

Constructing Test Functions for Global Optimization Using Continuous Formulations of Graph Problems

Balabhaskar Balasundaram* and Sergiy Butenko †

Department of Industrial Engineering, Texas A&M University, College Station, Texas 77843, USA

Abstract

A method for constructing test functions for global optimization which utilizes continuous formulations of combinatorial optimization problems is suggested. In particular, global optimization formulations for the maximum independent set, maximum clique and MAX CUT problems on arbitrary graphs are considered, and proofs for some of them are given. A number of sample test functions based on these formulations are proposed.

Keywords: Maximum independent set; Maximum clique; MAX CUT; Constrained global optimization; Test problems

1 Introduction

Good test functions are essential for experimental evaluation of optimization software. Test instances representing problems arising in important practical applications are especially valuable. In this paper, we present an approach which allows to generate difficult global optimization test problems of various sizes. The proposed approach is based on utilizing constrained global formulations of optimization problems on graphs, which appear in a wide variety of practical situations.

We start our presentation with a brief review of the literature related to the subject of our discussion. An insightful study of global optimization software, test problems and applications by Janos D. Pintér can be seen in [1]. Some of the popular convex and non-convex test problems can be found in Moré et al. [2], Dixon and Szegö [3] and Hock and Schittkowski [4]. On the web, several test problems, useful links and detailed discussions on pertinent issues can be found

*E-mail: baski@tamu.edu

†Corresponding author. E-mail: butenko@tamu.edu

at [5]. In [6] a comprehensive collection of test instances for quadratic programming based on the work of Vicente et al. [7] and Calamai et al. [8] and problems based on nonconvex quadratic programming formulations of quadratic assignment, maximum clique problem (Motzkin-Straus formulation [9]) and minimum concave cost transportation problems can be found. A collection of quadratically constrained problems, polynomial problems and other types of problems available in the literature can also be found in [6]. In a recent work, Gaviano et al. [10] present a procedure for generating non-differentiable, continuously differentiable, and twice continuously differentiable classes of test functions for global optimization. Their method involves distorting a convex quadratic function systematically to introduce local minima. Other related references can also be found in [10].

As we will show in this paper, continuous formulations of combinatorial optimization problems can be effectively utilized to generate hard test instances for global optimization software. Using such formulations, one can generate both small-size and large-scale multiextremal functions which are very useful for testing purposes.

Organizationally, this paper consists of five sections. In the next section, we give definitions and notations used in this work. In Section 3 we present some of the existing and new formulations for selected combinatorial optimization problems that are to provide the candidate functions for generating test instances. In Section 4 we discuss some ideas concerning the use of these formulations for construction of test functions and provide numerous examples. Finally, we conclude with some closing remarks in Section 5.

2 Definitions and Notations

Let $G = (V, E)$ be a simple undirected graph with vertex set $V = \{1, \dots, n\}$ and edge set $E \subseteq V \times V$. For $i \in V$, let $N(i)$ and d_i denote the set and number of neighbors of i in G , respectively. The *complement graph* of G is the graph $\bar{G} = (V, \bar{E})$, where \bar{E} is the complement of E . For a subset $W \subseteq V$ let $G(W)$ denote the subgraph induced by W on G .

A set of vertices $I \subseteq V$ is called an independent set if for every $i, j \in I$, $(i, j) \notin E$, that is the graph $G(I)$ induced by I , is edgeless. An independent set is *maximal* if it is not a subset of any larger independent set, and *maximum* if there are no larger independent sets in the graph. The maximum cardinality of an independent set of G is called the *independence number* of the graph G and is denoted by $\alpha(G)$. A *clique* C is a subset of V such that the subgraph $G(C)$ induced by C is complete. The maximum clique problem is to find a clique of maximum cardinality. The *clique number* $\omega(G)$ is the cardinality of a maximum clique in G .

Clearly, I is a maximum independent set of G if and only if I is a maximum clique of \bar{G} . Computation of $\alpha(G)$ and $\omega(G)$ for general graphs is difficult as the maximum independent set and the maximum clique problems are NP-hard [11]. These problems are very important due to a wide range of their practi-

cal applications, including information retrieval, signal transmission analysis, classification theory, economics, scheduling, experimental design, and computer vision. See [12] for a survey on maximum clique and maximum independent set problems.

Next, we define another problem on graphs that we will use in this paper. Given an undirected graph with edge weights, the MAX CUT problem is to find a partition of the set of vertices into two subsets, such that the sum of the weights of the edges having endpoints in different subsets is maximized. This well-known NP-hard problem has applications in VLSI design, statistical physics, and several other fields [13, 14, 15].

3 Global Optimization Formulations

Recently, several continuous global optimization formulations have been proposed which have led to bounds, approximation algorithms and effective heuristics for maximum independent set, maximum clique and MAX CUT problems [16, 17, 18, 19, 20, 21, 22, 23, 9]. However, when some of these formulations are applied to graphs of relatively small or moderate size, they appear to present a formidable challenge to general purpose global optimization software packages and hence can be used to generate good test instances for such packages.

Below in this section we will discuss some global optimization formulations of the maximum clique, maximum independent set and MAX CUT problems. All of the considered formulations are linearly-constrained, in fact most of them are box-constrained.

3.1 Maximum Clique/Independent Set Problems

Let A_G be the adjacency matrix of G and let e be the n -dimensional vector with all components equal to 1. For a subset C of vertices its characteristic vector x^C is defined by $x_i^C = 1/|C|$ if $i \in C$, $x_i^C = 0$, otherwise, for $i = 1, \dots, n$.

Theorem 3.1 ((Motzkin-Straus [9])) *The global optimal value of the following quadratic program (Motzkin-Straus QP):*

$$\max f(x) = x^T A_G x, \tag{1}$$

subject to

$$\begin{aligned} e^T x &= 1, \\ x &\geq 0 \end{aligned}$$

is given by

$$1 - \frac{1}{\omega(G)},$$

where $\omega(G)$ is the clique number of G . Moreover, a subset of vertices $C \subseteq V$ is a maximum clique of G if and only if the characteristic vector x^C of C is a global maximizer of the above problem.

In its original form, the Motzkin-Straus QP has maximizers which are not in the form of characteristic vectors. Although this property is considered a drawback when the above formulation is used for computing maximal cliques, it is rather a useful property for generating global optimization test problems.

Bomze [24] introduced a regularization of Motzkin-Straus formulation by replacing the objective function in the Motzkin-Straus QP with the function

$$g(x) = x^T \left(A_G + \frac{1}{2} I_n \right) x, \quad (2)$$

where I_n is the $n \times n$ identity matrix. In this formulation, x^* is a local maximum if and only if it is the characteristic vector of a maximal clique in the graph. An effective algorithm for computing all maximal cliques in a moderate size graph would allow to compute all local maxima for this formulation thus making it an attractive tool for creating test instances for local, as well as global optimization. Since the number of maximal cliques in a graph can be as large as $3^{n/3}$, where n is the number of vertices [25], instances with numerous local maxima can be constructed for testing purposes.

Theorem 3.2 *The independence number $\alpha(G)$ satisfies the following global optimization formulations:*

$$\alpha(G) = \max_{0 \neq x \in [0,1]^n} \frac{\left(\sum_{i \in V(G)} x_i \right)^2}{\sum_{i \in V(G)} x_i^2 + 2 \sum_{(i,j) \in E(G)} x_i x_j} \quad (3)$$

$$\alpha(G) = \max_{x \in [0,1]^n} \sum_{i \in V(G)} x_i \prod_{j \in N(i)} (1 - x_j) \quad (4)$$

$$\alpha(G) = \max_{x \in [0,1]^n} \sum_{i \in V(G)} x_i - \sum_{(i,j) \in E(G)} x_i x_j \quad (5)$$

$$\alpha(G) = \max_{x \in [0,1]^n} \sum_{i \in V(G)} \frac{x_i}{1 + \sum_{j \in N(i)} x_j} \quad (6)$$

Proof: Formulation (3) can be obtained from the Motzkin-Straus theorem (see [22]). Statements (4) and (5) were proved by Harant et al. using probabilistic methods in [23] and [22] respectively. These were also proved deterministically in [16]. Note that formulation (5) has been recently used by de Angelis et al. [19] in numerical experiments with their approach to box-constrained quadratic optimization. Formulation (6) can be deduced from a more “complicated” result in [22]. Below we provide an alternative deterministic proof.

Denote by $f(x)$ the objective function in (6), *i.e.*,

$$f(x) = \sum_{i \in V} \frac{x_i}{1 + \sum_{j \in N(i)} x_j}.$$

Let

$$f(G) = \max_{0 \leq x_i \leq 1, i=1, \dots, n} f(x).$$

We need to show that $f(G) = \alpha(G)$. Note that $f(x)$ is a continuous function and $[0, 1]^n = \{(x_1, x_2, \dots, x_n) : 0 \leq x_i \leq 1, i = 1, \dots, n\}$ is a compact set. Hence, there always exists $x^* \in [0, 1]^n$ such that $f(G) = f(x^*)$.

Next we show that (6) always has an optimal 0-1 solution. Partition V into three disjoint sets as follows, for some fixed $k \in V$:

$$V = \{k\} \cup N(k) \cup S,$$

where $S = V \setminus (k \cup N(k))$. We can rewrite $f(x)$ in the form

$$f(x) = x_k A_k(x) + B_k(x) + C_k(x) \quad (7)$$

where

$$A_k(x) = \frac{1}{1 + \sum_{j \in N(k)} x_j} \quad (8)$$

$$B_k(x) = \sum_{i \in N(k)} \frac{x_i}{x_k + 1 + \sum_{j \in N(i) \setminus \{k\}} x_j} \quad (9)$$

$$C_k(x) = \sum_{i \in S} \frac{x_i}{1 + \sum_{j \in N(i)} x_j} \quad (10)$$

We now show that $f(x)$ is a convex function with respect to each variable (x_k). For any $x \in [0, 1]^n$, we fix x_i for all $i \neq k$, and treat $f(x)$ as a function of single variable $x_k \in [0, 1]$. Observe that $x_k A_k(x) + C_k(x)$ is linear with respect to x_k as $A_k(x), C_k(x)$ are independent of x_k and hence is convex. $B_k(x)$ is a sum of functions of the form

$$g_i(x_k) = \frac{a_i}{x_k + b_i},$$

where $0 \leq a_i \leq 1, b_i = 1 + \sum_{j \in N(i) \setminus \{k\}} x_j$, and a_i, b_i do not depend on x_k . The above function is convex in the region $x_k > -b_i$, so

$$\frac{x_i}{x_k + 1 + \sum_{j \in N(i) \setminus \{k\}} x_j}$$

is convex as a function of x_k for $x_k > -1 - \sum_{j \in N(i) \setminus \{k\}} x_j$, and hence it is convex for $x_k \in [0, 1]$. Thus $B_k(x)$ is also convex as a function of x_k . Therefore,

$$f(x) = x_k A_k(x) + B_k(x) + C_k(x) \quad (11)$$

is a convex function with respect to x_k . In fact, $f(x)$ is strictly convex with respect to x_k , unless $x_i = 0$ for all $i \in N(k)$. A strictly convex function over a closed interval can have a local maximum only at an endpoint of the interval

(in our case 0 or 1). On the other hand, if $x_i = 0$ for all $i \in N(k)$ then $f(x) = x_k + C_k(x)$ is linear with respect to x_k and its only local maximum as a function of $x_k \in [0, 1]$ is attained at $x_k = 1$. Thus, any local maximum of $f(x)$ as a function of x_k would be attained at $x_k = 0$ or 1, the boundary points. This is true for every $k \in V$ and hence any local maximizer x^* of (6) is a 0-1 vector, *i.e.* $x^* \in \{0, 1\}^n$.

We have now shown that $\exists x^* \in \{0, 1\}^n$ such that $f(G) = f(x^*)$. To prove that $f(G) = \alpha(G)$, we establish the inequality in both directions.

1. $f(G) \geq \alpha(G)$.

Let I^* be a maximum independent set of G . Construct a feasible solution x^0 of (6) as follows,

$$x_i^0 = \begin{cases} 1, & \text{if } i \in I^* \\ 0, & \text{otherwise} \end{cases}$$

Then we have $f(x^0) = \alpha(G)$ and hence $f(G) \geq f(x^0) = \alpha(G)$.

2. $f(G) \leq \alpha(G)$.

Let x^* be an optimal 0-1 solution to (6). Let $f(G) = f(x^*)$. Without loss of generality we can assume that the optimal solution is $x_1^* = x_2^* = \dots = x_r^* = 1; x_{r+1}^* = x_{r+2}^* = \dots = x_n^* = 0$, for some r . Let $V^* = \{1, \dots, r\}$. Then we have:

$$f(G) = f(x^*) = \sum_{i \in V^*} \frac{x_i^*}{1 + \sum_{j \in N(i)} x_j^*} = \sum_{i \in V^*} \frac{1}{1 + |N(i) \cap V^*|}, \quad (12)$$

since $\sum_{j \in N(i)} x_j^* = |N(i) \cap V^*|$. Now consider $\tilde{G} = G(V^*)$, the subgraph induced by V^* .

Wei's lower bound on the independence number $\alpha(G)$ of an arbitrary graph G is given by the following inequality [26, 27]:

$$\alpha(G) \geq \sum_{i \in V(G)} \frac{1}{1 + d_i} \quad (13)$$

Using (13) for \tilde{G} , we obtain

$$\alpha(\tilde{G}) \geq \sum_{i \in V^*} \frac{1}{1 + |N_{\tilde{G}}(i)|} = f(G). \quad (14)$$

Here we used (12) and the observation that

$$\forall i \in V^* : |N(i) \cap V^*| = |N_{\tilde{G}}(i)|$$

But $\alpha(G) \geq \alpha(\tilde{G}) \geq f(G)$. Thus $\alpha(G) \geq f(G)$ which establishes the result in (6). \square

We will say that a 0-1 solution x^* of (6) corresponds to a set of vertices $W \subseteq V$ of G if $W = \{i : x_i^* = 1\}$. Note that an optimal solution to the problem does not necessarily correspond to a maximum independent set. Although there does exist at least one that does correspond to a maximum independent set, it is possible for alternate optima to exist. For instance, when $G = K_n$, the complete n -vertex graph, then $x_i^* = 1, i = 1, \dots, n$ is optimal. This may be the case even for graphs that are not complete. Consider the example in Figure 1. It can be easily verified that $\alpha(G) = 3$ and corresponds to $\{1,5,7\}, \{2,5,7\}, \{3,5,7\}, \{2,4,6\}$. But the objective function attains optimum for $\{1,2,3,5,7\}, \{1,2,5,7\}, \{2,3,5,7\}, \{1,3,5,7\}$ also, even though these are not independent sets.

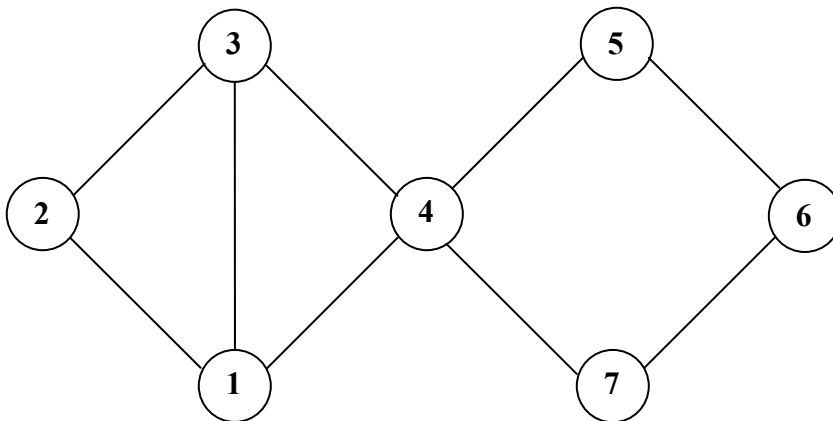


Figure 1: An example of a graph for which an optimal solution to (6) does not necessarily correspond to an independent set.

From the above proof we know that $\alpha(G) = f(G)$ and $\alpha(G) \geq \alpha(\tilde{G}) \geq f(G)$. Hence we have,

$$\alpha(G) = \alpha(\tilde{G}) = f(G) = \sum_{i \in V^*} \frac{1}{1 + |N_{\tilde{G}}(i)|} \quad (15)$$

Since Wei's lower bound (13) is sharp on \tilde{G} , it is a disjoint union of cliques [28]. Also, in the proof we have shown that any local maximum of (6) is a 0-1 vector. Hence any global optimal solution to the optimization problem (6) corresponds to a subset of vertices of the graph G which is a disjoint union of cliques. A maximum independent set can then be extracted by choosing one vertex from each of these cliques.

3.2 MAX CUT Problem

Similar to the maximum independent set problem, the MAX CUT problem can be formulated in terms of optimization of a quadratic over the unit hypercube.

Theorem 3.3 Given a graph $G = (V, E)$ with vertex set $V = \{1, 2, \dots, n\}$ and a set of weighted edges E , the optimal objective function value of MAX CUT problem is given by

$$\max_{x \in [0,1]^n} x^T W(e - x), \quad (16)$$

where $W = [w_{ij}]_{i,j=1}^n$ is the matrix of edge weights (with zero diagonal), and $e = [1, 1, \dots, 1]^T$ is the unit vector.

Proof: Denote the objective of (16) by

$$f(x) = x^T W(e - x).$$

First, since W has zero diagonal, $f(x)$ is linear with respect to each variable, and there always exist an optimal solution $\hat{x} \in \{0, 1\}^n$ of (16), *i.e.*,

$$\max_{x \in [0,1]^n} x^T W(e - x) = \max_{x \in \{0,1\}^n} x^T W(e - x).$$

Next, for any binary vector $\bar{x} \in \{0, 1\}^n$, consider the partition of the set of vertices into two nonoverlapping sets $V_1 = \{i | \bar{x}_i = 0\}$ and $V_2 = \{i | \bar{x}_i = 1\}$. Then

$$f(\bar{x}) = \sum_{(i,j) \in V_1 \times V_2} w_{ij},$$

which is the value of the cut defined by the partition $V = V_1 \cup V_2$.

Alternatively, we consider any partition of vertex set into two nonoverlapping sets,

$$V = V_1 \cup V_2, \quad V_1 \cap V_2 = \emptyset,$$

and construct the vector \bar{x} , such that

$$\bar{x}_i = \begin{cases} 0, & \text{if } i \in V_1, \\ 1, & \text{if } i \in V_2. \end{cases}$$

Then, again

$$f(\bar{x}) = \sum_{(i,j) \in V_1 \times V_2} w_{ij}.$$

Therefore, there is a one-to-one correspondence between binary vectors of length n and cuts in G . So,

$$\max_{x \in [0,1]^n} x^T W(e - x) = \max_{x \in \{0,1\}^n} x^T W(e - x) = \max_{V = V_1 \cup V_2, V_1 \cap V_2 = \emptyset} \sum_{(i,j) \in V_1 \times V_2} w_{ij}$$

□

4 Sample Test Functions and Numerical Experiments

Given a graph, the objective functions in formulations from the previous section can be used as test functions for constrained global optimization if $\alpha(G)$ or optimal MAX CUT value or their estimates are known. For small graphs these values are easily computable either by hand or using exact algorithms for the considered optimization problems on graphs. As we will demonstrate later, even for very small graphs some of these functions are nontrivial and can be used for testing purposes. However, one may wish to further “complicate” the functions available from the formulations without changing their optimal objective. Next, we briefly discuss some simple ideas along these lines.

Assume that the global maximum of $|f(x)|$, $x \in [0, 1]^n$ is attained at x^* , which is known and $f(x^*) > 0$. Let $g(x)$ be another function, such that $g(x^*) = \max_{x \in [0, 1]^n} |g(x)|$ and $g(x^*) > 0$. If we denote by $h(x) = f(x)g(x)$, $p(x) = f(x) + g(x)$ $x \in [0, 1]^n$, then

$$h(x^*) = f(x^*)g(x^*) = \max_{x \in [0, 1]^n} h(x),$$

$$p(x^*) = f(x^*) + g(x^*) = \max_{x \in [0, 1]^n} p(x),$$

i.e., we obtained other functions with known global maximum. In particular, we can use $f(x)$ and $g(x)$ from two different formulations for the independence number of some graph G with known $\alpha(G)$. Other functions can be constructed by introducing a change of the variables which would not modify the optimal objective. Some specific techniques for introducing additional local minima to a function by using “distorting” polynomials can be found in [10].

For sample numerical experiments with box-constrained problems we used the MATLAB[®] implementation of Multilevel Coordinate Search (MCS) [29] which is available online from [5]. MCS uses function values only and combines global and local search in an attempt to find a global minimum of a given function over a box. Experimentally, MCS is known to have performed better than other global optimization algorithms on many standard test problems and was particularly effective with low-dimensional problems [29, 5]. The numerical results are given in the tables below. We used MCS with default settings in all the experiments. We considered 30 maximum independent set problem instances and 10 MAX-CUT instances. The number of vertices in the graphs (*i.e.*, the number of variables in the corresponding formulations) ranged from 5 to 64. We experimented with four different formulations of the maximum independent set problem and one MAX-CUT formulation, thus providing the total of 130 test functions.

Table 1 contains the results of executing MCS on the maximum independent set problem instances using formulations (3)-(6). The columns in each row of this table contain the graph name (“Graph”) followed by the number of vertices (“|V|”), the number of edges (“|E|”), and the independence number of

the graph (“ $\alpha(G)$ ”). The remaining four columns contain the objective value of the solutions output by MCS for the four different functions corresponding to formulations (3)-(6). The graphs we used as the maximum independent set instances can be subdivided into several “families” (this fact reflects in their names). The graphs whose names start with “g” were constructed by the authors. The extension “.c” in the name of a graph `graphname.c` signifies that this graph is the complement of `graphname`. The list of edges for each of these graphs is given in the Appendix. Graph `MANN_a9`, as well as generators for `johnson`, `hamming` and `san` families can be downloaded from [30]. A description of the generators can also be found in [31]. Moreover, `johnson` and `hamming` graphs can be easily generated knowing their definitions. The vertices of `johnsonN-W-D` correspond to binary vectors of length N and weight W , with two vertices adjacent if the Hamming distance between the corresponding vectors is at least D . Similarly, `hammingN-D` has all binary vectors of length N as its vertices and the edges correspond to pairs of vertices with Hamming distance of at least D . Even though the `san` graph generator is available online, its output depends on random numbers, therefore, we provide the list of edges for this family of graphs in the Appendix. Finally, the graphs `1dc64` and `1et64` are available online from [32]. These graphs, as well as `johnson` and `hamming` families arise in coding theory (see also [33]).

The results of applying MCS to MAX-CUT formulation (16) are given in Table 2. Similar to Table 1, the first two columns in this table represent graph name and its number of vertices, while the other two columns contain the optimal objective value (“Opt”) and the terminal MCS objective value (“MCS”). All graphs used in this table were generated by the authors and the corresponding list of edge weights is provided in the Appendix.

As can be seen from the tables, the performance of MCS on the proposed set of functions was rather encouraging. In summary, global optima were found in 52 of the 130 cases and the objective was close to optimal in most cases that were not solved to optimality. However, even some of the smaller problems (*i.e.*, 5 and 8-vertex graphs in Table 2) appeared to be challenging for MCS.

In addition to the proposed examples, there are numerous larger-scale test instances for the maximum clique and maximum independent set problems which could be used for constructing large-scale test functions for global optimization. Graphs of various sizes available from [30, 32] arise in different important application areas. For all these graphs either optimal or the best known solutions are given. Test problems from the TSPLIB [34] are often used as test instances for MAX CUT problem (see, *e.g.*, [20]).

5 Conclusion

In this paper, we proposed a framework for constructing test functions for global optimization which utilizes continuous formulations of combinatorial optimization problems, namely the maximum independent set, maximum clique and MAX CUT problems on graphs. Due to the computational intractability of these

problems, their continuous formulations provide challenging test functions for constrained global optimization algorithms. On the other hand, development of efficient approaches for solving these types of global optimization problems may lead to new algorithms for the mentioned combinatorial optimization problems.

Even though we restricted our attention to only three graph problems and only linearly-constrained formulations, there are many other opportunities for use of combinatorial optimization problems to construct test functions for global optimization. Since many of combinatorial optimization problems can be expressed in terms of integer programs, the constraint $x \in \{0, 1\}^n$ can be replaced with the equivalent quadratic constraints $x_i(1 - x_i) = 0$, $i = 1, \dots, n$ [35], thus yielding a continuous constrained global optimization formulation. For example, Shor [36] used a similar idea to obtain a quadratically constrained global optimization formulation of the maximum weight independent set problem. Other interesting problems which could be used for testing purposes include graph coloring [37], quadratic assignment and maximum matching problems (see [38] for a convex quadratic approach to the maximum matching problem).

6 Acknowledgements

We would like to thank Arnold Neumaier for making MCS package available and the two anonymous referees for their valuable comments.

References

- [1] Pintér, J. D., 2002, Global optimization: software, test problems, and applications. In: P. M. Pardalos and H. E. Romeijn (Eds) *Handbook of Global Optimization*, **2** (Dordrecht, The Netherlands: Kluwer Academic), pp. 515–569.
- [2] Moré, J.J., Garbow, B.S. and Hillström, K.E., 1981, Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, **7**, 17–41.
- [3] Dixon, L. C. W. and Szegö, G. P., 1975, 1978, *Towards Global Optimisation*, **1–2** (Amsterdam, The Netherlands: North-Holland).
- [4] Hock, W. and Schittkowski, K., 1981, Test Examples for Nonlinear Programming Codes. *Lecture Notes in Economics and Mathematical Systems*, **187** (Berlin / Heidelberg / New York: Springer-Verlag).
- [5] Neumaier, A., *Global Optimization Website*. Available online at: <http://www.mat.univie.ac.at/~neum/glopt.html> (accessed November 2004).
- [6] Floudas, C. A., Pardalos, P. M., Adjiman, C. S., Esposito, W. R., Gumus, Z. H., Harding, S. T., Klepeis, J. L., Meyer, C. A. and Schweiger, C. A.

- (Eds), 1999, *Handbook of Test Problems in Local and Global Optimization* (Dordrecht, The Netherlands: Kluwer Academic).
- [7] Vicente, L. N., Calamai, P. H. and Judice, J. J., 1992, Generation of disjointly constrained bilinear programming test problems. *Computational Optimization and Applications*, **1**, 299–306.
- [8] Calamai, P. H. , Vicente, L. N. and Judice, J. J., 1993, A new technique for generating quadratic programming test problems. *Mathematical Programming*, **61**, 215–231.
- [9] Motzkin, T. S. and Straus, E. G., 1965, Maxima for graphs and a new proof of a theorem of Turán. *Canadian Journal of Mathematics*, **17**, 533–540.
- [10] Gaviano, M., Kvasov, D. E., Lera, D. and Sergeyev, Y. D., 2003, Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. *ACM Transactions on Mathematical Software*, **29**, 469–480.
- [11] Garey, M. R. and Johnson, D. S., 1979, *Computers and Intractability: A Guide to the Theory of NP-completeness* (New York: W .H. Freeman and Company).
- [12] Bomze, I. M., Budinich, M., Pardalos, P. M. and Pelillo, M., 1999, The maximum clique problem. In: D.-Z. Du and P. M. Pardalos (Eds) *Handbook of Combinatorial Optimization* (Dordrecht, The Netherlands: Kluwer Academic), pp. 1–74.
- [13] Deza, M. and Laurent, M., 1997, *Geometry of Cuts and Metrics* (New York: Springer-Verlag).
- [14] Festa, P., Pardalos, P. M., Resende, M. G. C. and Ribeiro, C. C., 2002, Randomized heuristics for the MAX-CUT problem. *Optimization Methods and Software*, **7**, 1033–1058.
- [15] Hager, W. and Krylyuk, Y., 1999, Graph partitioning and continuous quadratic programming. *SIAM Journal on Discrete Mathematics*, **12**, 500–523.
- [16] Abello, J., Butenko, S., Pardalos, P. M. and Resende M. G. C., 2001, Finding independent sets in a graph using continuous multivariable polynomial formulations. *Journal of Global Optimization*, **21**, 111–137.
- [17] Burer, S., Monteiro, R. D. C. and Zhang, Y., 2001, Rank-two relaxation heuristics for MAX-CUT and other binary quadratic programs. *SIAM Journal on Optimization*, **12**, 503–521.
- [18] Busygin, S., Butenko, S. and Pardalos, P. M., 2002, A heuristic for the maximum independent set problem based on optimization of a quadratic over a sphere. *Journal of Combinatorial Optimization*, **6**, 287–297.

- [19] de Angelis, P. L., Bomze, I. M. and Toraldo, G., 2004, Ellipsoidal approach to box-constrained quadratic problems. *Journal of Global Optimization*, **28**, 1–15.
- [20] Goemans, M. X. and Williamson, D. P., 1995, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, **42**, 1115–1145.
- [21] Harant, J., 1998, A lower bound on the independence number of a graph. *Discrete Mathematics*, **188**, 239–243.
- [22] Harant, J., 2000, Some news about the independence number of a graph. *Discussiones Mathematicae Graph Theory*, **20**, 71–79.
- [23] Harant, J., Pruchnewski, A. and Voigt, M., 1999, On dominating sets and independent sets of graphs. *Combinatorics, Probability and Computing*, **8**, 547–553.
- [24] Bomze, I. M., 1997, Evolution towards the maximum clique. *Journal of Global Optimization*, **10**, 143–164.
- [25] Moon, J. W. and Moser, L., 1965, On cliques in graphs. *Israel Journal of Mathematics*, **3**, 23–28.
- [26] Caro, Y. and Tuza, Z., 1991, Improved lower bounds on k -independence, *Journal of Graph Theory*, **15**, 99–107.
- [27] Wei, V. K., 1981, A lower bound on the stability number of a simple graph. Technical Report, **TM 81-11217-9**, Bell Laboratories, Murray Hill, NJ.
- [28] Selkow, S. M., 1994, A Probabilistic lower bound on the independence number of graphs. *Discrete Mathematics*, **132**, 363–365.
- [29] Huyer, W. and Neumaier, A., 1999, Global optimization by multilevel coordinate search, *Journal of Global Optimization*, **14**, 331–355.
- [30] DIMACS, 1995, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*. Available online at: <ftp://dimacs.rutgers.edu/pub/challenge/> (accessed November 2004).
- [31] Hasselberg, J., Pardalos, P. M. and Vairaktarakis, G., 1993, Test case generators and computational results for the maximum clique problem. *Journal of Global Optimization*, **3**, 463–482.
- [32] Sloane, N., 2000, *Challenge Problems: Independent Sets in Graphs*. Available online at: <http://www.research.att.com/~njas/doc/graphs.html> (accessed November 2004).
- [33] Butenko, S., Pardalos, P. M., Sergienko, I. V., Shylo, V. and Stetsyuk, P., Estimating the size of correcting codes using extremal graph problems. In: E. Hunt and C. Pearce (Eds) *Optimization: Structure and Applications*, In press (Dordrecht, The Netherlands: Kluwer Academic).

- [34] Reinelt, G., 1991, TSPLIB - A Traveling Salesman Problem Library. *ORSA Journal on Computing*, **3**, 376–384.
- [35] Horst, R., Pardalos, P. M. and Thoai, N. V., 2000, *Introduction to Global Optimization* (2nd edn) (Dordrecht, The Netherlands: Kluwer Academic).
- [36] Shor, N. Z., 1990, Dual quadratic estimates in polynomial and Boolean programming. *Annals of Operations Research*, **25**, 163–168.
- [37] Butenko, S., Festa, P. and Pardalos, P. M., 2001, On the chromatic number of graphs. *Journal of Optimization Theory and Applications*, **109**, 51–67.
- [38] Cardoso, D. M., 2001, Convex quadratic programming approach to the maximum matching problem. *Journal of Global Optimization*, **21**, 91–106.

Appendix: Graph Instances

This appendix contains the graph data used in our numerical experiments reported in Section 4. All these data are available in electronic format from the authors.

Maximum Independent Set Problem Instances

Below, $E_{\text{graphname}}$ represents the set of edges of graph **graphname** from Table 1.

$$E_{g7-2} = \{(1, 2), (1, 5), (1, 6), (1, 7), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7), (3, 4), (3, 5), (3, 6), (3, 7), (4, 5), (4, 6), (4, 7)\}.$$

$$E_{g10-1} = \{(1, 7), (1, 8), (1, 10), (2, 4), (2, 6), (3, 5), (3, 7), (4, 9), (4, 10), (5, 7), (5, 10), (6, 8), (6, 10)\}.$$

$$E_{g10-2} = \{(1, 4), (1, 5), (1, 8), (1, 10), (2, 3), (2, 5), (2, 6), (2, 7), (3, 6), (3, 8), (3, 9), (3, 10), (4, 5), (4, 7), (4, 9), (4, 10), (5, 6), (5, 7), (5, 8), (5, 10), (6, 9), (7, 8)\}.$$

$$E_{g15-1} = \{(1, 5), (1, 10), (1, 11), (1, 13), (1, 15), (2, 3), (2, 4), (2, 6), (2, 7), (2, 12), (2, 13), (2, 14), (2, 15), (3, 4), (3, 5), (3, 10), (3, 11), (3, 12), (3, 13), (3, 14), (4, 8), (4, 9), (4, 14), (4, 15), (5, 7), (5, 8), (5, 9), (5, 13), (6, 7), (6, 11), (6, 12), (6, 13), (7, 10), (7, 12), (7, 13), (7, 14), (8, 10), (8, 11), (8, 12), (8, 15), (9, 12), (9, 14), (10, 12), (10, 15), (11, 14)\}.$$

$$E_{g15-2} = \{(1, 6), (1, 14), (1, 15), (2, 10), (2, 12), (3, 5), (3, 10), (3, 12), (4, 5), (6, 13), (7, 9), (7, 13), (8, 13), (9, 11), (9, 12), (9, 13), (10, 11), (10, 15), (13, 15)\}.$$

$$E_{san15-1} = \{(1, 3), (1, 4), (1, 8), (1, 11), (1, 13), (2, 5), (2, 6), (2, 10), (3, 6), (3, 7), (3, 9), (3, 10), (3, 14), (4, 6), (4, 9), (4, 10), (4, 12), (5, 7), (5, 12), (6, 13), (6, 14), (6, 15), (7, 14), (7, 15), (8, 11), (8, 12), (8, 13), (9, 15), (10, 13), (10, 15), (11, 13), (11, 14), (11, 15), (12, 14), (12, 15)\}.$$

$$E_{san15-2.c} = \{(1,3), (1,6), (2, 5), (2, 7), (2, 15), (3, 4), (4, 8), (4, 12), (5, 8), (5, 10), (6, 7), (6, 9), (6, 13), (6, 15), (7, 8), (8, 9), (9, 14), (10, 15), (11, 13), (12, 13)\}.$$

$E_{san30-1.c}=(1, 2), (1, 23), (2, 3), (3, 12), (3, 16), (3, 24), (4, 5), (4, 7), (4, 8), (4, 18), (4, 28), (5, 6), (5, 11), (5, 22), (5, 24), (6, 7), (6, 20), (7, 11), (7, 26), (8, 30), (9, 11), (9, 19), (10, 12), (10, 14), (10, 23), (11, 17), (12, 15), (12, 17), (12, 27), (13, 17), (13, 18), (15, 16), (16, 18), (17, 26), (18, 19), (18, 26), (19, 27), (20, 23), (21, 25), (22, 27), (23, 28), (24, 29), (26, 29)\}.$

$E_{san40-1.c}=\{(1, 4), (1, 8), (1, 9), (1, 13), (1, 35), (2, 12), (2, 30), (2, 38), (3, 15), (4, 14), (4, 31), (4, 33), (4, 34), (5, 33), (5, 34), (6, 10), (6, 21), (7, 19), (7, 26), (7, 36), (8, 26), (8, 28), (8, 31), (8, 33), (9, 23), (10, 19), (10, 31), (10, 33), (10, 36), (10, 40), (11, 17), (11, 39), (12, 16), (12, 17), (12, 19), (12, 31), (12, 33), (13, 15), (13, 17), (13, 20), (13, 27), (13, 34), (13, 36), (13, 40), (15, 21), (16, 22), (16, 25), (16, 32), (16, 38), (17, 30), (18, 27), (18, 34), (18, 40), (19, 22), (19, 30), (19, 35), (20, 25), (20, 32), (22, 34), (22, 37), (24, 27), (24, 31), (24, 36), (25, 26), (25, 31), (25, 36), (25, 37), (25, 39), (26, 30), (27, 38), (29, 34), (29, 36), (30, 33), (30, 37), (30, 39), (31, 35), (32, 33), (38, 40)\}.$

$E_{san40-2.c}=\{(1, 3), (1, 10), (1, 17), (2, 4), (2, 5), (2, 8), (3, 18), (3, 22), (4, 15), (4, 16), (4, 19), (5, 14), (5, 29), (5, 31), (6, 19), (6, 31), (6, 34), (6, 39), (7, 32), (8, 17), (8, 29), (9, 12), (9, 13), (9, 23), (9, 30), (10, 23), (10, 38), (11, 15), (11, 19), (11, 21), (11, 36), (12, 16), (12, 17), (12, 33), (12, 34), (13, 15), (13, 21), (13, 36), (14, 24), (14, 27), (14, 38), (15, 38), (16, 18), (16, 26), (17, 23), (18, 36), (18, 39), (19, 25), (19, 40), (20, 21), (20, 32), (20, 35), (21, 23), (21, 25), (21, 26), (21, 28), (22, 36), (23, 29), (23, 32), (23, 37), (24, 33), (25, 34), (27, 36), (27, 39), (28, 32), (28, 33), (30, 31), (30, 34), (30, 36), (31, 38), (32, 38), (35, 40), (37, 40)\}.$

$E_{san50-1.c}=\{(1, 18), (1, 45), (2, 8), (2, 19), (2, 34), (2, 38), (2, 40), (3, 12), (3, 17), (3, 28), (3, 41), (3, 45), (4, 9), (4, 19), (4, 24), (4, 44), (4, 48), (5, 21), (5, 32), (5, 37), (5, 38), (5, 50), (6, 13), (6, 21), (6, 41), (6, 42), (7, 13), (7, 33), (7, 35), (7, 36), (8, 34), (9, 14), (9, 15), (9, 31), (9, 47), (9, 48), (10, 17), (10, 29), (10, 30), (10, 44), (10, 48), (11, 20), (11, 27), (11, 35), (12, 28), (12, 32), (13, 15), (13, 19), (13, 24), (13, 27), (13, 32), (13, 35), (13, 36), (13, 38), (13, 49), (13, 50), (14, 38), (14, 42), (15, 27), (15, 30), (15, 35), (15, 40), (15, 43), (15, 49), (16, 18), (16, 22), (16, 26), (16, 46), (17, 32), (17, 45), (18, 19), (18, 27), (18, 35), (18, 39), (18, 44), (18, 45), (19, 31), (19, 40), (19, 47), (20, 25), (20, 29), (20, 30), (21, 37), (21, 50), (22, 35), (22, 36), (22, 46), (23, 27), (23, 31), (23, 35), (23, 43), (24, 30), (24, 31), (25, 27), (25, 29), (25, 31), (25, 50), (26, 27), (26, 32), (26, 39), (27, 40), (27, 41), (28, 30), (28, 40), (28, 44), (29, 40), (30, 37), (30, 39), (30, 41), (30, 43), (30, 45), (31, 40), (31, 47), (32, 40), (32, 44), (33, 35), (33, 36), (35, 42), (38, 42), (39, 41), (40, 43), (40, 45), (42, 45), (46, 48), (48, 50)\}.$

$E_{san50-2.c}=\{(1, 32), (1, 40), (2, 47), (3, 6), (3, 39), (4, 33), (4, 45), (5, 17), (5, 29), (6, 9), (6, 20), (6, 29), (7, 8), (7, 38), (7, 49), (8, 14), (8, 26), (8, 39), (8, 44), (9, 19), (9, 29), (10, 21), (10, 48), (11, 20), (11, 40), (11, 42), (12, 19), (12, 41), (13, 15), (14, 37), (14, 45), (16, 34), (16, 37), (18, 23), (18, 35), (19, 41), (20, 44), (21, 24), (21, 43), (22, 28), (22, 30), (23, 37), (23, 42), (24, 49), (25, 30), (25, 43), (26, 40), (26, 41), (26, 46), (27, 46), (30, 46), (30, 49), (31, 42), (32, 44), (34, 35), (35, 40), (35, 45), (36, 43), (38, 50), (41, 43), (42, 47), (43, 48), (45, 47), (47, 49), (49, 50)\}.$

MAX-CUT Instances

This subsection contains the list of edge weights for each graph in Table 2, where $W_{\text{graphname}}$ represents the set of edge weights of graph **graphname**. All of the graphs are assumed to be complete. Each set of weights contains subsets separated by semicolons. The i^{th} such subset contains the weights of edges $(i, i+1), (i, i+2), \dots, (i, n)$ separated by spaces (here we assume that the graph has n vertices). In other words, $W_{\text{graphname}}$ can be viewed as a list of entries of the upper triangular part of **graphname**'s edge weight matrix, read line by line.

$$W_{G5-1} = \{23\ 16\ 18\ 17; 25\ 26\ 23; 14\ 24; 10\}.$$

$$W_{G5-2} = \{6\ 6\ 8\ 10; 5\ 5\ 6; 4\ 6; 6\}.$$

$$W_{G8-1} = \{85\ 141\ 186\ 108\ 107\ 127\ 47; 107\ 105\ 64\ 146\ 132\ 111; 142\ 163\ 92\ 80\ 106; 145\ 89\ 67\ 98; 83\ 94\ 102; 126\ 131; 116\}.$$

$$W_{G8-2} = \{96\ 116\ 123\ 38\ 81\ 110\ 160; 45\ 46\ 104\ 137\ 60\ 100; 73\ 133\ 48\ 74\ 51; 64\ 109\ 111\ 103; 152\ 103\ 95; 132\ 43; 102\}.$$

$$W_{G10-1} = \{69\ 58\ 85\ 64\ 53\ 42\ 56\ 63\ 33; 74\ 21\ 84\ 63\ 59\ 32\ 18\ 41; 39\ 48\ 22\ 42\ 88\ 41\ 73; 43\ 57\ 23\ 85\ 86\ 46; 37\ 28\ 40\ 69\ 67; 34\ 65\ 56\ 65; 71\ 58\ 72; 44\ 56; 69\}.$$

$$W_{G10-2} = \{61\ 45\ 77\ 56\ 70\ 2\ 30\ 43\ 33; 66\ 80\ 46\ 74\ 21\ 71\ 24\ 66; 78\ 68\ 18\ 36\ 36\ 59\ 40; 46\ 41\ 28\ 42\ 20\ 41; 50\ 42\ 32\ 39\ 27; 64\ 81\ 84\ 79; 64\ 57\ 44; 58\ 69; 13\}.$$

$$W_{G20-1} = \{1\ 2\ 3\ 3\ 2\ 4\ 4\ 2\ 3\ 1\ 0\ 2\ 3\ 5\ 2\ 2\ 3\ 3\ 1; 3\ 1\ 2\ 2\ 3\ 2\ 1\ 3\ 4\ 5\ 2\ 1\ 3\ 2\ 4\ 4\ 3\ 1; 0\ 3\ 5\ 3\ 1\ 1\ 2\ 3\ 3\ 2\ 3\ 2\ 2\ 2\ 3\ 2\ 3; 2\ 1\ 3\ 4\ 3\ 3\ 3\ 1\ 2\ 2\ 4\ 1\ 3\ 4\ 3\ 4; 5\ 3\ 1\ 0\ 4\ 2\ 4\ 4\ 1\ 4\ 3\ 0\ 2\ 0\ 2; 3\ 1\ 2\ 3\ 2\ 1\ 2\ 2\ 3\ 3\ 2\ 2\ 3\ 2; 1\ 2\ 2\ 0\ 3\ 0\ 3\ 3\ 3\ 2\ 2\ 4\ 3; 1\ 4\ 1\ 3\ 4\ 1\ 4\ 3\ 4\ 3\ 5\ 2; 1\ 0\ 3\ 3\ 2\ 3\ 2\ 2\ 4\ 3\ 1; 1\ 2\ 4\ 4\ 3\ 3\ 1\ 1\ 2\ 4; 2\ 4\ 2\ 0\ 3\ 0\ 1\ 3\ 1; 0\ 4\ 3\ 2\ 2\ 3\ 3\ 3; 1\ 0\ 4\ 3\ 1\ 3\ 4; 2\ 3\ 1\ 3\ 2\ 2; 2\ 1\ 4\ 5\ 1; 2\ 1\ 0\ 1; 0\ 4\ 5; 1\ 1; 1\}.$$

$$W_{G15-1} = \{7\ 6\ 7\ 10\ 10\ 6\ 10\ 4\ 10\ 2\ 5\ 4\ 5\ 3; 4\ 1\ 7\ 6\ 5\ 9\ 4\ 12\ 5\ 7\ 3\ 1\ 6; 3\ 2\ 2\ 6\ 8\ 6\ 10\ 3\ 4\ 7\ 7\ 3; 5\ 6\ 5\ 3\ 8\ 4\ 8\ 4\ 12\ 11\ 13; 13\ 8\ 10\ 5\ 5\ 6\ 8\ 9\ 5\ 3; 5\ 3\ 6\ 6\ 12\ 6\ 7\ 3\ 8; 12\ 3\ 5\ 6\ 7\ 3\ 3\ 4; 4\ 7\ 7\ 4\ 9\ 6\ 10; 8\ 1\ 5\ 4\ 3\ 9; 9\ 9\ 4\ 5\ 10; 9\ 8\ 6\ 6; 8\ 5\ 6; 13\ 1; 2\}.$$

$$W_{G15-2} = \{3\ 8\ 14\ 14\ 13\ 9\ 13\ 8\ 2\ 10\ 12\ 9\ 12\ 11; 10\ 5\ 16\ 8\ 12\ 7\ 7\ 11\ 10\ 10\ 15\ 16\ 3; 7\ 8\ 17\ 5\ 7\ 8\ 7\ 6\ 15\ 13\ 11\ 11; 11\ 11\ 8\ 15\ 7\ 11\ 10\ 12\ 14\ 4\ 8; 8\ 17\ 13\ 7\ 14\ 7\ 7\ 8\ 11\ 13; 5\ 11\ 7\ 3\ 3\ 10\ 12\ 8\ 5; 9\ 13\ 8\ 8\ 6\ 5\ 16\ 7; 16\ 6\ 7\ 13\ 12\ 9\ 2; 9\ 4\ 15\ 9\ 10\ 6; 1\ 10\ 13\ 17\ 13; 9\ 14\ 8\ 5; 17\ 15\ 12; 4\ 11; 2\}.$$

$$W_{G20-2} = \{4\ 1\ 1\ 0\ 1\ 3\ 3\ 3\ 4\ 5\ 3\ 4\ 1\ 3\ 2\ 1\ 2\ 4\ 1; 3\ 3\ 3\ 3\ 3\ 2\ 2\ 2\ 3\ 1\ 2\ 1\ 4\ 2\ 2\ 3\ 0\ 3; 4\ 1\ 2\ 2\ 5\ 2\ 3\ 2\ 3\ 2\ 3\ 4\ 5\ 1\ 3\ 4\ 4; 3\ 3\ 0\ 2\ 1\ 2\ 3\ 3\ 0\ 4\ 4\ 2\ 5\ 1\ 3\ 4; 3\ 0\ 2\ 3\ 2\ 3\ 2\ 4\ 2\ 3\ 2\ 3\ 4\ 3\ 2; 2\ 3\ 3\ 3\ 3\ 4\ 5\ 2\ 2\ 5\ 2\ 2\ 1\ 2; 3\ 3\ 2\ 3\ 2\ 2\ 2\ 5\ 3\ 1\ 1\ 1\ 4; 0\ 4\ 1\ 4\ 4\ 2\ 1\ 1\ 3\ 1\ 2\ 2; 1\ 4\ 3\ 2\ 4\ 3\ 1\ 3\ 1\ 2\ 4; 1\ 2\ 1\ 2\ 2\ 1\ 0\ 5\ 3\ 2; 4\ 3\ 5\ 2\ 1\ 3\ 2\ 1\ 2; 2\ 2\ 3\ 3\ 4\ 1\ 1\ 3; 2\ 3\ 3\ 2\ 4\ 4\ 1; 3\ 2\ 3\ 2\ 2\ 0; 2\ 5\ 1\ 3\ 3; 1\ 3\ 3\ 4; 2\ 1\ 4; 3\ 4; 4\}.$$

Table 1: Results of sample experiments carried out using MATLAB[®] implementation of MCS: Maximum independent set problem.

Graph	V	E	$\alpha(G)$	Formulation			
				(3)	(4)	(5)	(6)
g7-1	7	16	3	2.0000	3.0000	3.0000	2.0000
g7-2	7	13	3	3.0000	3.0000	2.0000	3.0000
g10-1	10	13	5	5.0000	5.0000	5.0000	5.0000
g10-1.c	10	32	3	2.0000	2.0000	3.0000	2.0000
g10-2	10	22	4	4.0000	4.0000	3.0000	4.0000
g10-2.c	10	23	4	3.0000	4.0000	4.0000	4.0000
g15-1	15	45	5	4.0000	4.0000	4.0000	4.0000
g15-1.c	15	60	4	4.0000	4.0000	4.0000	4.0000
g15-2	15	19	9	8.0000	9.0000	8.0000	9.0000
g15-2.c	15	86	3	2.0000	2.0000	2.0000	2.0000
johnson6-2-4.c	15	60	3	3.0000	3.0000	3.0000	2.3333
johnson6-3-5.c	20	180	2	1.2000	2.0000	2.0000	1.0769
johnson7-5-3.c	21	105	3	2.3333	3.0000	3.0000	2.3333
johnson8-2-4.c	28	168	4	2.4000	4.0000	4.0000	2.6667
johnson7-3-5.c	35	525	2	1.5000	2.0000	2.0000	1.3214
MANN_a9	45	918	3	3.0000	3.0000	3.0000	1.2778
MANN_a9.c	45	72	16	12.0000	16.0000	16.0000	16.0000
hamming6-2.c	64	102	32	32.0000	32.0000	32.0000	32.0000
hamming6-4.c	64	1312	4	2.0000	4.0000	4.0000	2.0220
1dc64	64	543	10	9.0000	9.0000	8.0000	10.0000
1et64	64	264	18	18.0000	18.0000	18.0000	15.6667
san15-1.c	15	70	4	2.0000	4.0000	4.0000	3.0000
san15-2.c	15	20	8	7.0000	7.0000	7.0000	8.0000
san20-1.c	20	60	8	7.0000	6.0000	6.0000	7.0000
san20-2.c	20	70	7	6.0000	6.0000	6.0000	6.0000
san30-1.c	30	43	15	13.0000	13.0000	13.0000	13.0000
san40-1.c	40	78	20	16.0000	16.0000	18.0000	17.0000
san40-2.c	40	73	21	17.2811	17.0000	18.0000	17.0000
san50-1.c	50	125	20	18.0000	18.0000	19.0000	19.0000
san50-2.c	50	65	24	22.0000	22.0000	21.0000	23.0000

Table 2: Results of sample experiments carried out using MATLAB[®] implementation of MCS: MAX-CUT problem.

Graph	$ V $	Opt	MCS
G5-1	5	126	125
G5-2	5	40	39
G8-1	8	1987	1802
G8-2	8	1688	1671
G10-1	10	1585	1513
G10-2	10	1373	1377
G15-1	15	399	389
G15-2	15	594	593
G20-1	20	273	267
G20-2	20	285	282