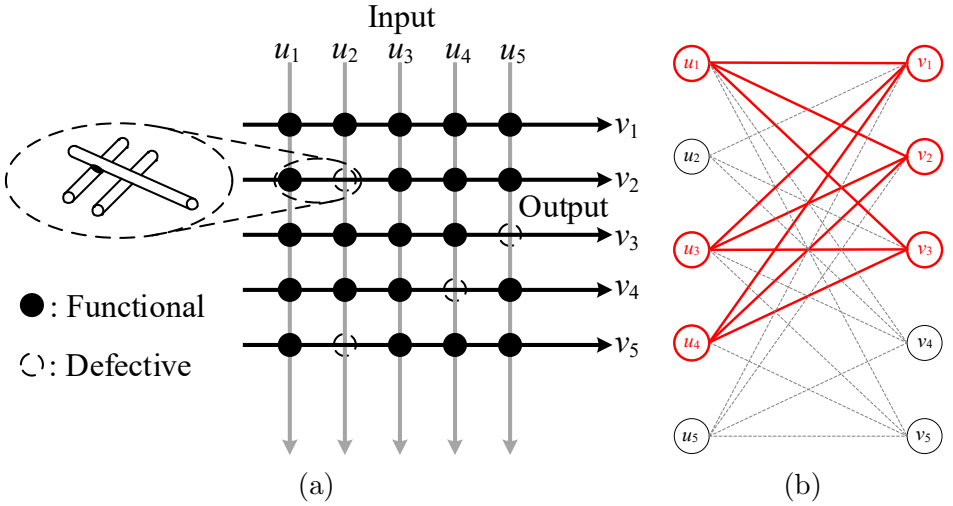


Fig. 1. A defective nanoelectronic crossbar (a) and its associated bipartite graph (b).

Due to the relevance of MBBP, a number of solution methods have been proposed to solve the problem in the literature. On the one hand, there are some exact methods which aim to find the optimal solutions. For instance, Tahoori [13] proposed a recursive exact algorithm to search a maximum balanced biclique with a given size, which was applied to optimally solve instances with size up to $(32, 32)$. Recently, an enhanced branch and bound (B&B) algorithm named ExtBBClq was proposed by Zhou et al. [23], which extends a simple B&B algorithm by integrating a pre-computed upper bounds. A more compact integer linear programming model was also introduced along with the enhanced B&B algorithm by the authors. ExtBBClq was tested on both random dense graphs and real-life large sparse graphs, showing an excellent performance on real-life large graphs.

Exact algorithms can guarantee the optimality of the found solutions, however, their computation time increases exponentially with the size and the density of the graph. For large and dense graphs, a number of heuristic and metaheuristic algorithms have been proposed to find near-optimal solutions in an acceptable computation time. For instance, Tahoori [13] first converted MBBP into a maximum balanced independent set problem and then used a greedy heuristic, which applies a vertex-deletion technique to remove vertices with maximum degrees in the complement graph until the remaining vertices become an independent set. Al-Yamani et al. [1] proposed an improved greedy algorithm by removing the vertex connecting the maximum number of vertices of the minimum degree. Yuan et al. [19,20] studied two other greedy heuristics which remove the vertices adjacent to the maximum number of vertices in a restricted set. The above greedy heuristics solve the equivalent maximum balanced independent set problem and rely on different vertex-deletion rules to remove vertices until a balanced independent set is obtained. Yuan et al. [21] proposed an effective evolutionary algorithm combining structure mutation and local search complemented with a repair-assisted restart process. Quintana et al. [10] introduced a reduced variable neighborhood search (RVNS) algorithm which relies on a random exploration of the considered neighborhoods. Recently, Wang et al. [15] investigated a local search algorithm (called PSRS), which alternates between an extension phase via adding vertex pairs and a restarting phase via removing vertex pairs. In order to solve massive MBBP instances, the authors introduced a two-mode perturbation heuristic (TMP) and a k-bipartite core reduction rule (KCR), leading to an improved algorithm called PSRS+ (PSRS with TMP and KCR). Very recently, Zhou et al. [24] presented a highly effective local search method (TSGR) integrating two graph reduction techniques to shrink the given graph within the tabu search framework. According to the computational results reported in [15,24], PSRS (PSRS+) and TSGR show the best performance among the heuristic approaches for MBBP.

In this work, we propose a general swap-based multiple neighborhood adaptive

search SBMNAS for MBBP. The proposed SBMNAS algorithm distinguishes itself by several important features. First, we introduce an original and generalized *k-SWAP* move operator, which removes one vertex from the biclique and adds $k \geq 1$ vertices adjacent to all but the removed vertex to the biclique such that the resultant biclique is still a feasible biclique. Second, to effectively explore the search space, we propose an adaptive strategy (Section. 2.5) to combine the neighborhoods induced by three different move operators instead of using traditional union or token-ring neighborhood exploration methods. Finally, to ensure a more global diversification, we introduce a dedicated frequency-based perturbation strategy which relies on a long-term memory to restart the search process from a distant region of the search space.

To show the effectiveness of the proposed SBMNAS approach, we carry out experiments on both random dense and real-life large sparse MBBP instances commonly used in the literature. Extensive experimental tests disclose that our SBMNAS approach exhibits an excellent performance. For the 60 random dense instances, the proposed algorithm finds new best solutions for 18 instances, while matching the best-known solutions on all the remaining instances. For the 25 real-life large sparse instances from the popular Koblenz Network Collection (KONECT), our algorithm attains the best-known solutions on all but one instances and improves one of the best-known results.

The rest of the paper is organized as follows. In Section 2, we present the proposed algorithm and its key algorithmic ingredients. In Section 3, we provide the computational results and comparisons with the current state-of-the-art algorithms. In Section 4, we analyze several significant features of the proposed algorithm, prior to conclusions shown in Section 5.

2 General swap-based multiple neighborhood adaptive search

In this section, we present our general swap-based multiple neighborhood adaptive search (SBMNAS) for MBBP. SBMNAS integrates several important features responsible for its effectiveness, including three complementary neighborhoods defined by three basic move operators, an adaptive rule for effective exploration of the neighborhoods, and a dedicated frequency-based perturbation strategy for global diversification.

2.1 Main framework

The general procedure of the SBMNAS approach is shown in Algorithm 1. Starting with an initial biclique (X, Y) (not necessarily balanced), which is

generated by a randomized construction procedure (Section 2.3), SBMNAS first applies its local search procedure (Sections 2.4-2.6) to improve the quality of (X, Y) as far as possible by maximizing its balanced size. The local search procedure jointly explores three neighborhoods and selects at each iteration an admissible neighboring solution to replace the current solution according to the search context. During the search process, each time the best biclique is updated, the graph reduction procedure (Section 2.8) is triggered to reduce the graph by removing vertices whose degrees are smaller than or equal to the balanced size of the current best biclique. If the best biclique is not improved for ω consecutive iterations, the search is deemed to be trapped in deep local optima. In this case, the frequency-based perturbation procedure (Section 2.7) is activated to escape from local optima.

2.2 Search space and solution evaluation

For a given MBBP instance $G = (U, V, E)$ with vertex sets U, V and edge set E, our algorithm explores an enlarged search space including both balanced and unbalanced bicliques, i.e., the explored search space Ω is composed of all complete bipartite subgraphs (X, Y):

$$\Omega = \{(X, Y) : X \subseteq U, Y \subseteq V, E(X \cup Y) = X \times Y\} \tag{1}$$

Noted that it is easy to change an unbalanced biclique to a balanced one by removing some vertices from the larger subset of the current biclique. As shown in Figure 2, the biclique marked in red $(X, Y) = \{\{u_3, u_4\}, \{v_2, v_3, v_4\}\}$ is an unbalanced biclique. Removing any vertex from $Y = \{v_2, v_3, v_4\}$ will lead to a balanced biclique.

For any candidate biclique $(X, Y) \in \Omega$, its quality is assessed by the following evaluation function that indicates the size of the biclique:

$$f(X, Y) = \min(|X|, |Y|) \tag{2}$$

Given two bicliques (A, B) and (C, D), (A, B) is better than (C, D) only if $f(A, B) > f(C, D)$.

Algorithm 1 The general swap-based multiple neighborhood adaptive search

Require: Input graph $G = (U, V, E)$, time limit t_{max}, search depth of the local search ω

Ensure: The best biclique (X^*, Y^*) found

$NoImprove \leftarrow 0$
$r \leftarrow 0$
$p \leftarrow 0.5$ /* p is a global parameter representing the probability of applying ADD operator (Section 2.5) */
$(X, Y) \leftarrow (\emptyset, \emptyset)$
$(X, Y) \leftarrow initial\_construct(G)$ /* Section 2.3 */
$(X^*, Y^*) \leftarrow (X, Y)$ /* (X^*, Y^*) records the best biclique found so far */
while $Time() \leq t_{max}$ **do**
  if $NoImprove < \omega$ **then**
    $(X, Y) \leftarrow local\_search(X, Y)$ /* Section 2.5 */
  else
    $NoImprove \leftarrow 0$
    $(X, Y) \leftarrow frequency\_based\_perturbation(X, Y)$ /* Section 2.7 */
  end if
  if $f(X, Y) > f(X^*, Y^*)$ **then**
    $NoImprove \leftarrow 0$
    $(X^*, Y^*) \leftarrow (X, Y)$ /* Update the best biclique found so far */
    $r \leftarrow f(X^*, Y^*)$
    $G \leftarrow graph\_reduction(r, G)$ /* Section 2.8 */
  else
    $NoImprove \leftarrow NoImprove + 1$
  end if
end while
return (X^*, Y^*)

2.3 Randomized procedure for initial solutions

Our SBMNAS approach begins with an initial biclique $S \in \Omega$ and then applies the local search procedure to improve S as far as possible. The initial biclique (X, Y) is constructed by alternatively adding vertices to the two sets X and Y such that (X, Y) forms a biclique. In the odd iteration, we pick at random a vertex $x \in U \setminus X$ subjected to the stipulation that x is adjacent to all the vertices in Y (i.e., x is taken from the current candidate set $C_X = \{i : i \in U \setminus X, \{i, j\} \in E, \forall j \in Y\}$). Then in the even iteration, we turn to the set $V \setminus Y$ and randomly select a vertex y from $V \setminus Y$ such that y is connected to all the vertices in X (i.e., y is taken from the current candidate set $C_Y = \{j : j \in V \setminus Y, \{i, j\} \in E, \forall i \in X\}$). This procedure is repeated until the current considered set C_X or C_Y becomes empty.

This construction procedure does not guarantee that the biclique is balanced. This poses no problem to our SBMNAS approach given that it explores both balanced and unbalanced bicliques.

2.4 Move operators and neighborhood structures

Typically, local search employs move operators to transform a given solution S into a neighboring solution S'. Let mv be a move operator, we use $S' = S \oplus mv$ to denote such a transformation and $N(S)$ to represent the set of neighboring solutions of S. Our SBMNAS approach jointly uses three basic move operators (denoted by ADD, $DROP$ and k-$SWAP$). Among them, ADD and $DROP$ have previously been explored in [15,24] while k-$SWAP$ is a new operator we introduced in the work. These operators are based on two special vertex subsets which are described below. For our presentation, we adapt the notations used for the maximum weight clique problem [16] to MBBP. Furthermore, for a vertex $i \in U$, we use $K(i)$ to denote the set of vertices adjacent to i in V, i.e., $K(i) = \{j : \{i, j\} \in E, \forall j \in V\}$. Similarly, for a vertex $j \in V$, $K(j)$ is the set of vertices adjacent to j in U, i.e., $K(j) = \{i : \{i, j\} \in E, \forall i \in U\}$.

PA: This subset contains the vertices that are excluded from the current biclique (X, Y) and adjacent to all the vertices of X or Y. Hence, PA can be divided into two disjoint subsets, denoted by PA_X and PA_Y, such that $PA_X = \{i : i \in U \setminus X, \{i, j\} \in E, \forall j \in Y\}$, and $PA_Y = \{j : j \in V \setminus Y, \{i, j\} \in E, \forall i \in X\}$. Note that adding all vertices in PA_X (or PA_Y) once at a time to the current solution (X, Y) can strictly keep (X, Y) to be a feasible biclique.

OM: This subset is composed of the vertices that are excluded from the biclique (X, Y) and adjacent to all but one vertex of X or Y. Similarly, OM can be divided into two disjoint subsets, denoted by OM_X and OM_Y, such that $OM_X = \{u : u \in U \setminus X \wedge |K(u) \cap Y| = |Y| - 1\}$, and $OM_Y = \{v : v \in V \setminus Y \wedge |K(v) \cap X| = |X| - 1\}$.

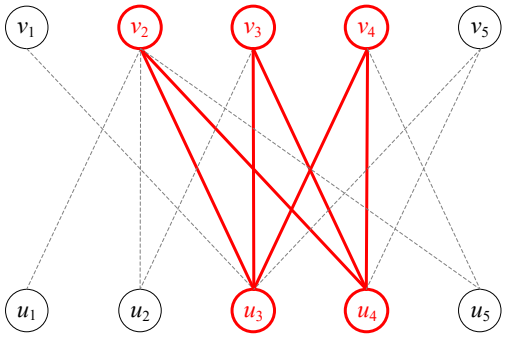


Fig. 2. A biclique and its associated subsets: $X = \{u_3, u_4\}$, $Y = \{v_2, v_3, v_4\}$, $PA_X = \emptyset$, $PA_Y = \{v_5\}$, $OM_X = \{u_2, u_5\}$ and $OM_Y = \{v_1\}$.

An example to illustrate the relationship between a biclique (X, Y) and its associated subsets PA and OM is provided in Figure 2, where $X = \{u_3, u_4\}$, $Y = \{v_2, v_3, v_4\}$, $PA_X = \emptyset$, $PA_Y = \{v_5\}$, $OM_X = \{u_2, u_5\}$ and $OM_Y = \{v_1\}$.

The subsets PA and OM described above constitute the basis for defining the ADD and k-$SWAP$ move operators while the $DROP$ move operator is defined independently of these two subsets.

- ***ADD operator:*** This move operator (which is applied only when PA is not empty) consists in adding all vertices in PA_X to X (when $\min(|X|, |Y|) = |X|$) or adding all vertices in PA_Y to Y (when $\min(|X|, |Y|) = |Y|$). The neighborhood induced by this move operator is defined by: $N_1 = \{(X', Y') : (X', Y') = (X, Y) \oplus ADD(PA_X), (X', Y') = (X, Y) \oplus ADD(PA_Y)\}$. After an ADD move, the change of the balanced size of current biclique (i.e., the move gain denoted by Δ_{ADD}) can be calculated as follows:

$$\Delta_{ADD}(P) = \begin{cases} \min\{|X| + |PA_X|, |Y|\} - \min\{|X|, |Y|\} & \text{if } P = PA_X \\ \min\{|X|, |Y| + |PA_Y|\} - \min\{|X|, |Y|\} & \text{if } P = PA_Y \end{cases} \quad (3)$$

  As the ADD moves insert vertices to the current solution, these moves always lead to neighboring solutions with larger or equal balanced size. Clearly, the size of this neighborhood is bounded by $O(1)$.

- ***DROP operator:*** This move operator deletes a vertex c from the current biclique (X, Y). The neighborhood induced by this move operator is defined by: $N_2 = \{(X', Y') : (X', Y') = (X, Y) \oplus DROP(c), c \in X \cup Y\}$, The move gain $\Delta_{DROP}(c)$ of removing a vertex c can be easily computed by:

$$\Delta_{DROP}(c) = \begin{cases} \min\{|X| - 1, |Y|\} - \min\{|X|, |Y|\} & \text{if } c \in X \\ \min\{|X|, |Y| - 1\} - \min\{|X|, |Y|\} & \text{if } c \in Y \end{cases} \quad (4)$$

  It is obvious that a $DROP$ operator can only lead to a neighboring biclique of equal or worse solution quality relative to the current biclique. The

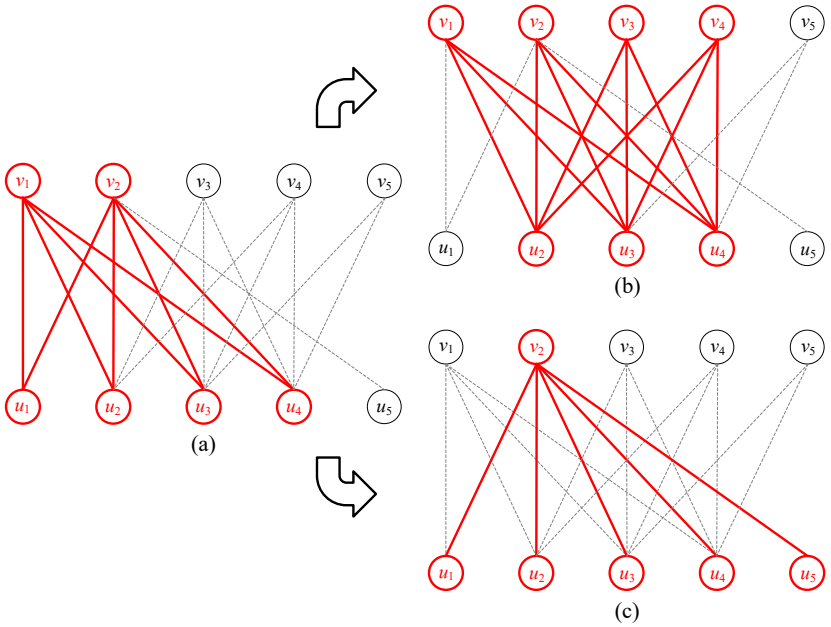


Fig. 3. (a) A biclique and its associated subsets: $X = \{u_1, u_2, u_3, u_4\}$, $Y = \{v_1, v_2\}$, $OM_X = \{u_5\}$ and $OM_Y = \{v_3, v_4\}$; (b) The biclique after swapping u_1 with v_3 and v_4; (c) The biclique after swapping v_1 with u_5.

size of this neighborhood is clearly bounded by $O(|U \cup V|)$.

- ***k-SWAP operator:*** This move operator (which is applied only when OM is not empty) consists in exchanging one vertex c in X (Y) against $k = |OM_Y \setminus (OM_Y \cap K(c))|$ ($k = |OM_X \setminus (OM_X \cap K(c))|$) vertices from the set OM_Y (OM_X) such that each of these k newly added vertices are commonly unconnected with c. Note that since each of these k vertices are adjacent to all vertices in X (Y) except the removed one, a *k-SWAP* move always leads to a feasible biclique. The neighborhood induced by this move operator is defined by: $N_3 = \{(X', Y') : (X', Y') = (X, Y) \bigoplus k\text{-}SWAP(c), c \in X \cup Y\}$. The move gain $\Delta_{k\text{-}SWAP}$ induced by the *k-SWAP* move can be conveniently calculated by:

$$\Delta_{k\text{-}SWAP}(c) = \begin{cases} \min\{|X| - 1, |Y| + |OM_Y \setminus (OM_Y \cap K(c))|\} - \min\{|X|, |Y|\} & \text{if } c \in X \\ \min\{|X| + |OM_X \setminus (OM_X \cap K(c))|, |Y| - 1\} - \min\{|X|, |Y|\} & \text{if } c \in Y \end{cases} \quad (5)$$

Clearly, each application of *k-SWAP* can increase the balanced size of (X, Y), keep its balanced size unchanged, or decrease the balanced size of (X, Y). The size of this neighborhood is bounded by $O(|U \cup V|)$.

Figure 3 provides an example to illustrate the *k-SWAP* move operator. Suppose that $(X, Y) = \{\{u_1, u_2, u_3, u_4\}, \{v_1, v_2\}\}$ is the current biclique with a balanced size of 2. When swapping $u_1 \in X$ with $\{v_3, v_4\} \subseteq OM_Y$, we obtain an improved neighboring solution $(X_1, Y_1) = \{\{u_2, u_3, u_4\}, \{v_1, v_2, v_3, v_4\}\}$

with a balanced size equal to 3. Similarly, we can also exchange $v_1 \in Y$ with $\{u_5\} \subseteq OM_X$, leading to a biclique $(X_2, Y_2) = \{\{u_1, u_2, u_3, u_4, u_5\}, \{v_2\}\}$ with a balanced size equal to 1.

One notices that the special case 1-*SWAP* (i.e., with $k = 1$) was previously used in [24], which exchanges a vertex in the current biclique (X, Y) against another vertex outside the current biclique. Compared with *1-SWAP*, our proposed *k-SWAP* operator implies an enlarged neighborhood with more candidate neighboring solutions with $k > 1$. A local search algorithm based on this operator is offered a higher chance to find high-quality solutions. As demonstrated in Section 4.1, the *k-SWAP* move operator plays a critical role to the overall performance of the proposed SBMNAS algorithm. Finally, the *k-SWAP* operator shares the basic idea of the general *PUSH* operator introduced in [22] for the related maximum clique problem. However, the *PUSH* operator inserts one vertex to the clique and removes all vertices non-adjacent to the inserted vertex from the clique, while the *k-SWAP* operator removes one vertex from the biclique and inserts k vertices into the biclique such that each added vertex is connected to all but the removed vertex in the biclique.

2.5 Adaptive local search procedure

As observed in previous studies such as [12,14], for strongly constrained optimization problems, strictly imposing problem constraints during the search often prevent the search from exploring new promising areas and sometimes may even make the search blocked. One effective method to avoid this situation is to relax the problem constraints and allow a controlled exploration of infeasible solutions relative to the problem constraints. Following this idea, the SBMNAS algorithm does not impose the balance constraint of candidate solutions. Instead, it explores both balanced and unbalanced bicliques, knowing that an unbalanced biclique can be transformed to a balanced one by simply removing some vertices from the largest subset.

On the other hand, to improve the balanced size of a given biclique (X, Y), we have to increase the size of the smaller subset during the search process. As such, it is useless to consider moves expanding the larger subset since these moves will not help to improve the quality of (X, Y). More importantly, such moves can make the situation even worse, since each newly inserted vertex in the smaller subset needs to be adjacent to all vertices in the larger subset, a larger number of vertices in the larger subset generally makes it more difficult to insert vertices to the smaller subset. Thus, to improve the quality of a biclique (X, Y), we should definitively increase the size of the smaller subset and keep the size difference between the two subsets as small as possible.

Based on the above consideration, we introduce the following local search strategy to explore the search space of MBBP (see also Algorithm 2). For a given solution (X, Y) with $|X| > |Y|$, the *right adaptive search* procedure is triggered to add vertices to subset Y by applying the ADD or k-$SWAP$ move. If no ADD or k-$SWAP$ move can be performed, some vertices are removed from subset X. Similarly when $|X| < |Y|$, the *left adaptive search* procedure is activated which works in the same way as the right adaptive search by adding vertices to subset X or removing vertices from Y. When $|X| = |Y|$, these two search procedures are selected to be performed with equal probability.

Algorithm 2 Adaptive local search procedure

Require: A biclique (X, Y)
Ensure: The maximum biclique (X^*, Y^*) found

if $|X| > |Y|$ **then**
  $(X, Y) \leftarrow right\_adaptive\_search(X, Y)$
else if $|X| < |Y|$ **then**
  $(X, Y) \leftarrow left\_adaptive\_search(X, Y)$
else if $|X| = |Y|$ **then**
  if $Random()\%2 == 0$ **then**
    $(X, Y) \leftarrow right\_adaptive\_search(X, Y)$
  else
    $(X, Y) \leftarrow left\_adaptive\_search(X, Y)$
  end if
end if
return (X^*, Y^*)

When several basic neighborhoods are available, one key issue is how to combine these neighborhoods so as to effectively explore the search space. There are several methods to combine different neighborhoods, such as token-ring search and neighborhood union. For instance, the neighborhoods induced by the ADD and 1-$SWAP$ moves are explored sequentially in [24]. One key motivation for considering combination of diverse neighborhoods is to allow the search to go beyond local optima and continue its exploration toward better solutions. In the right (left) adaptive search procedure of our SBMNAS approach, we propose an adaptive method to favor the selection of the particular neighborhood that produces solutions of better quality.

Without loss of generality, Algorithm 3 summarized the main procedure of the right adaptive search for the improvement of a biclique (X, Y) with $|X| \geq |Y|$. Specifically, we use p and $1-p$ to respectively indicate the probability of selecting the N_1 and N_3 neighborhoods. Before the first search, both neighborhoods are selected with equal probability by setting $p = 0.5$. During the search, when none of PA_Y and OM_Y is empty, we select one of the two neighborhoods according to the given probability. For the ADD move, we add all non-tabu (see Section 2.6) vertices of PA_Y into subset Y while for the k-$SWAP$ move,

a non-tabu vertex $x \in X$ with the maximum number of non-adjacent vertices in OM_Y is selected to be exchanged with these non-adjacent vertices in OM_Y. Whenever a move improves the balanced size of the best solution ever found, the probability of selecting such a move operator is increased by λ ($0 < \lambda < 1$) during the subsequent search. Finally, the $DROP$ move which randomly deletes a vertex of X is only applied when both PA_Y and OM_Y are empty. During the search process, in order to prevent the ADD and k-$SWAP$ operators from being applied with a too high or too low probability, we strictly limit p in the range $[\phi, 1-\phi]$, where ϕ (a parameter) takes its values in $[0, 0.5]$. The calibration of λ and ϕ is explained in Section 3.1.

2.6 Tabu list and aspiration rule

Our SBMNAS algorithm employes a general tabu rule [6] to avoid short-term cycles. Each time a vertex $i \in (X, Y)$ is removed from the current biclique (X, Y) (by the $DROP$ or k-$SWAP$ move operator), i is marked tabu for the next T_i iterations, during which i cannot be put back into the current biclique (X, Y). Similarly, when a vertex $o \notin (X, Y)$ is added to the current biclique (X, Y) (by the ADD or k-$SWAP$ move operator), o is marked tabu for the next T_o iterations, during this period o is forbidden to be removed from (X, Y). Both T_i and T_o are called tabu tenure and adjusted dynamically according to the balanced size of the current biclique (X, Y):

$$T_i = \min\{tt,\ 0.2r + 0.1Random(r)\} \text{ and}$$
$$T_o = \min\{0.6tt,\ 0.12r + 0.1Random(0.6r)\}$$

where tt is a parameter called basic tabu tenure, $r = \min\{|X|, |Y|\}$ and the function $Random(L)$ returns a random integer number in $\{0, 1, ..., L-1\}$. One notices that the tabu tenure T_i for i (the vertex leaving (X, Y)) is larger than the tabu tenure T_o for o (the vertex joining (X, Y)). This can be explained by fact that generally, there are much fewer vertices included in (X, Y) than outside (X, Y). As a consequence, preventing vertices in the current biclique (X, Y) from being removed is much more restrictive than preventing vertices outside (X, Y) from being added to (X, Y).

Finally, a move is declared tabu if one of its involved vertices is marked tabu. The tabu status of a move is however ignored if it produces a solution better than the best solution found so far.

Algorithm 3 Right adaptive search

Require: A biclique (X, Y) with $|X| \geq |Y|$
Ensure: The biclique (X^*, Y^*) with largest balanced size

if $PA_Y \neq \emptyset$ and $OM_Y \neq \emptyset$ **then**
  if with fixed probability p **then**
    $(X, Y) \leftarrow (X, Y) \bigoplus ADD(PA_Y)$
    if $f(X, Y) > f(X^*, Y^*)$ **then**
      $p \leftarrow p + \lambda$ /*Increase the probability of applying ADD operator by λ $(0 < \lambda < 1)$ */
    end if
  else
    $(X, Y) \leftarrow (X, Y) \bigoplus k\text{-}SWAP(i)$
    if $f(X, Y) > f(X^*, Y^*)$ **then**
      $p \leftarrow p - \lambda$ /*Increase the probability of applying k-$SWAP$ operator */
    end if
  end if
else if $OM_Y \neq \emptyset$ **then**
  $(X, Y) \leftarrow (X, Y) \bigoplus k\text{-}SWAP(i)$
  if $f(X, Y) > f(X^*, Y^*)$ **then**
    $p \leftarrow p - \lambda$
  end if
else if $PA_Y \neq \emptyset$ **then**
  $(X, Y) \leftarrow (X, Y) \bigoplus ADD(PA_Y)$
  if $f(X, Y) > f(X^*, Y^*)$ **then**
    $p \leftarrow p + \lambda$
  end if
else if $OM_Y == \emptyset$ and $PA_Y == \emptyset$ **then**
  $(X, Y) \leftarrow (X, Y) \bigoplus DROP(i)$
end if
if $p < \phi$ **then**
  $p \leftarrow \phi$
else if $p > 1 - \phi$ **then**
  $p \leftarrow 1 - \phi$
end if
return (X^*, Y^*)

2.7 Frequency-based perturbation

The above tabu rule is able to prevent short-term cycles, but this mechanism may be insufficient to escape deep local optima. To encourage the algorithm to explore new areas in the search space, and therefore establish a more global form of diversification, we introduce a dedicated perturbation strategy which is triggered when the search is judged to be definitively stagnating (i.e., when

Algorithm 4 Frequency-based perturbation

Require: A biclique (X, Y), perturbation strength coefficient β
Ensure: The biclique (X, Y) after perturbation

```
L = β * min(|X|, |Y|)
for count = 1, 2, ..., L do
    select u with the highest frequency in X
    select v with the highest frequency in Y
    (X, Y) ← (X, Y) ⊕ DROP(u)
    (X, Y) ← (X, Y) ⊕ DROP(v)
end for
return (X, Y)
```

the best biclique is not improved for ω consecutive iterations).

The perturbation strategy is achieved by applying a destructive procedure to remove vertices from the current biclique. To achieve better search diversification, we use frequency information gathered during the search to guide the selection of vertices to be removed. Specifically, we maintain a long-term frequency memory $Fre(i), i \in U \cup V$ to record the number of times that a vertex has been moved. Our proposed perturbation strategy is summarized in Algorithm 4 and can be described as follows.

1. Initially, set $Fre(i) = 0$ for each vertex $i \in U \cup V$.
2. Subsequently during the local search, each time vertex i is put into or removed from the current biclique (X, Y), the frequency of i is increased by 1.
3. During the destructive procedure, we select L vertices with the highest frequency in X and L vertices with the highest frequency in Y and remove these selected vertices from (X, Y). L is called perturbation strength and is computed as $L = \beta * \min(|X|, |Y|)$, where β is a parameter called perturbation strength coefficient.
4. After the destructive procedure is applied, we refresh the frequency counters by setting $Fre(i) = 0$ for each vertex $i \in U \cup V$. This refreshing strategy definitively prevents a vertex from reaching a too high frequency and thus always being selected during the whole search.

2.8 Graph reduction

Given a bipartite graph $G = (U, V, E)$ and a known balanced size r of the largest biclique found so far, if a vertex i has at most r adjacent vertices in the other subset, it is obviously impossible for i to extend the current biclique to a biclique with a balanced size larger than r. As a consequence, such a vertex can be safely removed from the graph to shrink the graph. This

graph reduction technique was previously used in [15,24], and shown to be very effective on massive sparse graphs. The graph reduction procedure can be implemented in time $O(|U \cup V| + |E|)$.

3 Computational assessment

In this section, we report the computational results of our proposed SBMNAS algorithm on two sets of benchmark instances commonly used in the literature, and show comparisons with state-of-the-art MBBP algorithms including three heuristic algorithms (PSRS/PSRS+ [15], TSGR [24]), and one exact algorithm (ExtBBClq [23]).

3.1 Benchmark instances and experimental protocol

For performance assessments, we used two sets of 85 benchmark instances including random dense graphs and real-life large sparse graphs.

- The first set of benchmark instances includes 60 random dense instances, including 30 smaller graphs introduced in [19,20] and 30 larger graphs produced recently [15]. In each graph, the two vertex subsets U and V have an equal size. An edge between two vertices respectively in U and V is generated with uniform probability p_1 which characterizes the density of the graph. For each combination of $|U| \in \{250, 500, 1000, 5000\}$ and $p_1 \in \{0.85, 0.90, 0.95\}$, 5 graphs are generated, leading to a total of 60 instances. These graphs are named as $< A > p < B > n < id >$ [15], where A denotes the size of the graph, B represents the density of the graph, and $id \in \{1, 2, 3, 4, 5\}$.
- The second set of benchmark instances includes 25 real-life large sparse graphs selected from the Koblenz Network Collection (KONECT) [8]. The KONECT dataset was originally collected from network analysis and contains hundreds of networks from various real-life applications, such as rating networks, affiliation networks, and interaction networks. These 25 selected graphs are massive sparse ranging in size from 829+551 vertices and 1476 edges to 1,425,813+4,000,150 vertices and 8,649,016 edges.

Our SBMNAS algorithm is programmed in C++ and compiled by GNU g++ compiler on a computer with an Intel Core i5-8400 CPU (2.80GHz) and 16GB RAM with the -O3 option flag under Linux.

SBMNAS requires five main parameters (see Table 1). In order to tune the parameters, we use the IRACE software package which applies the iterated racing method [2,3,9] to find an appropriate parameter configuration from a set

of finite parameter configurations. We run IRACE on 20 randomly selected instances including 10 random dense instances and 10 real-life large sparse instances, and set the tuning budget to 2000 executions with a time limit of 60s. The tested and the final values recommended by IRACE are shown in Table 1.

Table 1
Settings of the parameters.

Parameter	Section	Description	Considered values	Final value
ω	2	search depth of the local search (dense graph)	{100, 300, 500, 700, 1000}	500
		search depth of the local search (sparse graph)	{30, 50, 70, 100, 150}	100
λ	2.5	adjustment to the probability of applying the ADD and k-$SWAP$ operators	{0.03, 0.05, 0.07, 0.10, 0.15}	0.05
ϕ	2.5	adjustment to the probability of applying the ADD and k-$SWAP$ operators	{0, 0.1, 0.2, 0.3, 0.4, 0.5}	0.2
tt	2.6	basic tabu tenure (dense graph)	{10, 15, 20, 25, 30}	25
		basic tabu tenure (sparse graph)	{5, 10, 15, 20, 25}	10
β	2.7	perturbation strength coefficient (dense graph)	{0.04, 0.07, 0.10, 0.13, 0.16 }	0.10
		perturbation strength coefficient (dense graph)	{0.10, 0.30, 0.50, 0.70, 0.90 }	0.50

3.2 Computational results and comparisons with state-of-the-art methods

According to two recent studies [15,24], PSRS [15] and TSGR [24] exhibit an overall best performance among all existing heuristics for random dense instances while PSRS+ [15] and TSGR show an overall best performance for real-life large sparse instances. Thus, these three approaches constitute state-of-the-art heuristic algorithms for MBBP.

To make a fair comparison with the these reference algorithms (PSRS, PSRS+ and TSGR), we run their source codes as well as our algorithm under the same stopping condition as that used by PSRS. That is, for random dense instances, the time limit is set to 100 seconds for graphs with less than 1000 vertices, and 1000 seconds for larger graphs; for the KONECT graphs, the time limit is set to 1000 seconds. Due to the stochastic nature of these compared algorithms, each instance is independently solved 10 times with different random seeds by each method. To further verify the effectiveness of our SBMNAS algorithm, we also include the results of the exact algorithm ExtBBClq [23] in our comparison. For ExtBBClq, we run its source code under our computing platform under the time limit of 3 hours as in [23]. Since ExtBBClq is an exact algorithm, one run per instance suffices. Furthermore, as shown in [23], ExtBBClq is not really effective for random dense graphs with more than 50 vertices, so we only report the results of ExtBBClq on the real-life large sparse instances.

3.2.1 Computational results on random dense graphs

Table 2
Comparative results on the 30 small random dense graphs.

Instance	f_{bk}	PSRS			TSGR			SBMNAS			
		f_{best}	f_{avg}	t_s	f_{best}	f_{avg}	t_s	f_{best}	f_{avg}	t_s	Gap
250p95n1	68	**68**	**68**	0.28	**68**	**68**	0.00	**68**	**68**	0.00	0.00
250p95n2	66	**66**	**66**	0.86	**66**	**66**	0.01	**66**	**66**	0.01	0.00
250p95n3	70	**70**	**70**	0.47	**70**	**70**	0.01	**70**	**70**	0.01	0.00
250p95n4	68	**68**	**68**	1.34	**68**	**68**	0.03	**68**	**68**	0.01	0.00
250p95n5	68	**68**	**68**	12.37	**68**	**68**	0.07	**68**	**68**	0.01	0.00
250p90n1	44	**44**	**44**	0.30	**44**	**44**	0.00	**44**	**44**	0.00	0.00
250p90n2	45	**45**	**45**	3.83	**45**	**45**	0.10	**45**	**45**	0.02	0.00
250p90n3	44	**44**	**44**	0.91	**44**	**44**	0.01	**44**	**44**	0.04	0.00
250p90n4	45	**45**	**45**	3.74	**45**	**45**	0.09	**45**	**45**	0.05	0.00
250p90n5	45	**45**	**45**	2.26	**45**	**45**	0.02	**45**	**45**	0.02	0.00
250p85n1	33	**33**	**33**	0.87	**33**	**33**	0.01	**33**	**33**	0.02	0.00
250p85n2	33	**33**	**33**	0.44	**33**	**33**	0.01	**33**	**33**	0.00	0.00
250p85n3	34	**34**	33.9	22.46	**34**	**34**	0.09	**34**	**34**	0.13	0.00
250p85n4	33	**33**	**33**	0.24	**33**	**33**	0.01	**33**	**33**	0.01	0.00
250p85n5	33	**33**	**33**	2.31	**33**	**33**	0.06	**33**	**33**	0.01	0.00
500p95n1	93	**93**	92.6	38.73	**93**	**93**	0.91	**93**	**93**	0.13	0.00
500p95n2	91	**91**	**91**	30.75	**91**	**91**	0.76	**91**	**91**	0.20	0.00
500p95n3	91	**91**	90.1	22.52	**91**	90.6	8.28	**91**	**91**	2.47	0.00
500p95n4	90	89	89	13.68	**90**	**90**	4.39	**90**	**90**	2.77	0.00
500p95n5	91	**91**	**91**	31.35	**91**	**91**	0.75	**91**	**91**	0.19	0.00
500p90n1	56	**56**	55.8	33.60	**56**	**56**	0.78	**56**	**56**	0.32	0.00
500p90n2	56	**56**	**56**	34.36	**56**	**56**	0.20	**56**	**56**	0.21	0.00
500p90n3	56	**56**	55.1	77.11	**56**	**56**	2.20	**56**	**56**	8.49	0.00
500p90n4	56	**56**	55.1	26.48	**56**	**56**	0.23	**56**	**56**	2.52	0.00
500p90n5	56	**56**	55.9	55.67	**56**	**56**	3.07	**56**	**56**	1.17	0.00
500p85n1	40	**40**	**40**	29.97	**40**	**40**	0.40	**40**	**40**	0.16	0.00
500p85n2	41	**41**	**41**	22.90	**41**	**41**	0.31	**41**	**41**	0.31	0.00
500p85n3	41	**41**	40.4	39.88	**41**	**41**	4.17	**41**	**41**	7.33	0.00
500p85n4	40	**40**	**40**	11.47	**40**	**40**	0.17	**40**	**40**	0.12	0.00
500p85n5	41	**41**	40.9	36.02	**41**	**41**	0.38	**41**	**41**	0.39	0.00
#Best		29	20	0/29	30	29	13/29	30	30	21/29	
p-value		0.32	1.60e-3	7.24e-8	1.00	0.32	0.20				

Table 2 summarizes the comparative results of these three compared heuristic algorithms on the 30 small random dense graphs with 250 and 500 vertices. In Table 2, column 'f_{bk}' presents the previous best-known results reported in the literature, column 'Gap' gives the percent gap between the best balanced size obtained with SBMNAS and the best-known balanced size f_{bk}, which is computed as $100 * (f_{bk} - f_{best})/f_{bk}$. For each compared approach, columns 'f_{best}', 'f_{avg}' and 't_s' respectively show the best balanced size, the average balanced size and the average computation time in seconds used to reach the best solution. The best values in terms of 'f_{best}' and 'f_{avg}' among the compared algorithms are marked in bold. The row '#Best' indicates the number of cases for which each algorithm obtains the best results among the compared methods. Row 'p-value' shows the results from the non-parametric Friedman test applied to the results of SBMNAS and each compared algorithm, and a p-value smaller than 0.05 implies a significant difference between the compared results. Note that to compute the p-values in terms of average computation time (t_s), the Friedman test is conducted only on these instances where the compared algorithms reach the same biclique size, this indeed shows how fast an algorithm finds a solution of same quality.

From Table 2, we observe that TSGR and SBMNAS match all the current best-known results for these 30 small random dense instances while PSRS

misses the best result for one instance. In terms of the average objective values, SBMNAS yields the best values on all 30 instances whereas PSRS and TSGR produce the best values on 20 and 29 instances respectively. Among the 29 instances where the compared algorithms obtain the same best objective value, SBMNAS requires the shortest computation times for 21 instances against 0 case for PSRS and 12 cases for TSGR.

The non-parametric Friedman test applied to the results between SBMNAS and the two reference algorithms leads to the following p-values: 0.32 for PSRS, and 1.00 for TSGR in terms of best objective values; 1.60e-3 for PSRS, and 0.32 for TSGR in terms of average objective values; 7.24e-8 for PSRS, and 0.20 for TSGR in terms of average computation time. These results reveal that for the set of small dense instances, the difference between SBMNAS and TSGR is not statistically significant in terms of the used comparison indicators, while the difference between SBMNAS and PSRS is significant in both terms of average objective values and average computation time, but not statistically significant in terms of best objective values.

Table 3
Comparative results on the 30 large random dense graphs.

Instance	f_{bk}	PSRS			TSGR			SBMNAS			
		f_{best}	f_{avg}	t_s	f_{best}	f_{avg}	t_s	f_{best}	f_{avg}	t_s	Gap
1000p95n1	114	114	113.8	321.36	114	113.7	230.31	**115**	**114.2**	912.71	-0.88
1000p95n2	114	115	114.8	387.17	115	114.4	237.59	**116**	**115.7**	355.35	-1.75
1000p95n3	115	114	113.7	424.70	114	113.6	269.01	**115**	**114.1**	798.54	0.00
1000p95n4	115	114	113.4	341.41	114	113.4	125.19	**115**	**114.1**	854.63	0.00
1000p95n5	115	114	113.9	325.56	114	113.5	201.85	**115**	**114.2**	965.32	0.00
1000p90n1	66	66	66	312.17	66	66	32.73	**67**	**67**	368.15	-1.52
1000p90n2	67	**67**	66.6	455.19	**67**	66.3	182.05	**67**	**67**	40.98	0.00
1000p90n3	66	66	66	305.63	66	66	32.96	**67**	**67**	240.49	-1.52
1000p90n4	67	**67**	66.7	315.72	**67**	**67**	240.55	**67**	**67**	9.62	0.00
1000p90n5	67	**67**	66.7	483.92	**67**	66.5	221.61	**67**	**67**	14.19	0.00
1000p85n1	47	**47**	46.3	233.80	**47**	**47**	119.42	**47**	**47**	11.36	0.00
1000p85n2	47	**47**	46.2	611.13	**47**	46.9	120.48	**47**	**47**	21.78	0.00
1000p85n3	47	47	46.7	201.57	47	47	41.30	**48**	**48**	261.49	-2.13
1000p85n4	47	**47**	46.7	532.62	**47**	**47**	96.93	**47**	**47**	3.13	0.00
1000p85n5	47	**47**	46.4	389.70	**47**	**47**	88.57	**47**	**47**	6.55	0.00
5000p95n1	159	159	158	883.47	156	155.7	234.09	**161**	**160.3**	461.68	-1.26
5000p95n2	158	159	157.5	521.38	156	155.7	181.16	**161**	**160.1**	564.79	-1.90
5000p95n3	159	158	157.3	565.53	156	155.6	350.58	**161**	**160**	842.32	-1.26
5000p95n4	158	159	158	702.56	156	155.6	145.91	**162**	**160.6**	240.75	-2.53
5000p95n5	158	159	157.4	830.08	156	155.5	214.23	**161**	**160.1**	866.69	-1.90
5000p90n1	86	85	83.6	925.85	85	84.8	152.39	**86**	**85.3**	842.24	0.00
5000p90n2	85	84	83.7	406.00	**86**	85.1	195.66	**86**	**85.2**	353.89	-1.18
5000p90n3	86	85	83.7	765.43	85	84.9	156.86	**86**	**85.3**	581.86	0.00
5000p90n4	85	85	83.9	513.07	85	84.7	135.68	**86**	**85.1**	7.01	-1.18
5000p90n5	85	85	84.2	603.86	85	84.7	179.43	**86**	**85.1**	875.12	-1.18
5000p85n1	57	57	56.9	571.99	**59**	57.9	169.95	**59**	**58.8**	468.00	-3.51
5000p85n2	58	57	56.7	451.83	58	57.9	153.15	**59**	**58.9**	445.08	-1.72
5000p85n3	58	57	56.6	452.84	58	57.9	197.13	**59**	**58.8**	434.87	-1.72
5000p85n4	57	57	56.4	737.23	58	57.7	131.64	**59**	**59**	393.41	-3.51
5000p85n5	57	58	56.8	248.34	58	57.8	160.57	**59**	**58.8**	440.40	-3.51
#Best		7	0	0/7	9	4	0/7	30	30	7/7	
p-value		1.62e-6	4.32e-8	8.15e-3	4.59e-6	3.41e-7	8.15e-3				

Table 3 summarizes the comparative results of these three compared heuristic algorithms on the 30 large random dense graphs with 1000 and 5000 vertices. It can be seen from Table 3 that, our SBMNAS algorithm achieves excellent results with respect to the previous best-known results reported in the literature. Precisely, SBMNAS improves the best-known results for 18 instances

(indicated by negative '*Gap*' values), and matches all the best-known results on the remaining 12 instances. When comparing SBMNAS with the two reference algorithms, the results obtained with SBMNAS remain highly competitive. In terms of the best objective values, SBMNAS achieves the best results on all 30 instances, while PSRS and TSGR yield the best results on 7, and 9 cases respectively. In terms of the average objective values, SBMNAS yields the best values on all 30 instances whereas PSRS and TSGR attain the best values only on 0 and 4 instances. In terms of the average computation time, one observes that to reach the same objective value, SBMNAS runs faster than PSGR and TSGR.

The small p-values (<0.05, see Table 3, row 'p-value') from the non-parametric Friedman test further confirm the differences between SBMNAS and the two reference algorithms on the set of large random dense instances. This experiment shows the effectiveness of our SBMNAS algorithm on the set of large random dense graphs.

3.2.2 Computational results on the real-life large sparse graphs

Table 4 summarizes the comparative results of the three compared heuristic algorithms PSRS+, TSGR, and SBMNAS on the 25 real-life large sparse graphs. In Table 4, we also include the results of the exact algorithm ExtBBClq. Optimal values proven by ExtBBClq are marked with a '*' symbol.

From Table 4, we can make the following observations. First, ExtBBClq is able to optimally solve 21 (out of 25) instances and attains 23 best-known results. Both SBMNAS and TSGR attains the best-known results for 24 out of the 25 instances including the 21 optimal results proved by ExtBBClq, while PSRS+ reports 22 best-known results (it misses one optimal result). Interestingly, SBMNAS improves the best-known result for one graph (edit-frwiki). In terms of the average objective values, SBMNAS reaches the best results on 23 instances, while both PSRS+ and TSGR yield the best results on 21 cases. In terms of average computation time, SBMNAS runs the fastest on 10 instances, while PSRS+ and TSGR yield the best time on 7 and 8 cases respectively. For the performance indicators, the obtained p-values with the Friedman test (8.32e-2, 0.18 and 7.36e-2 for SBMNAS vs PSRS+, 1.00, 0.41 and 0.18 for SBMNAS vs TSGR) indicate the performances of the three compared heuristic algorithms (SBMNAS, PSRS+ and TSGR) are only marginally different on the 25 real-life large sparse graphs. When comparing SBMNAS with the exact algorithm ExtBBClq, the Friedman test results reveal that, the difference between SBMNAS and ExtBBClq is not statistically significant in terms of best objective values, but SBMNAS runs significantly faster than ExtBBClq in terms of average computation time.

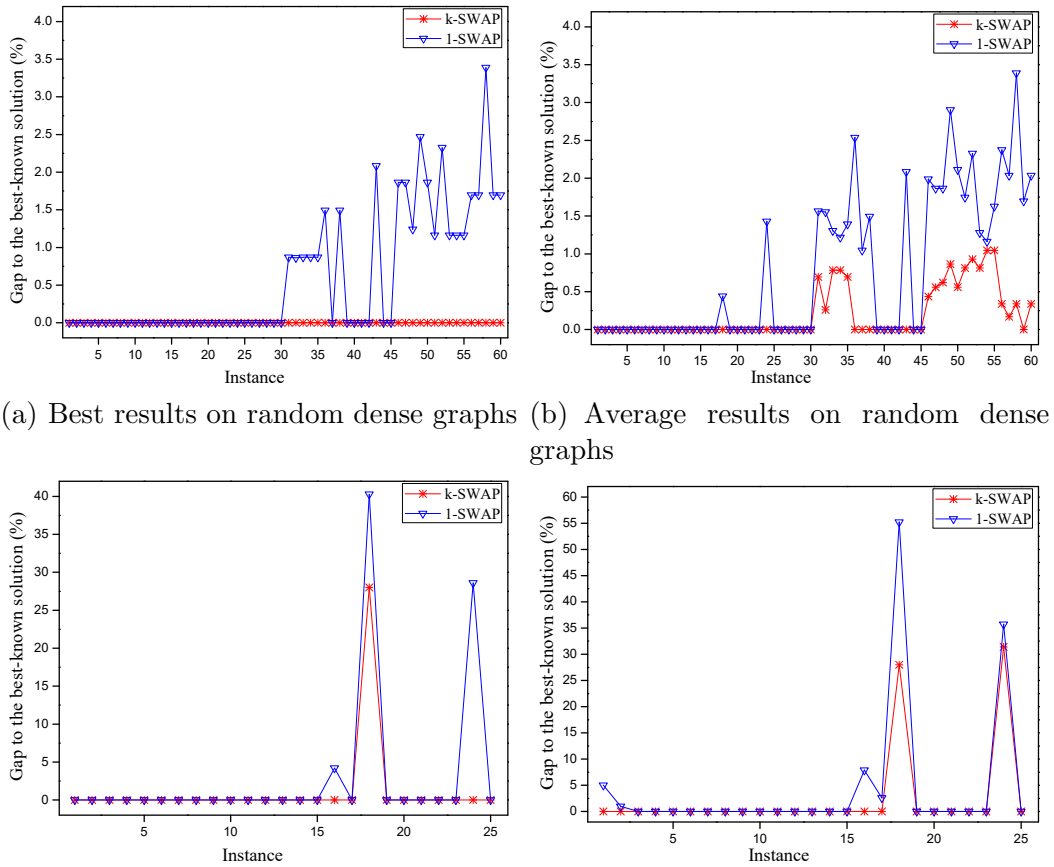
Table 4
Computational results on the 25 real-life large sparse instances from KONECT.

Instance	f_{bk}	PSRS+			TSGR			ExtBBClq		SBMNAS			
		f_{best}	f_{avg}	t_s	f_{best}	f_{avg}	t_s	f_{best}	t_s	f_{best}	f_{avg}	t_s	Gap
actor-movie	8*	**8**	7.4	320.12	**8**	**8**	0.29	**8***	896.31	**8**	**8**	12.51	0.00
bibsonomy-2ui	8*	**8**	**8**	2.51	**8**	**8**	0.06	**8***	5.94	**8**	**8**	0.08	0.00
bookcrossing_full-rating	13*	**13**	**13**	30.01	**13**	**13**	14.42	**13***	246.38	**13**	**13**	0.66	0.00
dblp-author	10*	**10**	**10**	488.12	**10**	**10**	1.06	**10***	12.4	**10**	**10**	2.16	0.00
dbpedia-genre	7*	**7**	**7**	0.25	**7**	**7**	1.05	**7***	3.64	**7**	**7**	0.11	0.00
dbpedia-location	5*	**5**	**5**	0.03	**5**	**5**	0.10	**5***	0.42	**5**	**5**	0.05	0.00
dbpedia-occupation	6*	**6**	**6**	0.06	**6**	**6**	1.01	**6***	0.75	**6**	**6**	0.04	0.00
dbpedia-producer	6*	**6**	**6**	0.02	**6**	**6**	0.02	**6***	0.38	**6**	**6**	0.05	0.00
dbpedia-recordlabel	6*	**6**	**6**	0.03	**6**	**6**	0.06	**6***	11.36	**6**	**6**	0.04	0.00
dbpedia-starring	6*	**6**	**6**	0.50	**6**	**6**	0.02	**6***	3.15	**6**	**6**	0.15	0.00
dbpedia-team	6*	**6**	**6**	6.02	**6**	5.5	144.06	**6***	102.35	**6**	**6**	2.51	0.00
dbpedia-writer	6*	**6**	**6**	0.01	**6**	**6**	0.02	**6***	0.28	**6**	**6**	0.02	0.00
discogs_affiliation	26*	**26**	**26**	6.48	**26**	**26**	4.32	**26***	759.31	**26**	**26**	2.62	0.00
discogs_lgenre	15*	**15**	**15**	2.57	**15**	**15**	4.94	**15***	0.61	**15**	**15**	0.13	0.00
discogs_style	38	**38**	**38**	30.88	**38**	37.5	160.07	**38**	10800	**38**	**38**	3.75	0.00
edit-frwiki	41	33	30	386.68	37	30.3	349.16	40	10800	**48**	**48**	330.03	-17.07
edit-frwiktionary	19*	18	17.2	328.46	**19**	**19**	144.06	**19***	68.62	**19**	**19**	3.03	0.00
flickr-groupmemberships	67	24	22	351.97	**67**	**67**	48.18	36	10800	48	33.1	163.28	28.36
github	12*	**12**	**12**	0.18	**12**	**12**	1.51	**12***	81.32	**12**	**12**	0.13	0.00
moreno_crime	2*	**2**	**2**	0.00	**2**	**2**	0.00	**2***	0.06	**2**	**2**	0.00	0.00
opsahl-collaboration	8*	**8**	**8**	0.06	**8**	**8**	0.01	**8***	0.12	**8**	**8**	0.02	0.00
opsahl-ucforum	5*	**5**	**5**	0.00	**5**	**5**	0.02	**5***	141.32	**5**	**5**	0.00	0.00
stackexchange-stackoverflow	9*	**9**	**9**	0.25	**9**	8.8	99.39	**9***	0.02	**9**	**9**	0.31	0.00
wiki-en-cat	14*	**14**	**14**	5.08	**14**	**14**	1.01	**14***	18.1	**14**	9.6	28.40	0.00
youtube-groupmemberships	12*	**12**	**12**	2.01	**12**	**12**	1.02	**12***	6.3	**12**	**12**	0.90	0.00
#Best		22	21	7/22	24	21	8/22	23	1/22	24	23	10/22	
p-value		8.32e-2	0.18	7.36e-2	1.00	0.41	0.18	0.16	1.24e-4				

4 Analysis

In this section, we analyze several essential ingredients of our proposed SBMNAS algorithm to shed lights on their influences on the performance of the algorithm. We study in particular the *k-SWAP* move operator, the adaptive strategy to explore the neighborhood, and the frequency-based perturbation strategy.

4.1 Impact of the k-SWAP operator



(a) Best results on random dense graphs

(b) Average results on random dense graphs

(c) Best results on real-life large sparse graphs

(d) Average results on real-life large sparse graphs

Fig. 4. Comparisons of SBMNAS with its variant that uses the *1-SWAP* operator.

As described in Section 2.4, SBMNAS introduces a general *k-SWAP* move operator which consists in removing one vertex from the biclique and adding k ($k \geq 1$) vertices to the biclique such that each added vertex is adjacent to all but the removed vertex in the other subset. Compared to the 1-*SWAP* move operator previously applied in studies like [24], our *k-SWAP* move operator

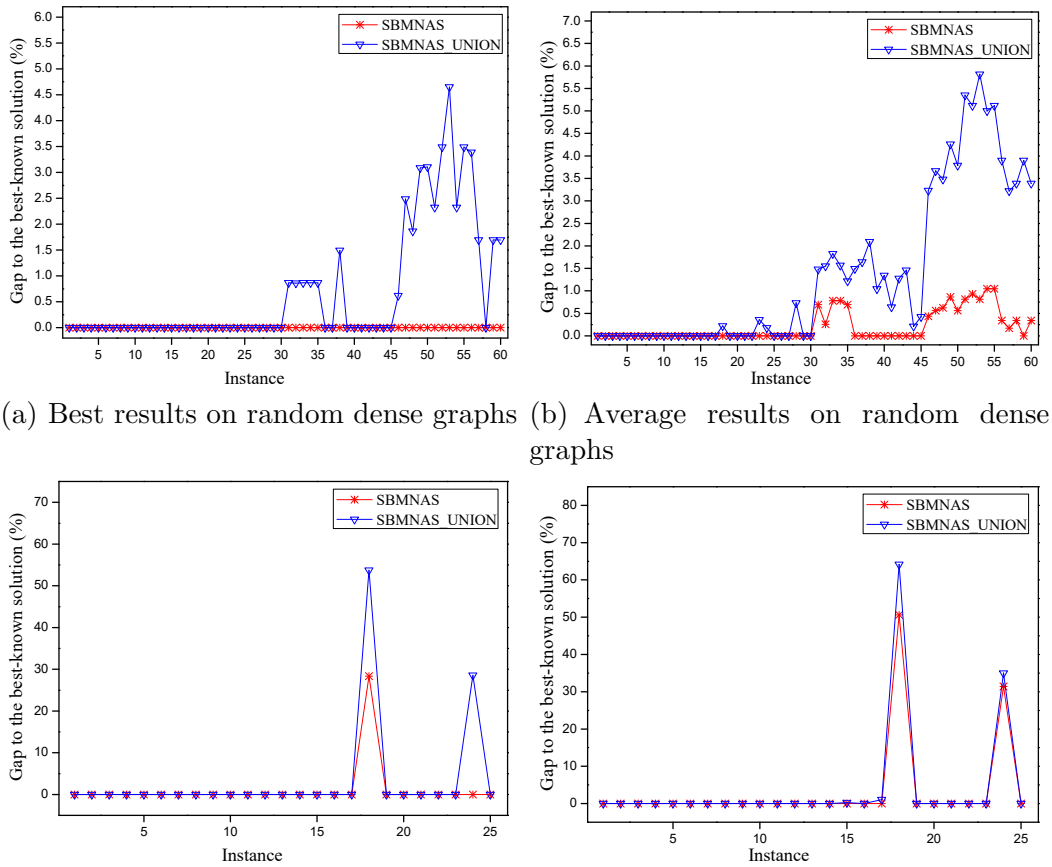
leads to an enlarged neighborhood containing more neighboring solutions, increasing the chance for the algorithm to find solutions of better quality. To show the merit of the k-$SWAP$ move operator, we compare our SBMNAS approach with a variant that relies on the 1-$SWAP$ move operator. To implement this variant, we simply replace the k-$SWAP$ move operator by the 1-$SWAP$ move operator in our SBMNAS approach. Each algorithm is run under the same experimental protocol as described in Section 3.1 to solve each of the 85 instances. We compute for each instance the absolute gap between the best and average results obtained by each algorithm and the best-known results. The best-known results include the results reported by our SBMNAS algorithm in Section 3. The comparative results are summarized in Figure 4.

From Figure 4, one observes that our SBMNAS approach performs significantly better than its variant based on the 1-$SWAP$ move operator. Indeed, for each of the 85 tested instances, SBMNAS with the k-$SWAP$ move operator is able to achieve a smaller or equal gap to the best-known results than the SBMNAS variant with the 1-$SWAP$ move operator in terms of both best and average results. This experiment highlights the contribution of the k-$SWAP$ operator for our SBMNAS algorithm to solve MBBP.

4.2 Effect of the adaptive neighborhood exploration strategy

When several neighborhoods are available, one key issue is how to combine these neighborhoods so as to efficiently explore the search space. In our work, we proposed an adaptive strategy to combine the neighborhoods which favors the selection of a particular neighborhood that produces solutions of better quality. To show the effectiveness of the proposed adaptive neighborhood exploration strategy, we compared it with a popular method to combine the neighborhoods in the literature, i.e., the neighborhood union method which selects the best non-tabu neighboring solution from all considered neighborhoods. This neighborhood union method was shown to be quite effective in the context of multiple neighborhood search for the related maximum weight clique problem [16] and is capable of making an aggressive examination of the search space.

By keeping other ingredients unchanged, we experiment both strategies within our SBMNAS algorithm using the same experimental protocol as described in Section 3.1. We denote the variant with the neighborhood union strategy by SBMNAS_UNION. Figure 5 summarizes the comparative results between these two methods on all 85 instances. From Figure 5, one observes that our proposed adaptive neighborhood exploration method shows an overall better performance than the neighborhood union method for the 85 tested graphs. In terms of the best results obtained by two methods, SBMNAS achieves



(a) Best results on random dense graphs (b) Average results on random dense graphs

(c) Best results on real-life large sparse graphs (d) Average results on real-life large sparse graphs

Fig. 5. Comparisons of SBMNAS with its variant SBMNAS_UNION.

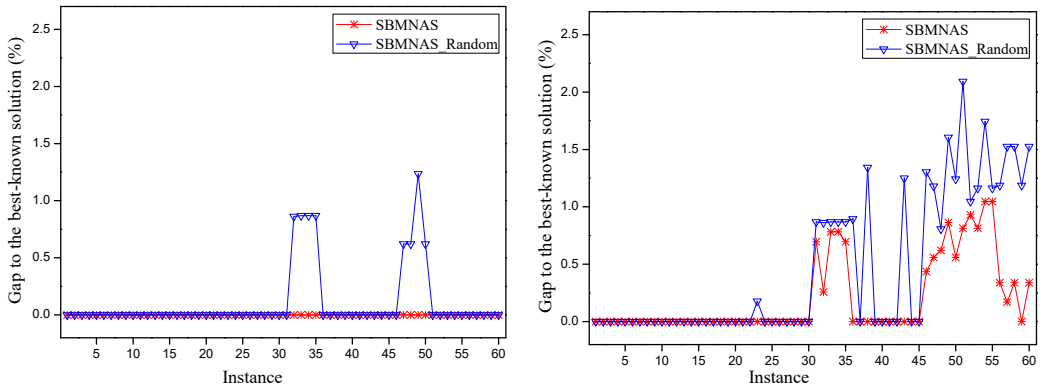
better results than SBMNAS_UNION for 22 instances, while in terms of the average results, SBMNAS reaches better results than SBMNAS_UNION for 27 instances.

Furthermore, the results of the non-parametric Friedman test also demonstrate that SBMNAS performs significantly better than SBMNAS_UNION (p-value = 2.73e-6 in terms of the best results and p-value = 1.18e-9 in terms of the average results). The above observation confirms that our adaptive method for exploring the search space constitutes an important feature to enhance the performance of the local search approach.

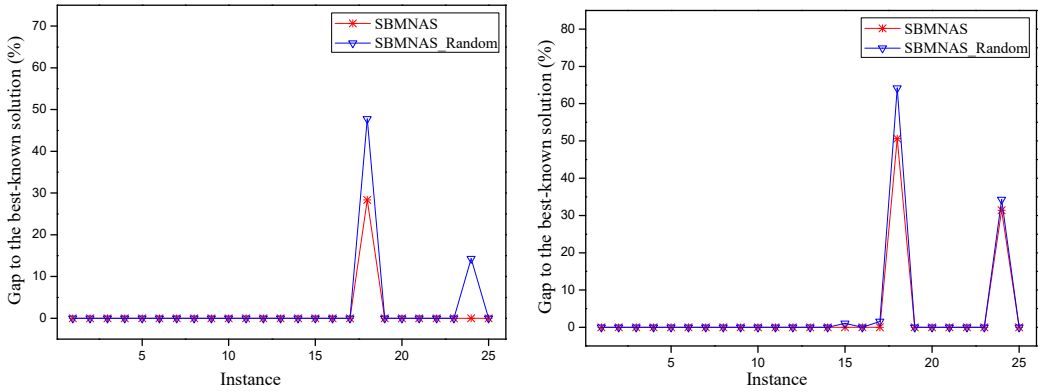
4.3 Usefulness of the frequency-based perturbation

As shown in Section 2.7, our SBMNAS approach applies a frequency-based perturbation strategy to ensure a more global diversification. The frequency-based perturbation strategy operates by dropping several vertices with high move frequency from the solution. To evaluate the effect of the frequency-

based perturbation, we compare it with a random perturbation strategy where the vertices to be removed are randomly selected from the biclique (denoted as SBMNAS_Random). To make a fair comparison, we run both strategies under the same experimental protocol as described in Section 3.1. Figure 6 summarizes the comparative results between these two perturbation strategies. From Figure 6, we observe that SBMNAS performs better than its variant SBMNAS_Random in terms of both best and average results. Indeed, in terms of the best results, SBMNAS outperforms SBMNAS_Random for 10 out of the 85 instances, and matches SBMNAS_Random for the remaining 75 instances. In terms of the average results, SBMNAS outperforms SBMNAS_Random for 28 out of the 85 instances , and matches SBMNAS_Random for the remaining 57 instances. Furthermore, the Friedman test in terms of the best and average results reveals a significant difference (p-value = 1.6e-3 in terms of the best results and p-value = 1.21e-7 in terms of average results) between these two strategies, further confirming that the frequency-based perturbation strategy plays an important role to the performance of the SBMNAS approach.



(a) Best results on random dense graphs (b) Average results on random dense graphs



(c) Best results on real-life large sparse graphs (d) Average results on real-life large sparse graphs

Fig. 6. Comparisons of SBMNAS with its variant SBMNAS_Random.

5 Conclusion

The maximum balanced biclique problem (MBBP) is an NP-hard problem with many practical applications. In this work, we proposed a general swap-based multiple neighborhood adaptive search (SBMNAS) for MBBP. The proposed SBMNAS approach jointly uses three neighborhoods induced by three types of move operators (*ADD*, *DROP* and *k-SWAP*) and applies an adaptive strategy to select the neighborhood according to changing probabilities. The *k-SWAP* move operator which is a new operator proposed in the study for MBBP consists in removing a vertex in one subset of the biclique and adding k vertices to the other subset such that each newly added vertex is connected to all but the removed vertex in the other subset. To escape deep local optima, SBMNAS also combines a guided frequency-based perturbation strategy to ensure a more global diversification.

We evaluated the performance of the proposed SBMNAS approach on two sets of 85 commonly used MBBP instances and show its effectiveness in comparison with two leading reference MBBP algorithms in the literature. The experimental results show that SBMNAS attains the best-known results for all but one graphs and find improved best-known results (i.e., new lower bounds) for 19 graphs, including 18 random dense graphs and 1 real-life large sparse graph from KONECT. Moreover, we carried out additional experiments to demonstrate the key role of the proposed *k-SWAP* move operator and investigate the impact of the adaptive neighborhood exploration strategy as well as the dedicated frequency-based perturbation strategy over the performance of the algorithm.

To go beyond the SBMNAS algorithm, we can consider the population based memetic framework by using SBMNAS as its key local search procedure. Moreover, the proposed adaptive neighborhood exploration strategy is of general nature and can be used in other local search procedures to effectively explore the search space when different neighborhoods are available.

Acknowledgments

We are grateful to the reviewers for their helpful comments. This work is partially supported by the National Natural Science Foundation Program of China [Grant No. 71771099, 71821001,71620107002, 71931005, 71831007].

References

[1] Al-Yamani, A. A., Ramsundar, S., & Pradhan, D. K. (2007). A defect tolerance scheme for nanotechnology circuits. IEEE Transactions on Circuits and Systems I, 54(11), 2402-2409.

[2] Balaprakash, P., Birattari, M., & Stützle, T., (2007). Improvement strategies for the F-race algorithm: sampling design and iterative refinement. In: Bartz-Beielstein, T., Blesa, M.J., Blum, C., Naujoks, B., Roli, A., Rudolph, G., Sampels, M. (Eds.), Hybrid Metaheuristics. Lecture Notes in Computer Science, Vol. 4771. Springer, Heidelberg, Germany, pp. 108-122.

[3] Birattari, M., Stützle, T., Paquete, L., & Varrentrapp, K., (2002). A Racing Algorithm for Configuring Metaheuristics. In Langdon, W.B. et al. (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufman, San Francisco, CA, pp. 11-18.

[4] Cheng, Y., & Church, G. M. (2000). Biclustering of expression data. In International Conference on Intelligent Systems for Molecular Biology, Menlo Park, AAAI Press, pp. 93-103.

[5] Garey M. R.,& Johnson D. S. (1979). Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman & Co Ltd.

[6] Glover, F., & Laguna, M. (1997). Tabu Search, Kluwer Academic. Boston, USA.

[7] Johnson, D. S. (1985). The NP-completeness column: an ongoing guide. Journal of Algorithms, 6(3), 434-451.

[8] Kunegis J. (2013). KONECT – The Koblenz Network Collection. In Proceedings of the 22nd International Conference on World Wide Web (Companion), pp. 1343–1350.

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

[10] Quintana, J. D., Sánchez-Oro, J., & Duarte, A. (2019). Finding balanced bicliques in bipartite graphs using variable neighborhood search. In Proceedings of International Conference on Variable Neighborhood Search. Lecture Notes in Computer Science, volume 11328, pp. 114-124.

[11] Ravi, S. S., & Lloyd, E. L. (1988). The complexity of near-optimal programmable logic array folding. SIAM Journal on Computing, 17(4), 696-710.

[12] Sun, W., Hao, J. K., Lai, X., & Wu, Q. (2018). Adaptive feasible and infeasible tabu search for weighted vertex coloring. Information Sciences, 466, 203-219.

[13] Tahoori, M. B. (2006). Application-independent defect tolerance of reconfigurable nanoarchitectures. ACM Journal on Emerging Technologies in Computing Systems (JETC), 2(3), 197-218.

[14] Wang, W., Hao, J. K., & Wu, Q. (2018). Tabu search with feasible and infeasible searches for equitable coloring. Engineering Applications of Artificial Intelligence, 71, 1-14.

[15] Wang, Y., Cai, S., & Yin, M. (2018). New heuristic approaches for maximum balanced biclique problem. Information Sciences, 432, 362-375.

[16] Wu, Q., Hao, J. K., & Glover, F. (2012). Multi-neighborhood tabu search for the maximum weight clique problem. Annals of Operations Research, 196(1), 611-634.

[17] Wu, Q., & Hao, J. K. (2015). A review on algorithms for maximum clique problems. European Journal of Operational Research, 242(3), 693-709.

[18] Yang, J., Wang, H., Wang, W., & Yu, P. (2003). Enhanced biclustering on expression data. In Third IEEE Symposium on Bioinformatics and Bioengineering, IEEE, pp. 321-327.

[19] Yuan, B., & Li, B. (2011). A low time complexity defect-tolerance algorithm for nanoelectronic crossbar. In International Conference on Information Science and Technology, IEEE, pp. 143-148.

[20] Yuan, B., & Li, B. (2014). A fast extraction algorithm for defect-free subcrossbar in nanoelectronic crossbar. ACM Journal on Emerging Technologies in Computing Systems, 10(3), 25.

[21] Yuan, B., Li, B., Chen, H., & Yao, X. (2014). A new evolutionary algorithm with structure mutation for the maximum balanced biclique problem. IEEE Transactions on Cybernetics, 45(5), 1054-1067.

[22] Zhou, Y., Hao, J. K., & Goëffon, A. (2017). PUSH: A generalized operator for the maximum vertex weight clique problem. European Journal of Operational Research, 257(1), 41-54.

[23] Zhou, Y., Rossi, A., & Hao, J. K. (2018). Towards effective exact methods for the Maximum Balanced Biclique Problem in bipartite graphs. European Journal of Operational Research, 269(3), 834-843.

[24] Zhou, Y., & Hao, J. K. (2019). Tabu search with graph reduction for finding maximum balanced bicliques in bipartite graphs. Engineering Applications of Artificial Intelligence, 77, 86-97.