J. Oper. Res. Soc. China (2014) 2:237–262
DOI 10.1007/s40305-014-0045-z

Combining Clustered Adaptive Multistart and Discrete Dynamic Convexized Method for the Max-Cut Problem

Geng Lin · Wenxing Zhu

Received: 21 February 2014 / Revised: 9 May 2014 / Accepted: 12 May 2014 /
Published online: 17 June 2014

Abstract Given an undirected graph with edge weights, the max-cut problem is to find a partition of the vertices into two subsets, such that the sum of the weights of the edges crossing different subsets is maximized. Heuristics based on auxiliary function can obtain high-quality solutions of the max-cut problem, but suffer high solution cost when instances grow large. In this paper, we combine clustered adaptive multistart and discrete dynamic convexized method to obtain high-quality solutions in a reasonable time. Computational experiments on two sets of benchmark instances from the literature were performed. Numerical results and comparisons with some heuristics based on auxiliary function show that the proposed algorithm is much faster and can obtain better solutions. Comparisons with several state-of-the-science heuristics demonstrate that the proposed algorithm is competitive.

Keywords Max-cut · Local search · Dynamic convexized method · Clustered adaptive multistart

1 Introduction

Given an undirected graph $G = (V, E)$ with vertex set $V = \{1, 2, \cdots, n\}$, edge set $E \subseteq V \times V$. Let $W = (w_{ij})_{n\times n}$ be the symmetric weighted adjacency matrix of the

This research was supported partially by the National Natural Science Foundation of China (Nos. 11226236 and 11301255); the Natural Science Foundation of Fujian Province of China (No. 2012J05007); and the Science and Technology Project of the Education Bureau of Fujian, China (Nos. JA13246 and JK2012037).

G. Lin (✉)
Department of Mathematics, Minjiang University, Fuzhou 350108, China
e-mail: lingeng413@163.com

W. Zhu
Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou 350002, China

graph G with $w_{ij} \neq 0$ for $\{i,j\} \in E$ and $w_{ij} = 0$ otherwise. The max-cut problem consists in finding a partition of the set V into two disjoint subsets $(S, \overline{S})$ such that the sum of the weights of the edges between S and $\overline{S}$ is maximized. The sum of the weights of the edges between S and $\overline{S}$ is called the cut value of partition $(S, \overline{S})$, and given by

$$w(S, \overline{S}) = \sum_{i \in S, j \in \overline{S}} w_{ij}.$$

Let $x_i \in \{1, -1\}, i = 1, 2, \cdots, n$, be variables such that $x_i = 1$ if $i \in S$, and $x_i = -1$ otherwise. Let $L = \frac{1}{4}(\mathrm{Diag}(\mathrm{W}_e) - W)$, where $e \in \mathbb{R}^n$ is a column vector with all components ones, and $\{mathrmDiag(\mathrm{W}_e)$ is a diagonal matrix with elements of the vector (W_e) being the diagonal entries. Then the max-cut problem can be formulated as the following integer quadratic program [24, 25]:

$$(MC)\begin{cases} \max & f(x) = x^{\mathrm{T}} L x \\ \text{s.t.} & x_i \in \{1, -1\}, \quad i = 1, \cdots, n. \end{cases}$$

The max-cut problem is one of Karp's [17] original NP-complete problems, and it is NP-complete even for un-weighted graph [10]. Besides its theoretical significance, the max-cut problem arises in a variety of real-world applications, including very large scale integration design [6, 21, 30], statistical physics [2, 34], scientific computing [16], network optimization, sports team scheduling [7].

The max-cut problem has recently gained much attention because of its theoretical significance and wide range of applications. A number of methods have been developed to solve this problem. These solution methods can be categorized into three classes: exact approaches [11, 19, 31], approximate approaches [12, 32], and heuristic approaches [1, 3–5, 8, 14, 18, 20, 22–26, 28, 33, 37].

Rendl [31] proposed a branch-and-bound framework to solve the max-cut problem to optimality. At each node of the branch-and-bound tree, the bound was calculated using a dynamic version of the bundle method that solved a basic semidefinite relaxation of the max-cut problem strengthened by triangle inequalities. Their experiments showed that their proposed exact algorithm can solve max-cut problems up to $n = 100$ in a reasonable time. Recently, Ghaddar [11] developed a branch-and-cut algorithm based on a semidefinite programming relaxation of the minimum k-partition problem, whose special case with $k = 2$ is max-cut problem. Inside the branch-and-cut algorithm, they used positive semidefinite relaxations that were further tightened using polyhedral results, and feasible solutions were obtained by an iterative clustering heuristic. Exact solution approaches can find optimal solutions of the max-cut problem, however, since the problem is NP-complete, the practical usefulness of these algorithms is limited to fairly small instances.

The max-cut problem is known to be APX-complete [29]. It does not exist a polynomial time approximation scheme (PTAS) [36] unless P = NP. Some approximation algorithms have been proposed to get approximate solutions of the max-cut problem. In 1976, Sahni and Gonzales [32] proposed a $\frac{1}{2}$-approximation algorithm. Let $X = xx^{\mathrm{T}}$, the max-cut problem can be relaxed to the following semidefinite programming problem:

$$(SDP)\begin{cases}\max & L\cdot X\\ \text{s.t.} & \mathrm{diag(X)}=e,\\ & X\succcurlyeq 0,\end{cases}$$

where $\mathrm{diag(X)}=(\mathrm{X}_{11},\cdots,\mathrm{X}_{\mathrm{nn}})^{\mathrm{T}},\mathrm{X}_{\mathrm{ii}},\mathrm{i}=1,2,\cdots,\mathrm{n}$, are the diagonal entries of the matrix X. For nonnegative weighted graphs, Goemans and Williams [12] presented a 0.878 56-approximation algorithm by solving the semidefinite programming relaxation of the max-cut problem, and using randomized rounding to obtain a solution of problem (MC).

Because of the NP-hardness of the max-cut problem, heuristic approach plays a crucial role for the solution on large graphs. A number of heuristic algorithms, based on different ideas, were proposed recently in the literature. Most of them are based on the semidefinite programming relaxation (SDP) of the max-cut problem. Burer et al. [5] proposed a rank-2 heuristic (CirCut) for the max-cut problem, which performed better in practice than the method in [12] in terms of solution quality. The authors in [4, 14, 26] used different methods to deal with the semidefinite programming problem (SDP) and proposed different algorithms for the max-cut problem. Some well-known metaheuristics such as variable neighborhood search [8], path-relinking [8], scatter search [20], grasp [37], tabu search [1], breakout local search [3], were used to solve the max-cut problem effectively.

The other methods deal with the max-cut problem based on auxiliary functions [22–25]. In 2008, Ling et al. [24] presented a discrete filled function, whose parameters don't need to be adjusted, for the max-cut problem, and proposed a discrete filled function algorithm for approximate global solutions of the problem. In 2009, they [25] presented a new discrete filled function for the max-cut problem, and employed a continuation optimization algorithm to find local solutions of a continuous relaxation of the max-cut problem, then global search was performed by minimizing the proposed new filled function. More recently, Ling et al. [23] showed that the max-cut problem is equivalent to the following discrete optimization problem:

$$(\mathrm{MMC})\begin{cases}\min & \mathrm{h(x)=x^{T}Wx}\\ \text{s.t.} & \mathrm{x_i}\in\{1,-1\},\quad \mathrm{i}=1,\cdots,\mathrm{n}.\end{cases}$$

A new filled function for the problem (MMC) was presented in [23] as follows:

$$H(x;x^*;\alpha,\beta)=\begin{cases}-[h(x)-h(x^*)]+\dfrac{\beta}{a+\|x-x^*\|_p}, & h(x)\geqslant h(x^*),\\ \alpha[h(x^*)-h(x)]+\dfrac{\beta}{a+\|x-x^*\|_p}, & h(x)\leqslant h(x^*),\end{cases}$$

where $a>0$ and $1\leqslant p<\infty$ are constants, $\alpha>0$ and $\beta>0$ are two adjustable parameters. Then a discrete filled function algorithm was proposed to solve the max-cut problem. Numerical results and comparisons with [5, 8] were reported to show that the proposed algorithm is efficient. In 2012, Lin et al [22] proposed a discrete dynamic convexized method for solving the max-cut problem. Experiments were conducted on three sets of standard test instances from the literature. It showed that the proposed algorithm is effective for the three sets of standard test instances.

The heuristics based on auxiliary functions can escape successfully from previously converged discrete local optima and find high-quality solutions. However, the filled function method may spend relative more time on large scale graphs due to the number of the local optima of the auxiliary function and the times of minimization (or maximization) of the auxiliary function from different initial points [22]. The discrete dynamic convexized method used random multistart approach, i.e., each time it restarted from random starting point. The number of runs required to achieve good solutions grows and the time of maximization of the auxiliary function increases with problem size. Clustered adaptive multistart (CAMS) approach [13] reduces the problem size by clustering, and generates new starting points from previously found local optima, then the efficiency of the search is rapidly improved.

In this paper, we focus primarily on combining clustered adaptive multistart approach CAMS [13] and the discrete dynamic convexized algorithm (DCM) [22] for the max-cut problem to find high-quality solutions in acceptable computing times on large scale graphs. In the work, we combine clustered adaptive multistart CAMS [13] and discrete dynamic convexized algorithm DCM into an algorithm (CAMS_DCM). The new algorithm takes advantage of both CAMS and DCM, so that it can be capable of solving large scale problem, and rapidly obtains high-quality solutions. Our main contributions can be summarized as follows:

- Reducing the problem size by dynamically clustering based on previously found elite solutions, such that a smaller, more easily solvable problem instance is obtained.
- Maximizing the discrete dynamic convexized function from new starting points, which are based on previously found elite solutions, in order to escape from the current best local maximizer.

The remainder of this paper is organized as follows. In Sect. 2, we review the discrete dynamic convexized algorithm DCM for the max-cut problem in [22]. Section 3 describes the clustering method for obtaining a smaller graph. The proposed algorithm CAMS_DCM combines the clustered adaptive multistart and discrete dynamic convexized method, which is given in Sect. 4. Section 5 provides a computational evaluation of the proposed algorithm on benchmark instances from the literature, and the results are compared with other existing algorithms in the literature. Concluding remarks are given in Sect. 6.

2 Discrete Dynamic Convexized Method for the Max-Cut Problem

This Section briefly reviews the discrete dynamic convexized method [22] for the max-cut problem in order to explain the proposed hybrid algorithm clearly and make the paper self-contained.

Local search heuristics are effective for solving NP-hard combinatorial optimization problems, but early get struck in local optima. Dynamic convexized method is one of the effective approaches to help the algorithm find better local optima. It was originally proposed for solving nonlinear global optimization and nonlinear integer programming problems [39–41]. This method uses an auxiliary

function of the original problem, and then uses a local search method to minimize the auxiliary function in order to escape from the current best local optima and thus obtains better results. Recently, this method has been successfully applied to some optimization problems, such as nonconvex mixed integer nonlinear programming [42], max-k-cut problem [22, 43], etc.

2.1 Definitions and Local Search Method MCFM

Now, we introduce some following definitions used in [22]. A neighborhood $N(x)$ of a given solution $x \in \{1,-1\}^n$ is obtained by moving a vertex i from its original subset to the complement subset, i.e.,

$$N(x) = \{y \in \{1,-1\}^n : \|y - x\|_1 \leqslant 2\}.$$

Therefore, for any $x = (x_1, \cdots, x_n) \in \{1,-1\}^n$, the size of the neighborhood is $n+1$.

Definition 2.1 [22] A solution $y \in \{1,-1\}^n$ is called a discrete local maximizer of the problem (MC), if $f(x) \leqslant f(y)$, for all $x \in N(y)$.

Definition 2.2 [22] A solution $y \in \{1,-1\}^n$ is called a discrete global maximizer of the problem (MC), if $f(x) \leqslant f(y)$, for all $x \in \{1,-1\}^n$.

An iterative improvement local search method (MCFM) was proposed in [22] to find discrete local maximizers of problem (MC). It is a simple modification of the Fiduccia-Mattheyses heuristic (FM) [9] for circuit partitioning.

Defining the gain $gain(i,x)$ of a vertex i as the objective value of the problem would increase by moving the vertex i from its current subset to the complement subset, which is as follows:

$$\begin{aligned} gain(i,x) &= f(x_1, \cdots, x_{i-1}, -x_i, x_{i+1}, \cdots, x_n) - f(x_1, \cdots, x_i, \cdots, x_n) \\ &= \begin{cases} \sum\limits_{\{i,j\} \in E, j \in S} w_{ij} - \sum\limits_{\{i,j\} \in E, j \in \overline{S}} w_{ij} & i \in S; \\ \sum\limits_{\{i,j\} \in E, j \in \overline{S}} w_{ij} - \sum\limits_{\{i,j\} \in E, j \in S} w_{ij} & i \in \overline{S}. \end{cases} \end{aligned}$$

MCFM proceeds in a series of passes. At the beginning of a pass, all vertices are free to be moved. MCFM iteratively moves a free vertex with the highest gain (but not necessarily positive). After a move is carried out, the moved vertex is not allowed to move again during that pass, and the gains of adjacent vertices are updated accordingly. The moving process is iterated until p_1 vertices have been moved, then the best partition solution during the pass is adopted as the starting solution of the next pass. The algorithm terminates when a pass fails to improve solution quality.

Denote the set of free vertices as UNLOCK, and let p_1 be the number of vertices which are allowed to be moved in a pass. The pseudo-code of the MCFM is given in Algorithm 1. One pass of MCFM is bounded by $O(n^2)$.

Algorithm 1 MCFM (G, x)

Input: graph $G = (V, E)$, initial solution $x = (x_1, x_2, \cdots, x_n) \in \{1, -1\}^n$, positive number $p_1 < n$.

Output: discrete local maximizer $x_{\max}$.

repeat
  Set $UNLOCK = \{1, 2, \cdots, n\}$, $x_{\max} := x$, and $x_{\text{ini}} := x$. Calculate $gain(i, x)$, for all $i \in UNLOCK$.
  repeat
    Let $gain(j, x) = \max\{gain(i, x) : i \in UNLOCK\}$. Set $x' := (x_1, x_2, \cdots, -x_j, \cdots, x_n)$, and $UNLOCK := UNLOCK \backslash \{j\}$.
    if $f(x_{\max}) < f(x')$ **then**
      Set $x_{\max} := x'$.
    end if
    if $\{i, j\} \in E$, $i \in UNLOCK$, and $x_i \neq x_j$ **then**
      Set $gain(i, x') := gain(i, x) + 2w_{ij}$.
    end if
    if $\{i, j\} \in E$, $i \in UNLOCK$, and $x_i = x_j$ **then**
      Set $gain(i, x') := gain(i, x) - 2w_{ij}$.
    end if
    if $\{i, j\} \notin E$, and $i \in UNLOCK$ **then**
      Set $gain(i, x') := gain(i, x)$.
    end if
    Set $x := x'$.
  until $|UNLOCK| \leqslant n - p_1$
  Set $x := x_{\max}$.
until $f(x_{\max}) = f(x_{\text{ini}})$
return $x_{\max}$ as a discrete local maximizer of problem (MC).

2.2 Auxiliary Function and Discrete Dynamic Convexized Method

Let x^* be the current best discrete local maximizer of problem (MC). Following auxiliary function is constructed:

$$T(x, k) = \begin{cases} f(x) - k\|x - x^*\|_1, & \text{if } f(x) \leqslant f(x^*), \\ f(x), & \text{if } f(x) > f(x^*), \end{cases} \tag{2.1}$$

where k is a nonnegative parameter, $\|\cdot\|_1$ designates the 1-norm. Then the following nonlinear integer programming problem (AMC) is constructed:

$$(AMC) \begin{cases} \max & T(x, k) \\ \text{s.t.} & x_i \in \{1, -1\}, \quad i = 1, \cdots, n. \end{cases}$$

It have been showed that if x^* is not a discrete global maximizer of problem (MC), then problems (MC) and (AMC) have the same discrete global maximizers and global maximal values.

When using the local search algorithm MCFM to maximize the auxiliary function $T(x, k)$, the gain $gain(i, x)$ of a vertex i is redefined as

$$\text{gain}(\text{i}, \text{x}) = \text{T}(\text{x}', \text{k}) - \text{T}(\text{x}, \text{k}),$$

where $x = (x_1, x_2, \cdots, x_n), x' = (x_1, \cdots, x_{i-1}, -x_i, x_{i+1}, \cdots, x_n), i = 1, \cdots, n$.

[22] showed that if k was large enough, then maximizing $T(x, k)$ from any initial solution, the maximization sequence will converge to the current best discrete local maximizer x^*, or converge to a better solution.

Theorem 2.1 [22] For any $x \in A = \{x \in \{1, -1\}^n : f(x) \leqslant f(x^*)\}$, $x \neq x^*$, let

$$A(x) = \max\{0, \min_{z \in N(x), \|z - x^*\|_1 < \|x - x^*\|_1} \frac{1}{2}(f(x) - f(z))\}.$$

If $k > A(x)$, then starting from any initial solution in $\{1, -1\}^n$ to maximize $T(x, k)$ using the algorithm MCFM will not converge to x. Especially, for all $x' \in A = \{x \in \{1, -1\}^n : f(x) \leqslant f(x^*)\}$, $x' \neq x^*$, if

$$k > \max_{x \in \{1, -1\}^n} A(x), \tag{2.2}$$

then starting from any initial solution in $\{1, -1\}^n$ to maximize $T(x, k)$ using the algorithm MCFM will not converge to x'.

Theorem 2.1 suggests that the value of k should be large enough in order to escape from the current best discrete local maximizer. However, too large value of k may make the search converge to the current best discrete local maximizer x^* quickly. So, an updating scheme is developed to identify a suitable value of k.

The general idea of the discrete dynamic convexized method is as follows: At the beginning, initial $k = 0$, find a solution x' by local search method MCFM starting from a random solution. If $x' \neq x^*$, and $f(x') \leqslant f(x^*)$, by Theorem 2.1, the value of k does not satisfy the inequality (2.2). Then the value of k increases, and applying MCFM to maximize $T(x, k)$ starting from x'. If the obtained solution x'' satisfies $x'' \neq x^*$ and $f(x'') \leqslant f(x^*)$. It implies that the value of k is still too small, then the value of k increases, and applying MCFM to maximize $T(x, k)$ from x'' again, till the maximization sequence converges to x^* or a better solution.

3 Clustering Method

The discrete dynamic convexized method presented in [22] can obtain a better discrete local maximizer of problem (MC) by applying the local search method MCFM. Finally, an approximate global maximizer of (MC) can be obtained. However, like the filled function method [23–25] for the max-cut problem, the computing time may be relatively high for large graphs. Hence, it is necessary to develop an effective way to reduce the computing time for large graphs.

One effective way to reduce the computing time for large scale optimization problems is to reduce the problem size. Hagen and Kahng [13] proposed a clustered adaptive multistart (CAMS) methodology for circuit partitioning. Their Numerical results showed that the CAMS method was surprisingly fast and stable for large benchmark instances. In this paper, we adopt the CAMS to construct a clustering,

Vertex / Solution	1	2	3	4	5	6	7	8	9	10
x^1	1	−1	-1	1	-1	1	-1	-1	1	-1
x^2	1	-1	1	1	-1	1	1	-1	-1	1
x^3	-1	1	-1	-1	1	1	-1	1	-1	-1
x^4	1	1	-1	1	1	-1	-1	1	1	-1

a. 4 different solutions

Vertex / Solution	2	3	5	7	8	10	1	4	6	9
x^1	-1	-1	-1	-1	-1	-1	1	1	1	1
x^2	-1	1	-1	1	-1	1	1	1	1	-1
x^3	1	-1	1	-1	1	-1	-1	-1	1	-1
x^4	1	-1	1	-1	1	-1	1	1	-1	1

b. After x^1 is sorted

Vertex / Solution	2	5	8	9	3	7	10	1	4	6
x^1	-1	-1	-1	1	-1	-1	-1	1	1	1
x^2	-1	-1	-1	-1	1	1	1	1	1	1
x^3	1	1	1	-1	-1	-1	-1	-1	-1	1
x^4	1	1	1	1	-1	-1	-1	1	1	-1

c. After x^2 is sorted

Vertex / Solution	9	3	7	10	1	4	2	5	8	6
x^1	1	-1	-1	-1	1	1	-1	-1	-1	1
x^2	-1	1	1	1	1	1	-1	-1	-1	1
x^3	-1	-1	-1	-1	-1	-1	1	1	1	1
x^4	1	-1	-1	-1	1	1	1	1	1	-1

d. After x^3 is sorted

Vertex / Solution	3	7	10	6	9	1	4	2	5	8
x^1	-1	-1	-1	1	1	1	1	-1	-1	-1
x^2	1	1	1	1	-1	1	1	-1	-1	-1
x^3	-1	-1	-1	1	-1	-1	-1	1	1	1
x^4	-1	-1	-1	-1	1	1	1	1	1	1

e. After x^4 is sorted

Fig. 1 A simple example for finding clusters

which groups the vertices in V into disjoint subsets or clusters. Then we contract the vertices of each cluster into a single vertex, such that a smaller graph is generated.

Let $x^i, i = 1, \cdots, t$, be t different elite solutions, which can be initially generated by applying local search method MCFM to problem (MC) starting from t different solutions, respectively. Our clustering method (CAM) constructs a clustered graph $G' = (V', E')$ from t previous elite solutions of the original graph G. It groups the vertices of G, which are partitioned in the same subset in all of the t solutions, into a single condensed vertex in G'. Note that, for any solution vector $x \in \{1, -1\}^n$, $x_i = 1$ means that $i \in S$, and $i \in \overline{S}$ otherwise. Then, finding the clusters is to identify vertices which have identical components in the given t solution vectors. The clusters can be easily found by the bucket or radix sort on the t solution vectors.

We take a simple example to show how to find clusters by the radix sort. Let $V = \{1, 2, \cdots, 10\}$ be the set of vertices, let $x^1 = (1, -1, -1, 1, -1, 1, -1, -1, 1, -1)$, $x^2 = (1, -1, 1, 1, -1, 1, 1, -1, -1, 1)$, $x^3 = (-1, 1, -1, -1, 1, 1, -1, 1, -1, -1)$, $x^4 = (1, 1, -1, 1, 1, -1, -1, 1, 1, -1)$ be 4 different elite solutions. An illustration of the radix sort on the solution vectors is provided in Fig. 1.

From Fig. 1e, one can observe that the data under the columns "3", "7", and "10" are equal, and the same is between the columns "2", "5", and "8". And the data under the columns "1" and "4" are equal too. It means that vertices sets $\{3, 7, 10\}$, $\{1, 4\}$, and $\{2, 5, 8\}$ are partitioned in the same subset in 4 solutions. Then the proposed clustering algorithm groups vertices sets $\{3, 7, 10\}$, $\{1, 4\}$ and $\{2, 5, 8\}$ into condensed vertices in G', respectively.

From the idea of the CAM, we can make the following observation.

Observation 3.1

(1) Applying CAM on the same elite solution sets will construct identical clustered graph.
(2) The better the quality of solutions in the elite solution set, the more easily solved the clustered graph.
(3) If the solutions in the elite solution set are too similar, it is hard to find better solutions by applying MCFM on the clustered graph.

Vertices to be clustered together are chosen based on the previous found elite solutions. However, if the elite solutions are too similar to each other, and the clusters found by radix sort are too large, the number of the vertices of the clustered graph G' is too small. Then it is hard to find better solution by applying local search algorithm MCFM on the clustered graph G'. So, if the number of vertices in a cluster is bigger than s, which is a parameter of CAM, then CAM randomly decomposes the cluster into some smaller subclusters until the number of vertices in each subcluster is smaller than or equal to s. Algorithm 3 shows the clustering method CAM procedure for the max-cut problem.

Algorithm 2 CAM$(G = (V, E), M, s)$

Input: graph $G = (V, E)$, elite solution set $M = \{x^1, x^2, \cdots, x^t\}, s > 0$.
Output: a clustered graph $G' = (V', E')$.

1: **for** $i = 1, \cdots, t$ **do**
2: sorting x^i by radix sort.
3: **end for**
4: finding vertices which have identical components in the t solutions in M.
5: **while** there exists a cluster in which the number of vertices is bigger than s **do**
6: decomposing the cluster into some smaller subclusters until the number of vertices in each subcluster is smaller than or equal to s.
7: **end while**
8: group all found clusters into a single condensed vertex, respectively. Denote the set of condensed vertices as V'.
9: Let E' be the set of edges over V' that is induced by E.
10: **return** $G' = (V', E')$.

The time complexity of the algorithm CAM can be analyzed as follows. It takes $O(tn)$ times to sort t elite solutions by the radix sort. The decomposition of clusters into smaller subclusters need $O(n)$. In time $O(m)$, we can construct G' and E'. Therefore, the total running time of the clustering method CAM is $O(m)$.

4 The Proposed Algorithm

4.1 Discrete Dynamic Convexized Method on Clustered Graph

Let $M = \{x^1, x^2, \cdots, x^t\}$ be the elite solution set, and $x^b = \mathrm{Argmax}_{\mathrm{x^i}\in\mathrm{M}}\{\mathrm{f(x^i)}\}$. Denote $V_{1'}, V_{2'}, \cdots, V_{n'}$ as the clusters found by the CAM from t elite solutions in M. Let $V' = \{1', 2', \cdots, n'\}$. Then, we have $V_{i'} \subseteq V$, for all $i' \in V'$, and $\bigcup_{i'\in V'} V_{i'} = V$. The vertices in each cluster $V_{i'}$ are grouped into a condensed vertex i' in the clustered graph $G' = (V', E')$. Denote n' and m' be the number of vertices and edges in the clustered graph G' constructed by algorithm CAM, respectively.

Let $L' = \frac{1}{4}(\mathrm{Diag(W'e')} - \mathrm{W'})$, where W' is the symmetric weighted adjacency matrix of the graph G', $e' \in R^{n'}$ is a column vector with all components ones, and Diag $(W'e')$ is a diagonal matrix with elements of the vector $W'e'$ being the diagonal entries. Then the max-cut problem on the clustered graph G' can be formulated as follows:

$$(\mathrm{CMC})\begin{cases}\max & \mathrm{g(y) = y^T L' y}\\ \mathrm{s.t.} & \mathrm{y_i \in \{1,-1\},\ \ i = 1,\cdots,n'.}\end{cases}$$

Let $y^* \in \{1,-1\}^{n'}$ be a solution of problem (CMC). Like [22], the auxiliary function (CMC) is constructed as follows:

$$T'(y,k') = \begin{cases} g(y) - k'\|y - y^*\|_1, & \text{if } \ \mathrm{g(y) \leqslant g(y^*)};\\ g(y), & \text{if } \ \mathrm{g(y) > g(y^*)}, \end{cases} \tag{4.1}$$

where k' is a nonnegative parameter, $\|\cdot\|_1$ designates the 1-norm. Then the following auxiliary problem is constructed:

$$(\mathrm{ACMC})\begin{cases}\max & \mathrm{T'(y,k')}\\ \mathrm{s.t.} & \mathrm{y_i \in \{1,-1\},\ \ i = 1,\cdots,n'.}\end{cases}$$

The following result follows from Theorem 2.1 [22].

Corollary 1 For any $y \in A = \{y \in \{1,-1\}^{n'} : g(y) \leqslant g(y^b)\}$, $y \neq y^b$, let

$$A'(y) = \max\{0, \min_{z\in N(y), \|z-y^b\|_1 < \|y-y^b\|_1} \frac{1}{2}(g(x) - g(z))\}.$$

If $k' > A'(x)$, then starting from any initial solution in $\{1,-1\}^{n'}$ to maximize $T'(y,k')$ using the algorithm MCFM will not converge to y. Especially, for all $y' \in A = \{y \in \{1,-1\}^{n'} : g(y) \leqslant g(y^b)\}$, $y' \neq y^b$, if

$$k' > \max_{y\in\{1,-1\}^{n'}} A'(y), \tag{4.2}$$

then starting from any initial solution in $\{1,-1\}^{n'}$ to maximize $T'(y,k')$ using the algorithm MCFM will not converge to y'.

Like [22], the discrete dynamic convexized method DCM for problem (CMC) is given by Algorithm 3.

Algorithm 3 DCM$(G', y^*, \delta_{k'}, N_1)$

Input: graph $G' = (V', E')$, a solution $y^* \in \{1, -1\}^{n'}$, a positive number $\delta_{k'}$, the tolerance parameter N_1 for terminating the algorithm.

Output: a solution y'.

1: Set $N = 0$, and construct a function $T'(y, k')$ with k', y^*.
2: **repeat**
3: Set $k' = 0$, and $N = N + 1$.
4: Generate randomly a solution $y \in \{1, -1\}^{n'}$.
5: **repeat**
6: Search for a discrete local maximizer of problem (ACMC) using algorithm MCFM starting from y. When using the algorithm MCFM to maximize $T'(y, k')$ with $k' = 0$, we set $p_1 = \frac{n'}{3}$. When using the algorithm MCFM to maximize $T'(y, k')$ with $k' > 0$, we set $p_1 = n'$. Suppose that y' is an obtained local maximizer.
7: Set $k' := k' + \delta_{k'}$, $y := y'$
8: **until** $y' = y^*$ or $g(y') > g(y^*)$
9: **if** $f(y') > f(y^*)$ **then**
10: Set $y^* = y'$, and construct a function $T'(y, k')$ with k', y^*.
11: **end if**
12: **until** $N > N_1$
13: $y' = y^*$.
14: **return** y'.

In order to describe the relationship between the solutions of the clustered graph and the original graph clearly, we introduce the following two definitions.

Definition 4.1 Suppose that the vertices in $V_{i'}, i' = 1', \cdots, n'$ are grouped into a condensed single vertex i' in the clustered graph V', respectively. For a given solution $x = (x_1, \cdots, x_n) \in \{1, -1\}^n$, its clustered solution $y = (y_{1'}, \cdots, y_{n'}) \in \{1, -1\}^{n'}$ is defined as $y_{i'} = x_j, i' \in V'$, where $j \in V_{i'}$.

Definition 4.2 Suppose that the vertices in $V_{i'}, i' = 1', \cdots, n'$ are grouped into a condensed single vertex i' in the clustered graph V', respectively. For a given solution $y = (y_{1'}, \cdots, y_{n'}) \in \{1, -1\}^{n'}$, its projected solution $x = (x_1, \cdots, x_n) \in \{1, -1\}^n$ is defined as $x_i = y_{j'}, i \in V$, where $i \in V_{j'}$.

We take the example mentioned in Sect. 3 to illustrate the definitions of clustered solution and projected solution. From Fig. 1e, we have $V'_{1'} = \{3, 7, 10\}$, $V'_{2'} = \{6\}$, $V'_{3'} = \{9\}$, $V'_{4'} = \{1, 4\}$, $V'_{5'} = \{2, 5, 8\}$, and $V' = \{1', 2', 3', 4', 5'\}$. Then by Definitions 4.1 and 4.2, the clustered solutions of $x^1 = (1, -1, -1, 1, -1, 1, -1, -1, 1, -1)$ and $x^2 = (1, -1, 1, 1, -1, 1, 1, -1, -1, 1)$ are $(-1, 1, 1, 1, -1)$, and $(1, 1, -1, 1, -1)$, respectively. Suppose that $y = (1, 1, -1, -1, 1)$ is a solution of the clustered graph, then its projected solution is $(-1, 1, 1, -1, 1, 1, 1, 1, -1, 1)$.

By the construction of G', and Definitions 4.1 and 4.2, it is easy to have following proposition.

Proposition 4.1 if $x \in \{1,-1\}^n$ is the projected solution of $y \in \{1,-1\}^{n'}$, or $y \in \{1,-1\}^{n'}$ is the clustered solution of $x \in \{1,-1\}^n$, then $f(x) = g(y)$.

Suppose that $G' = (V', E')$ is constructed by CAM from the elite solution set $M = \{x^1, \cdots, x^t\}$, $x^b = \text{Argmax}_{\text{x}^\text{i} \in \text{M}}\{\text{f}(\text{x}^\text{i})\}$, and $y^b \in \{1,-1\}^{n'}$ such that y^b is the clustered solution of x^b. We construct a function $T'(y, k')$ with k', y^b. By Corollary 1, if k' satisfies the inequality (4.2), then starting from any initial solution in $\{1,-1\}^{n'}$ to maximize $T'(y, k')$ using the algorithm MCFM will converge to (1) y^b; or (2) a better solution $y^{b'}$ such that $g(y^{b'}) > g(y^b)$.

If a better solution $y^{b'}$ is found by DCM on the clustered graph, then we can get its projected solution $x^{b'}$ by Definition 4.1. Since $f(x^{b'}) = g(x^{b'}) > g(y^b)$, by Proposition 4.1, $g(y^b) = f(x^b)$, we have $f(x^{b'}) > f(x^b)$. Then we find a better solution than x^b. Therefore, we can find better partition of V by applying DCM on clustered graph obtained from the previous partitions of V.

4.2 Algorithm

In this subsection, a hybrid algorithm, called CAM_DCM, is proposed for solving the max-cut problem. The pseudo-code of CAM_DCM is given in Algorithm 4. It uses clustering method CAM to construct smaller graphs from the previous elite solution set M ($|M| = t$). From the idea of CAM, we have that the same elite solution sets will construct identical clustered graph. In order to generate different clustered graphs in different iterations, CAM_DCM maintains a elite solution set P such that $|P| > t$.

Denote $x^* \in \{1,-1\}^n$ as the current best solution found. In each iteration, CAM_DCM randomly selects t solutions from P to construct clustered graphs by CAM, and denotes the best solution in M and its clustered solution as x^b and y^b, respectively (Algorithm 4, line 6). Then, the local search algorithm MCFM is used to find a discrete local maximizer of problem (CMC) starting from a randomly generated solution $y^0 \in \{1,-1\}^{n'}$. Suppose that y^l is an obtained local maximizer of problem (CMC). We construct auxiliary function $T'(y, k')$ with k', y^b, and maximize $T'(y, k')$ by DCM from y^l (Algorithm 4, line 7). By Corollary 1, if k' is large enough, then the search process will converge to a better solution or y^b. If a better solution y' is found, CAM_DCM finds a discrete local maximizer x' of the problem (MC) by the local search algorithm MCFM from the projected solution of y', and an updating strategy, which will be given in the next subsection, is used to update the elite solution set P (Algorithm 4, lines 8, and 9).

Otherwise, CAM_DCM finds a discrete local maximizer x'' of the problem (MC) by the local search algorithm MCFM from the projected solution of y^l (Algorithm 4, line 14), and applies CAM to a construct clustered graph G' from $\{x'', x^*\}$ (Algorithm 4, line 15). An auxiliary function $T'(y, k')$ with k', y^b is constructed, where y^b is the clustered solution of the current best solution x^*. By Proposition 4.1, we have $f(x^*) = g(y^b)$. Then, CAM_DCM maximizes $T'(y, k')$ by DCM (Algorithm

4, line 16). Suppose y' is the obtained solution. By Corollary 1, y' must be either equal to y^b or better than y^b. If $g(y') > g(y^b)$, since $g(y^b) = f(x^*)$, we have

$$g(y') > f(x^*). \tag{4.3}$$

CAM_DCM finds a discrete local maximizer x' of problem (MC) by MCFM from the projected solution of y' (Algorithm 4, line 18). By (4.3) and Proposition 4.1, we have $f(x') > f(x^*)$. Then the current best solution x^* is updated, and an updating strategy is applied to updated the elite solution set P (Algorithm 4, line 19).

Algorithm 4 CAM_DCM

Input: graph $G = (V, E)$, two positive numbers p, t such that $p > t$, the tolerance parameter N_3 for terminating the algorithm.

Output: approximate global maximal solution x^*.

1: **for** $i = 1, \cdots, p$ **do**
2: Use local search algorithm MCFM to search for a discrete local maximizer of problem (MC) starting from a randomly generated solution x^i, also denote it by x^i;
3: **end for**
4: Let $N = 0$, and $P = \{x^1, x^2, \cdots, x^p\}$. Set $x^* = \text{argmax}\{f(x^i), i = 1, \cdots, p\}$.
5: **repeat**
6: Select t solutions from P, and put them into elite solution set M. Apply the clustering algorithm CAM to construct a clustered graph $G' = (V', E')$ from elite solution set M. Let $x^b = \text{argmax}_{x^i \in M}\{f(x^i)\}$, and denote its clustered solution as y^b.
7: Set $y' = \text{DCM}(G', y^b, \delta_{k'}, N_1)$, and y^l be a discrete local maximizer of problem (CMC) obtained by MCFM in DCM (i.e., a discrete local maximizer obtained from maximizing $T'(y, k')$ with $k' = 0$ in Algorithm 3 line 6).
8: **if** $g(y') > g(y^b)$ **then**
9: Find a discrete local maximizer x' of problem (MC) by MCFM from the projected solution of y'. Use updating method (Section 4.3) to update P.
10: **if** $f(x') > f(x^*)$ **then**
11: Set $x^* = x'$.
12: **end if**
13: **else**
14: Find a discrete local maximizer x'' of problem (MC) by MCFM from the projected solution of y^l.
15: Apply CAM to construct a clustered graph $G' = (V', E')$ from solution set $\{x'', x^*\}$. Set y^b be the clustered solution of x^*.
16: Set $y' = \text{DCM}(G', y^b, \delta_{k'}, N_2)$.
17: **if** $g(y') > g(y^b)$ **then**
18: Find a discrete local maximizer x' of problem (MC) by MCFM from the projected solution of y'.
19: Set $x^* = x'$. Use updating method (Section 4.3) to update P.
20: **end if**
21: **end if**
22: Set $N = N + 1$.
23: **until** $N > N_3$
24: **return** x^* and $f(x^*)$ as an approximate global maximal solution and global maximal value of the max cut problem.

Remark 1 In Algorithm 4, line 7, we set $N_1 = 1$ for algorithm DCM. Then y^l is the only discrete local maximizer of (CMC) found by MCFM, and its projected solution may be used as an initial solution of MCFM (Algorithm 4, line 14).

Remark 2 In Algorithm 4, line 15, CAM is used to construct clustered graph G' from $\{x'', x^*\}$. In this case, we set the parameters of CAM equal to n, that is to say, we do not split the clusters into small clusters.

Remark 3 In Algorithm 4, line 16, we set $N_2 = 5$ for algorithm DCM. As the algorithm progresses, the number of vertices, which are partitioned in the same subset in x'' and the current best solution x^*, becomes larger, and the clustered graph G' becomes smaller.

4.3 Elite Solution Set Updating Method

By the Observation 3.1, both solution quality and diversification of the selected solution set M interplay the quality of clustered graph G' generated by CAM. It is necessary to preserve the diversity of the elite solution set P. A number of strategies have been presented to control the diversity of the population in memetic algorithms [27, 35, 38].

[27, 35, 38] use a function to determine whether an offspring is added to the population or not. The function takes two factors into account: the quality of the solution and the diversity of the population after addition of the solution. This approach has been successfully applied to solve many combinatorial optimization problems, such as the graph coloring problem, the max-bisection problem, the multidimensional knapsack problem, and the total weighted tardiness single-machine scheduling problem.

In this paper, we adopt an elite solution set updating strategy used in [27, 38]. It takes both the quality and the diversity of the set P into account. Its basic idea is using a function to decide whether a solution should be added to P or not, and which solution in P should be deleted. A distance measure is used to evaluate how much a solution diversifies the population. We first give the definitions of the distance between two solutions and the distance between a solution and a solution set.

Definition 4.3 Given two solutions $x = (x_1, \cdots, x_n), y = (y_1, \cdots, y_n)$, the distance $d(x, y)$ between x and y can be defined as the least number of vertices necessary to transform x to y.

Since $x, y \in \{1, -1\}^n$, We can calculate the distance $d(x, y)$ as follows:

$$d(x, y) = \frac{1}{2}\sum_{i=1}^{n} |x_i - y_i|. \tag{4.4}$$

If a solution that has a small distance to x is already in the elite solution set P, then inserts x into P , which will not diversify the elite solution set P.

Definition 4.4 Given a solution set $Q = \{x^1, \cdots, x^q\}$, the distance of a given solution $x \notin Q$ to the solution set Q is defined as follows:

$$d(x, Q) = \min_{x^i \in Q} \mathrm{d}(\mathrm{x}, \mathrm{x}^{\mathrm{i}}). \tag{4.5}$$

The quality of a solution x can be measured by the objective function value $f(x)$. We adopt a quality-and-distance scoring function, which was originally proposed in [27], to decide whether a new solution x should be added to an elite solution set P or not.

Definition 4.5 Given a solution set $Q = \{x^1, \cdots, x^q\}$ and a solution $x \notin Q$, the quality-and-distance scoring function [27, 38] is defined as:

$$h(x, Q) = \beta \widetilde{A}(f(x)) + (1 - \beta)\widetilde{A}(d(x, Q)). \tag{4.6}$$

where $f(x)$ is the objective function value, β is a parameter set to 0.6 according to [27, 38], and $\widetilde{A}(.)$ represents the normalized function:

$$\widetilde{A}(x) = \frac{x - x_{\min}}{x_{\max} - x_{\min} + 1}, \tag{4.7}$$

where $x_{\min}$ and $x_{\max}$ are respectively the minimum and maximum of x in the solution set Q, and "+1" is used to avoid the possibility of a 0 denominator.

Algorithm 5 shows the solution set updating strategy. Suppose x' is a new solution which obtained by MCFM. Algorithm 5 uses the following strategy to update the elite solution set $P = \{x^1, \cdots, x^p\}$. The values of $h(x, P)$ and $h(x^i, P - \{x^i\} \cup \{x\})$, for each $x^i \in P$ are calculated according to (4.6), and suppose x^f and x^s are the solutions with the smallest value and the second smallest value in $P \cup \{x\}$, respectively. If $x^f \neq x$, then x is inserted into the population and x^f is deleted from the population. It either improves the quality of the population or diversifies the population. Otherwise, the solution x^s is replaced by x with a probability $p_r = 0.2$. The population updating procedure is presented as follows.

Algorithm 5 solution set updating method

Input: an elite solution set $P = \{x^1, \cdots, x^p\}$, and a solution x', update probability p_r.
Output: Updated elite solution set $P = \{x^1, \cdots, x^p\}$.

calculate $d(x, P)$ according to (7).
calculate $h(x, P)$ according to (8).
for $i = 1, \cdots, p$ **do**
  calculate $d(x^i, P - \{x^i\} \cup \{x\})$ according to (7).
  calculate $h(x^i, P - \{x^i\} \cup \{x\})$ according to (8).
end for
Let $x^f = \text{argmin}\{h(x, P), h(x^i, P - \{x^i\} \cup \{x\}), i = 1, \cdots, p\}$.
if $x^f \neq x$ **then**
  x is inserted into P, and x^f is deleted from P, i.e., $P = P - \{x^f\} \cup \{x\}$.
else
  if rand$(0, 1) < p_r$ **then**
    Let $x^s = \text{argmin}\{h(x^i, P - \{x^i\} \cup \{x\}) | i = 1, \cdots, p\}$.
    x is inserted into P, and x^s is deleted from P, i.e., $P = P - \{x^s\} \cup \{x\}$.
  end if
end if
return P.

5 Experimental Results

In this section, we report some computational results and comparisons to show the efficiency of the proposed algorithm. The proposed algorithm CAM_DCM was implemented using C++ language and was run on a 2.11 GHz clockpulse and 1.0 GB RAM under Windows XP.

5.1 Test Instances

Two well-known instance sets from the literature are used to evaluate the performance of the proposed algorithm.

The first set of standard benchmarks are G-set graphs. These graphs were created by Helmberg and Rendl [15], and have been used to test a lot of algorithms for the max-cut problem, such as [5, 8, 20, 24, 25, 41]. These graphs vary in size from 800 to 20 000 vertices, and can be downloaded from http://www.stanford.edu/yyye/yyye/Gset/.

The second set of benchmarks, which were proposed by Burer, Monteiro, and Zhang [5], arising from Ising spin glasses cubic lattice graphs. It consists of two groups of benchmarks. The first group contains 10 graphs with 1 000 vertices and density 0.60 %. The second group contains 10 graphs with 2 744 vertices and density 0.22 %.

5.2 Parameters Setting

Table 1 gives the parameters used in the proposed algorithm CAM_DCM. The local search algorithm MCFM was original proposed in [22]. Like [22], if MCFM is used to search the original problem, we set $p_1 = \frac{n}{3}$, otherwise, we set $p_1 = n$. CAM_DCM maintains an elite solution set P with $p = 20$, each time $t = 12$; elite solutions are selected randomly from P to construct clusters, in which the vertices have identical components of the t solutions. If there exists a cluster, in which the number of vertices is bigger than s, then we randomly decompose the cluster into small

Table 1 Parameters setting

Parameters	Section	Value
p_1	3.1	$\frac{n}{3}$ or n
p	3.1	20
t	3.1	12
s	3.1	2 or 3
N_1	4.1	1
N_2	4.1	5
N_3	4.1	300 or 2 000
$\delta_{k'}$	4.1	$\frac{w'}{10}$
β	4.3	0.4
p_r	4.3	0.2

clusters. For the graph with $n < 3\,000$, we set $s = 2$; otherwise, $s = 3$. In Algorithm 4, lines 7 and 16, we set $N_1 = 1$ and $N_2 = 5$, respectively. Let

$$w' = \max\{w'_{i'j'} : \{i',j'\} \in E'\},$$

where E' is the edge set of the clustered graph G', and $w'_{i'j'}$ is the weight of the edge $\{i',j'\}$. Like [22], we set $\delta_{k'} = \frac{w'}{10}$.

5.3 Comparison with Heuristics Based on Auxiliary Function

In this experiment, we compare the proposed algorithm CAM_DCM with the discrete dynamic convexized method (DCMMC) [22] and with the new discrete filled function method DF^{2}A due to Ling et al. [23].

We run our proposed algorithm with the parameters in Table 1 ($N_3 = 300$, $s = 2$) and DCMMC [22] with the parameters $N_L = 1\,000$ on benchmark G40 on our computer, respectively. The cut value obtained by each algorithm and the corresponding CPU time (in seconds) are reported in Fig. 2.

As shown in Fig. 2, compared with DCMMC, CAM_DCM used less CUP time, and found the better solution on G40.

We run the proposed algorithm CAM_DCM with parameters in Table 1 ($N_3 = 300$) on the two sets of benchmark instances. We also implemented DCMMC [22] with parameters $N_L = 1\,000$ and $\delta_k = \frac{w}{10}$ on our computer. The experimental results on two instance sets are reported in Tables 2 and 3, respectively, where $w = \max\{w_{ij} : \{i,j\} \in E\}$. The experimental results of DF^{2}A on the first set instances are also listed in Table 2. In Tables 2 and 3, the subcolumns "Cut" and "Time" list the best cut value and CPU time (in seconds) obtained by CAM_DCM,

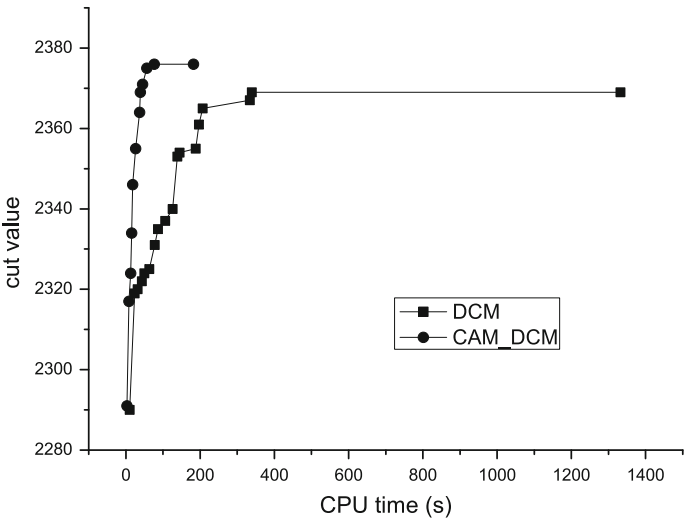

Fig. 2 Cut value and CPU time obtained by CAM_DCM and DCM on G40.

Table 2 Experimental results and comparisons on the fist instance set

Instance			DF^2A [23]		DCMMC [22]			CAM_DCM		
Name	n	m	Cut	Time	Cut	Time	To best	Cut	Time	To best
G1	800	19 176	–	–	11 624	266.016	47.112	11 624	37.375	18.281
G2	800	19 176	–	–	11 616	287.953	35.515	11 617	41.360	20.766
G3	800	19 176	–	–	11 622	275.094	77.576	11 622	38.735	14.641
G4	800	19 176	–	–	11 646	290.157	84.875	11 646	38.876	10.563
G5	800	19 176	–	–	11 626	297.017	111.656	11 631	35.829	4.282
G6	800	19 176	–	–	2 178	288.578	115.875	2 178	46.422	36.563
G7	800	19 176	–	–	1 992	324.128	15.375	2 001	48.719	28.172
G8	800	19 176	–	–	2 004	312.234	297.547	2 005	39.047	26.638
G9	800	19 176	–	–	2 043	310.047	69.617	2 046	47.859	33.234
G10	800	19 176	–	–	1 996	311.313	244.453	1 998	39.797	5.156
G11	800	1 600	556	74.71	564	127.015	6.594	564	58.906	11.934
G12	800	1 600	558	96.24	556	121.140	62.828	556	53.297	36.938
G13	800	1 600	576	102.17	582	121.687	78.812	582	54.579	23.297
G14	800	4 694	3 055	103.61	3 052	135.344	110.392	3 058	48.297	31.707
G15	800	4 661	3 041	114.13	3 048	134.234	105.684	3 049	39.254	36.938
G16	800	4 672	–	–	3 042	134.375	44.156	3 039	79.829	39.572
G17	800	4 667	–	–	3 033	152.937	33.437	3 039	45.641	26.422
G18	800	4 694	–	–	984	179.765	48.219	989	28.704	10.266
G19	800	4 661	–	–	901	161.594	51.469	906	32.110	17.563
G20	800	4 672	–	–	935	168.890	34.703	936	27.083	14.750
G21	800	4 667	–	–	921	163.154	79.781	927	28.391	19.388
G22	2 000	19 990	13 329	151.27	13 358	1 042.758	728.681	13 332	128.282	90.763
G23	2 000	19 990	13 272	162.09	13 325	962.759	439.427	13 332	137.916	57.937
G24	2 000	19 990	13 291	157.15	13 321	955.407	432.860	13 321	163.141	135.937

Table 2 continued

Instance			DF^{2}A [23]		DCMMC [22]			CAM_DCM		
Name	n	m	Cut	Time	Cut	Time	To best	Cut	Time	To best
G25	2 000	19 990	–	–	13 298	1 339.437	246.765	13 312	144.219	142.203
G26	2 000	19 990	–	–	13 283	1 609.251	1 154.078	13 298	188.594	145.453
G27	2 000	19 990	–	–	3 257	1 634.895	461.828	3 289	126.064	121.377
G28	2 000	19 990	–	–	3 268	1 531.531	323.257	3 269	124.437	108.562
G29	2 000	19 990	–	–	3 294	1 009.344	363.359	3 338	215.281	143.984
G30	2 000	19 990	–	–	3 396	1 313.375	267.827	3 392	123.735	110.964
G31	2 000	19 990	–	–	3 287	1 349.231	1 186.653	3 283	136.798	88.406
G32	2 000	4 000	–	–	1 408	1 199.609	817.172	1 406	337.297	66.251
G33	2 000	4 000	–	–	1 380	830.157	404.922	1 380	318.274	72.149
G34	2 000	4 000	–	–	1 378	842.734	116.359	1 378	330.843	97.710
G35	2 000	11 778	–	–	7 641	747.687	276.094	7 645	354.485	254.947
G36	2 000	11 766	–	–	7 635	765.109	692.234	7 654	294.703	277.266
G37	2 000	11 785	–	–	7 644	753.671	399.312	7 661	252.926	231.781
G38	2 000	11 779	–	–	7 624	649.625	289.703	7 652	267.578	254.031
G39	2 000	11 778	–	–	2 375	1 354.640	1 135.578	2 375	109.813	40.625
G40	2 000	11 766	–	–	2 369	1 332.547	339.407	2 376	182.360	76.578
G41	2 000	11 785	–	–	2 354	1 349.219	1 133.713	2 385	85.937	70.078
G42	2 000	11 779	–	–	2 453	1 344.631	1 095.703	2 455	84.265	62.531
G43	1 000	9 990	–	–	6 657	281.782	166.250	6660	37.422	33.219
G44	1 000	9 990	–	–	6 649	282.796	59.547	6650	33.483	8.641
G45	1 000	9 990	–	–	6 649	272.254	64.094	6650	41.718	35.513
G46	1 000	9 990	–	–	6 636	370.250	40.922	6641	37.031	35.594
G47	1 000	9 990	–	–	6 651	509.157	137.794	6646	45.706	26.857
G48	3 000	6 000	–	–	6 000	331.895	0.327	6000	77.172	0.321

Table 2 continued

Instance			DF^{2}A [23]		DCMMC [22]			CAM_DCM		
Name	n	m	Cut	Time	Cut	Time	To best	Cut	Time	To best
G49	3 000	6 000	–	–	6 000	405.250	0.426	6000	90.047	0.399
G50	3 000	6 000	–	–	5 880	491.625	0.729	5880	194.078	0.609
G51	1 000	5 909	–	–	3 823	352.594	301.250	3842	55.922	46.985
G52	1 000	5 916	–	–	3 827	342.641	34.718	3832	73.839	61.172
G53	1 000	5 914	–	–	3 831	351.219	97.787	3846	64.937	42.765
G54	1 000	5 916	–	–	3 831	325.094	271.752	3830	101.594	75.236
G55	5 000	12 498	–	–	10 191	4 794.016	1 085.860	10 208	394.837	313.703
G56	5 000	12 498	–	–	3 925	4 598.126	1 081.347	3 935	360.672	357.765
G57	5 000	10 000	–	–	3 460	7 118.651	4 574.750	3 462	1 699.906	458.859
G58	5 000	29 570	–	–	19 112	6 211.235	4 704.391	19 175	1 565.032	1 558.594
G59	5 000	29 570	–	–	5 882	5 917.951	4 716.238	5 967	528.562	442.578
G60	7 000	17 148	–	–	14 061	7 432.047	7 069.714	14 061	762.891	553.391
G61	7 000	17 148	–	–	5 650	7 797.327	7 605.530	5 718	696.874	499.765
G62	7 000	14 000	–	–	4 818	6 969.531	6 537.062	4 798	3 013.795	2 588.813
G63	7 000	41 459	–	–	26 779	7 051.937	6 381.905	26 880	2 796.859	2 775.468
G64	7 000	41 459	–	–	8 538	7 539.183	6 817.247	8 561	924.156	839.859
G65	8 000	16 000	–	–	5 476	7 903.089	6 978.362	5 494	3 807.129	2 944.371
G66	9 000	18 000	–	–	6 258	8 272.867	7 728.425	6 270	3 788.453	2 637.766
G67	10 000	20 000	–	–	6 820	8 749.832	7 303.683	6 858	4 590.359	4 090.754
G70	10 000	20 000	–	–	9 355	8 634.692	8 012.839	9 365	921.913	736.032
G72	10 000	20 000	–	–	6 868	8 692.736	7 938.257	6 894	4 218.357	4 027.872
Average time						2 035.943	1 503.942		520.865	410.283

Table 3 Experimental results and comparisons on the second instance set

Instance			DCMMC [22]			CAM_DCM		
Name	n	m	Cut	Time	To best	Cut	Time	To best
sg3dl101000	1 000	3 000	894	218.078	56.016	894	35.953	11.257
sg3dl102000	1 000	3 000	900	224.547	61.688	900	32.197	6.969
sg3dl103000	1 000	3 000	892	226.109	81.617	892	32.620	17.906
sg3dl104000	1 000	3 000	896	219.390	14.015	898	33.361	7.577
sg3dl105000	1 000	3 000	882	231.781	163.031	884	61.813	49.235
sg3dl106000	1 000	3 000	886	232.062	42.265	888	36.907	7.672
sg3dl107000	1 000	3 000	898	237.485	105.209	898	38.531	30.719
sg3dl108000	1 000	3 000	880	249.079	30.313	880	37.623	27.313
sg3dl109000	1 000	3 000	900	238.516	46.891	900	41.704	17.172
sg3dl1010000	1 000	3 000	892	236.371	106.687	892	34.781	22.906
sg3dl141000	2 744	8 232	2 426	1 869.624	589.343	2 432	361.563	278.453
sg3dl142000	2 744	8 232	2 424	1 872.515	1 580.089	2 448	365.406	304.435
sg3dl143000	2 744	8 232	2 426	1 940.157	1 003.578	2 430	345.109	284.562
sg3dl144000	2 744	8 232	2 426	1 845.516	822.672	2 432	420.266	272.174
sg3dl145000	2 744	8 232	2 420	1 837.141	207.157	2 428	442.110	265.297
sg3dl146000	2 744	8 232	2 426	1 870.938	1 508.922	2 434	512.109	346.842
sg3dl147000	2 744	8 232	2 416	1 885.176	489.381	2 428	390.781	369.867
sg3dl148000	2 744	8 232	2 422	1 873.584	872.382	2 438	442.328	284.031
sg3dl149000	2 744	8 232	2 394	1 843.579	1 479.308	2 412	455.121	301.953
sg3dl1410000	2 744	8 232	2 430	1 829.057	723.412	2 440	440.922	410.456
Average time				1 049.035	499.198		228.060	165.839

DCMMC and DF^{2}A, respectively. And the subcolumn "To best" lists for each graph the time taken to meet the best solution in the whole run. The last rows of Tables 2 and 3 list the average time of the first and second instance sets, respectively. the "–" in Table 2 denotes that a result for that particular instance was not available. The data under column "DF^{2}A" are completely cited from [23].

We can make the following observations about the results in Tables 2 and 3:

(1) [23] used 8 instances in the first instance set to test their algorithm. In terms of solution quality, our proposed algorithm found better solutions on these instances, except G12.

(2) The proposed algorithm obtained the cut value as large as the cut value obtained by DCMMC in 14 out of 69 instances in the first instance set, and found better solutions in 46 instances of the first instance set. For the first instance set, the average CPU time of DCMMC and CAM_DCM is 2 035.943s and 520.865s, respectively. The average time taken to meet the best solution in the whole run of DCMMC and CAM_DCM are 1 503.942s and 410.283s. It shows that runs for our proposed algorithm were about 3 times faster than DCMMC in 1 000 iterations.

(3) For the second instance set, the proposed algorithm obtained cut value at least as large as the cut value obtained by DCMMC on each test instance. Moreover, the proposed algorithm found better solutions in 23 of 40 instances. The average CPU time of DCMMC and CAM_DCM is 1 049.035s and 228.060s, respectively. The average time taken to meet the best solution in the whole run of DCMMC and CAM_DCM is 499.198s and 165.839s. It shows that runs for our proposed algorithm were about 3 times faster than DCMMC in 1 000 iterations.

The above observations show that CAM_DCM benefits a lot from clustered adaptive multistart method both in terms of solution quality and solution time, and CAM_DCM can obtain high-quality solutions of the max-cut problem in an acceptable time.

5.4 Comparison with Other Heuristics

In this second experimental study, we focus on comparing the performance of the proposed algorithm CAM_DCM with respect to scatter search (SS) [28], rank-two relaxation heuristic [5], tabu search (TS) [18], breakout local search (BLS) [3]. These heuristics are able to obtain high-quality solutions of the max-cut problem. In particular, the breakout local search (BLS) [3] has shown to be the best existing technique to solve the max-cut problem up to now.

We ran the proposed algorithm CAM_DCM 20 times with parameters in Table 1 ($N_3 = 2\,000$) on the first instance set. The column "CAM_DCM" of Table 4 presents the best cut values of the test instances among 20 runs of CAM_DCM. For comparison purposes, the best cut values of the test instances obtained by scatter search [28], rank-two relaxation heuristic [5], tabu search [18], breakout local search [3] are also cited in the columns "SS", "CirCut", "TS", and "BLS" of Table 4, respectively. A "–" in the Table 4 means that the algorithm does not report result for the test instance.

The data under the columns "SS" and "TS" are from [18]. [18] presented the best cut values found by tabu search [18] in 2.36, 8, 12, 20, and 24 h on their computer for instances with $n \leqslant 3\,000$, $5\,000 \leqslant n \leqslant 7\,000$, $n = 8\,000$, $n = 9\,000$, and $n = 10\,000$, respectively. The results listed under "BLS" are taken from [3]. [3] ran the breakout local search 20 times on each instance of the first instance set, each run was limited to $200\,000n$ iterations. And the best cut values were reported among 20 runs. The data under the column "CirCut" are taken from [37].

We make the following observations on the results shown in Table 4:

(1) Table 4 gives the best solutions found by the scatter search (SS) [28], rank-two relaxation heuristic (CirCut) [5] on instances G1–G54. Compared with the scatter search (SS), the proposed algorithm found better solutions on 44 of the 54 instances, and found the same solutions on 8 instances. The proposed algorithm found the cut values at least as large as the cut values obtained by the rank-two relaxation heuristic (CirCut) on the 54 instances, moreover, the proposed algorithm found better solutions on 45 of the 54 instances.

Table 4 Comparison with the state-of-the-art algorithms in terms of the best solutions obtained

Name	n	m	SS	CirCut	TS	BLS	CAM_DCM
G1	800	19 176	11 624	11 624	11 624	1 1624	1 1624
G2	800	19 176	11 620	11 617	11 620	1 1620	1 1620
G3	800	19 176	11 622	11 622	11 622	1 1622	1 1622
G4	800	19 176	11 646	11 641	11 646	1 1646	1 1646
G5	800	19 176	11 631	11 627	11 631	1 1631	1 1631
G6	800	19 176	2 165	2 178	2 178	2 178	2 178
G7	800	19 176	1 982	2 003	2 006	2 006	2 006
G8	800	19 176	1 986	2 003	2 005	2 005	2 005
G9	800	19 176	2 040	2 048	2 054	2 054	2 054
G10	800	19 176	1 993	1 994	2 000	2 000	2 000
G11	800	1 600	562	560	564	564	564
G12	800	1 600	552	552	556	556	556
G13	800	1 600	578	578	580	582	582
G14	800	4 694	3 060	3 060	3 061	3064	3 063
G15	800	4 661	3 049	3 049	3 050	3050	3 050
G16	800	4 672	3 045	3 045	3 052	3052	3 052
G17	800	4 667	3 043	3 043	3 046	3047	3 047
G18	800	4 694	9 88	978	991	992	992
G19	800	4 661	903	888	904	906	906
G20	800	4 672	941	941	941	941	941
G21	800	4 667	930	931	931	931	931
G22	2 000	19 990	13 346	13 346	13 359	13 359	13 359
G23	2 000	19 990	13 317	13 317	13 342	13 344	13 339
G24	2 000	19 990	13 303	13 314	13 337	13 337	13 337
G25	2 000	19 990	13 320	13 326	13 332	13 340	13 333
G26	2 000	19 990	1 3294	13 314	13 328	13 328	13 324
G27	2 000	19 990	3 318	3 306	3 36	3341	3 336
G28	2 000	19 990	3 285	3 260	3 295	3298	3 295
G29	2 000	19 990	3 389	3 376	3 391	3405	3 404
G30	2 000	19 990	3 403	3 385	3 403	3412	3 412
G31	2 000	19 990	3 288	3 285	3 288	3309	3 309
G32	2 000	4 000	1 398	1 390	1 406	1410	1 410
G33	2 000	4 000	1 362	1 360	1 378	1382	1 382
G34	2 000	4 000	1 364	1 368	1 378	1384	1 382
G35	2 000	11 778	7 668	7 670	7 678	7684	7 675
G36	2 000	11 766	7 660	7 660	7 670	7 678	7 671
G37	2 000	11 785	7 664	7 666	7 682	7 689	7 681
G38	2 000	11 779	7 681	7 646	7 683	7 687	7 677
G39	2 000	11 778	2 393	2 395	2 397	2 408	2 398
G40	2 000	11 766	2 374	2 387	2 390	2 400	2 394

Table 4 continued

Name	n	m	SS	CirCut	TS	BLS	CAM_DCM
G41	2 000	11 785	2 386	2 398	2 400	2 405	2 405
G42	2 000	11 779	2 457	2 469	2 469	2 481	2 473
G43	1 000	9 990	6 656	6 656	6 660	6 660	6 660
G44	1 000	9 990	6 648	6 643	6 639	6 650	6 650
G45	1 000	9 990	6 642	6 652	6 652	6 654	6 654
G46	1 000	9 990	6 634	6 645	6 649	6 649	6 649
G47	1 000	9 990	6 649	6 656	6 656	6 657	6 657
G48	3 000	6 000	6 000	6 000	6 000	6 000	6 000
G49	3 000	6 000	6 000	6 000	6 000	6 000	6 000
G50	3 000	6 000	5 880	5 880	5 880	5 880	5 880
G51	1 000	5 909	3 846	3 837	3 847	3 848	3 847
G52	1 000	5 916	3 849	38 33	3 849	3 851	3 850
G53	1 000	5 914	3 846	3 842	3 848	3 850	3 847
G54	1 000	5 916	3 846	3 842	3 851	3 852	3 848
G55	5 000	12 498	–	–	1 0236	10 294	10 265
G56	5 000	12 498	–	–	3 934	4 012	3 984
G57	5 000	10 000	–	–	3 460	3 492	3 472
G58	5 000	29 570	–	–	19 248	19 263	1 9231
G59	5 000	29 570	–	–	6 019	6 078	6 025
G60	7 000	17 148	–	–	14 057	14 176	14 129
G61	7 000	17 148	–	–	5 680	5 798	5 720
G62	7 000	14 000	–	–	4 822	4 898	4 830
G63	7 000	41 459	–	–	2 6963	26 997	26 933
G64	7 000	41 459	–	–	8 610	8 735	8 628
G65	8 000	16 000	–	–	5 518	5 558	5 522
G66	9 000	18 000	–	–	6 304	6 360	6 302
G67	10 000	20 000	–	–	6 894	6 940	6 874
G70	10 000	20 000	–	–	9 458	9 541	9 444
G72	10 000	20 000	–	–	6 922	6 998	6 918

(2) Compared with the tabu search (TS) [18], the proposed algorithm found better solutions on 30 of the 69 instances, and found the same solutions on 26 instances.

(3) In terms of solution quality, the breakout local search (BLS) [3] found the best solutions on all tested instances in the first instance set. Compared with the breakout local search, the proposed algorithm found the same solutions on 35 out of 69 instances, and the average percent deviation from the best solution obtained by breakout local search (BLS) in the first instance set is about 0.20 %.

Above observations show that the proposed algorithm is very competitive in terms of solution quality.

6 Conclusions

In this paper, we have presented a hybrid algorithm CAM_DCM by combining the clustered adaptive multistart [13] and the discrete dynamic convexized method for solving the max-cut problem. The proposed algorithm starts from building an initial elite set of solutions by the local search procedure MCFM. In each subsequent iteration, the proposed algorithm constructs a clustered graph from the previous elite solutions. The local search procedure MCFM is applied to the clustered graph from random solutions, and the discrete dynamic convexized method is used to escape from the previous discrete local maximizers. Then, by collapsing the groups of vertices in the clustered graph, a new solution is obtained, and inserted into the elite solution set by an updating method. This iterative process ends when the termination criteria are satisfied. Experiments were done on two sets of well-known benchmark instances. Comparisons with the dynamic convexized method and the filled function method showed that the proposed algorithm can find high-quality solutions in an acceptable time. Compared with some well-known algorithms for the max-cut problem, it reveals that our proposed algorithm is very competitive.

Acknowledgments We are grateful to the anonymous referees for valuable suggestions and comments which have helped us to improve the paper.

References

[1] Arráiz, E., Olivo, O.: Competitive simulated annealing and tabu search algorithms for the max-cut problem, In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2009), pp. 1797–1798 (2009)
[2] Barahona, F., Grötschel, M., Reinelt, G.: An application of combinatorial optimization to statistical physical and circuit layout design. Oper. Res. **36**(3), 493–513 (1988)
[3] Benlic, U., Hao, J.K.: Breakout local search for the max-cut problem. Eng. Appl. Artif. Intell. **26**(3), 1162–1173 (2013)
[4] Burer, S., Monteiro, R.D.C.: A projected gradient algorithm for solving the maxcut SDP relaxation. Optim. Methods Softw. **15**(3–4), 175–200 (2001)
[5] Burer, S., Monteiro, R.D.C., Zhang, Y.: Rank-two relaxation heuristics for max-cut and other binary quadratic programs. SIAM J. Optim. **12**(2), 503–521 (2002)
[6] Chang, K.C., Du, D.-Z.: Efficient algorithms for layer assignment problems. IEEE Trans. Comput.-Aided Design **6**(1), 67–78 (1987)
[7] Elf, M., Jünger, M., Rinaldi, G.: Minimizing breaks by maximizing cuts. Oper. Res. Lett. **31**(5), 343–349 (2003)
[8] Festa, P., Pardalos, P.M., Resende, M.G.C., Ribeiro, C.C.: Randomized heuristics for the max-cut problem. Optim. Methods Softw. **17**(6), 1033–1058 (2002)
[9] Fiduccia, C.M., Mattheyses, R.M.: A linear time heuristic for improving network partitions, in Proc. ACM/IEEE DAC, 175–181 (1982)
[10] Garey, M., Johnson, D., Stochmeter, L.: Some simplified NP-complete graph problems. Theor. Comput. Sci. **1**(3), 237–267 (1976)
[11] Ghaddar, B., Anjos, M.F., Liers, F.: A branch-and-cut algorithm based on semidefinite programming for the minimum k-partition problem. Ann. Oper. Res. **188**(1), 155–174 (2011)
[12] Goemans, M.X., Williams, D.P.: Improved approximation algorithms for max-cut and satisfiability problems using semidefinite programming. J. ACM **42**(6), 1115–1145 (1995)
[13] Hagen, L.W., Kahng, A.B.: Combining problem reduction and adaptive multistart: a new technique for superior iterative partitioning. IEEE Transac. Comput.-Aided Design Integerated Circuits Sys. **16**(7), 709–717 (1997)

[14] Han, Q., Ye, Y., Zhang, J.: An improved rounding method and semidefinite programming relaxation for graph partition. Math. Program. **92**(3), 509–535 (2002)
[15] Helmberg, C., Rendl, F.: A spectral bundle method for semidefinite programming. SIAM J. Optim. **10**(3), 673–696 (2000)
[16] Hendrickson, B., Leland, R.: An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM J. Sci. Comput. **16**(2), 452–469 (1995)
[17] Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R., Thatcher, J. (eds.) Complex. Comput. Comput., pp. 85–103. Plenum Press, New York (1972)
[18] Kochenberger, G.A., Hao, J.K., Lü, Z.P., Wang, H.B., Glover, F.: Solving large scale max cut problems via tabu search. J. Heuristics **19**(4), 565–571 (2013)
[19] Krishnan, K., Mitchell, J.: A semidefinite programming based polyhedral cut and price approach for the max-cut problem. Comput. Optim. Appl. **33**(1), 51–71 (2006)
[20] Laguna, M., Duarte, A., Martí, R.: Hybridizing the cross-entropy method: an application to the max-cut problem. Comput. Oper. Res. **36**(2), 487–498 (2009)
[21] Lin, R.B., Chen, S.Y.: Conjugate conflict continuation graphs for multi-layer constrained via minimization. Inf. Sci. **177**(12), 2436–2447 (2007)
[22] Lin, G., Zhu, W.X.: A discrete dynamic convexized method for the max-cut problem. Ann. Oper. Res. **196**(1), 371–390 (2012)
[23] Ling, A.F., Xu, C.X.: A new discrete filled function method for solving large scale max-cut problems. Numer. Algorithms **60**(3), 435–461 (2012)
[24] Ling, A.F., Xu, C.X., Xu, F.M.: A discrete filled function algorithm for approximate global solutions of max-cut problems. J. Comput. Appl. Math. **220**(1–2), 643–660 (2008)
[25] Ling, A.F., Xu, C.X., Xu, F.M.: A discrete filled function algorithm embedded with continuous approximation for solving max-cut problem. Eur. J. Oper. Res. **197**(2), 519–531 (2009)
[26] Liu, H.W., Wang, S.H., Liu, S.Y.: Feasible direction algorithm for solving SDP relaxation of the quadratic-1, 1 programming. Optim. Methods Softw. **19**(2), 125–136 (2004)
[27] Lü, Z., Glover, F., Hao, J.K.: A hybrid metaheuristic approach to solving the UBQP problem. Eur. J. Oper. Res. **207**(3), 1254–1262 (2010)
[28] Martí, R., Duarte, A., Laguna, M.: Advanced scatter search for the max-cut problem. INFORMS J. Comput. **21**(1), 26–38 (2009)
[29] Papadimitriou, C.H., Yannakakis, M.: Optimization, approximation, and complexity classes. J. Comput. System Sci. **43**(3), 425–440 (1991)
[30] Pinter, R.Y.: Optimal layer assignment for interconnect. J. VLSI Comput. Systems **1**(2), 123–137 (1984)
[31] Rendl, F., Rinaldi, G., Wiegele, A.: Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. Math. Program. **121**(2), 307–335 (2010)
[32] Sahni, S., Gonzales, T.: P-complete approximation problem. J. ACM **23**(3), 555–565 (1976)
[33] Shylo, V.P., Shylo, O.V.: Solving the maxcut problem by the global equilibrium search. Cybern. Syst. Anal. **46**(5), 744–754 (2010)
[34] De Simone, C., Diehl, M., Jünger, M., Mutzel, P., Reinelt, G., Rinaldi, G.: Exact ground states of Ising spin glasses: new experimental results with a branch-and-cut algorithm. J. Stat. Phys. **80**(1–2), 487–496 (1995)
[35] Sörensen, K., Sevaux, M.: MA|PM: memetic algorithms with population management. Comput. Oper. Res. **33**(5), 1214–1225 (2006)
[36] Trevisan, L., Sorkin, G.B., Sudan, M., Williamson, D.P.: Gadgets, approximation, and linear programming. SIAM J. Comput. **29**(6), 2074–2097 (2000)
[37] Wang, Y., Lü, Z.P., Glover, F., Hao, J.K.: Probabilistic grasp-tabu search algorithms for the UBQP problem. Comput. Oper. Res. **40**(12), 3100–3107 (2013)
[38] Wu, Q.H., Hao, J.K.: Memetic search for the max-bisection problem. Comput. Oper. Res. **40**(1), 166–179 (2013)
[39] Zhu, W.X.: A dynamic convexized function with the same global minimizers for global optimization. Lect. Notes Comput. Sci. **4221**, 939–948 (2006)
[40] Zhu, W.X., Ali, M.M.: Solving nonlinearly constrained global optimization problem via an auxiliary function method. J. Comput. Appl. Math. **230**(2), 491–503 (2009)
[41] Zhu, W.X., Fan, H.: A discrete dynamic convexized method for nonlinear integer programming. J. Comp. Appl. Math. **223**(1), 356–373 (2009)
[42] Zhu, W.X., Lin, G.: A dynamic convexized method for nonconvex mixed integer nonlinear programming. Comput. Oper. Res. **38**(12), 1792–1804 (2011)
[43] Zhu, W.X., Lin, G., Ali, M.M.: Max-*k*-cut by the discrete dynamic convexized method. INFORMS J. Comp. **25**(1), 27–40 (2013)