On greedy construction heuristics for the MAX–CUT problem

4 authors, including:

Sergiy Butenko
Texas A&M University
129 PUBLICATIONS   **3,760** CITATIONS

SEE PROFILE

I. V. Hicks
Rice University
61 PUBLICATIONS   **1,093** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Optimization problems on graphs View project

Continuous Approaches to Cliques and Stable Sets View project

On Greedy Construction Heuristics for the MAX-CUT problem

Sera Kahruman, Elif Kolotoglu, Sergiy Butenko, and Illya V. Hicks

Department of Industrial and Systems Engineering
Texas A&M University
College Station, TX 77843-3131
{sera,elif,butenko,ivhicks}@tamu.edu

Abstract: Given a graph with non-negative edge weights, the MAX-CUT problem is to partition the set of vertices into two subsets so that the sum of the weights of edges with endpoints in different subsets is maximized. This classical NP-hard problem finds applications in VLSI design, statistical physics, and classification among other fields. This paper compares the performance of several greedy construction heuristics for MAX-CUT problem. In particular, a new "worst-out" approach is studied and the proposed edge contraction heuristic is shown to have an approximation ratio of at least 1/3. The results of experimental comparison of the worst-out approach, the well-known best-in algorithm, and modifications for both are also included.

1 Introduction

Given an undirected graph $G = (V, E)$ with the set of vertices $V = \{1, 2, 3, \ldots, n\}$ and the set of edges E with weights $w_{ij} \geq 0$ for each $(i, j) \in E$, the MAX-CUT problem is to find a partition of vertices into two disjoint subsets S_1 and S_2 such that the sum of weights of the edges with endpoints in different subsets is maximized. Each partition of vertices into two subsets S_1 and S_2 with $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = V$ is called a cut, and the total weight of edges with endpoints in different subsets is called the weight of the cut or the cut value and is denoted by $cut(S_1, S_2)$:

$$cut(S_1, S_2) = \sum_{i \in S_1, j \in S_2} w_{ij}.$$

The MAX-CUT problem finds applications in statistical physics and circuit layout design [6]. Other applications include social networks, where the MAX-CUT value is generally a measure of robustness of the network [2,7], and classification [10].

Like many other graph theory problems, MAX-CUT is very easy to state but hard to solve. It is a well known NP-hard problem [23]. The MAX-CUT problem

can be formulated as the following mixed integer linear program:

$$\max \sum_{\substack{i,j=1 \\ i<j}}^{n} w_{ij}y_{ij}, \tag{1}$$

subject to:

$$y_{ij} - x_i - x_j \leq 0, \ i,j = 1,2,\ldots,n, \ i < j; \tag{1a}$$
$$y_{ij} + x_i + x_j \leq 2, \ i,j = 1,2,\ldots,n, \ i < j; \tag{1b}$$
$$x_i \in \{0,1\}, \ i = 1,2,\ldots,n. \tag{1c}$$

In the above, an optimal solution (x^*, y^*) corresponds to an optimal MAX-CUT partition of V into two subsets $S_1 = \{i : x_i^* = 0\}$ and $S_2 = \{i : x_i^* = 1\}$. A detailed polyhedral study of the problem is given in [12]. Note that the mixed integer formulation that we propose above has the same number of integer variables as the number of vertices in the graph, while the known integer programming formulations, including that proposed in [12], have a quadratic number of integer variables with respect to the number of vertices. However, the above formulation also has a quadratic number of non-integer variables in addition to the integer variables. We will also mention the following nonconvex quadratic formulation [4]. The optimal objective function value of MAX-CUT problem is given by

$$\max_{x \in [0,1]^n} x^T W(e - x), \tag{2}$$

where $W = [w_{ij}]_{i,j=1}^n$ is the matrix of edge weights (with zero diagonal), and $e = [1,1,\ldots,1]^T$ is the unit vector of length n. Similar quadratic formulations with binary variables are typically used to obtain semidefinite programming relaxations for the MAX-CUT problem and derive approximation algorithms and heuristics based on such relaxations [8, 9, 17, 25].

The known polynomially solvable cases include planar graphs [19], graphs without K_5 minors [5], and weakly bipartite graphs with nonnegative weights [18]. MAX-CUT on dense graphs [13] and Metric MAX-CUT have polynomial time randomized approximation schemes [14]. However, metric MAX-CUT is not known to be NP-hard. The general version of MAX-CUT problem is also known to be APX-complete [26], meaning that unless P=NP, it does not allow a polynomial time approximation scheme (PTAS) [28]. Thus, approximation algorithms or heuristics are used for finding acceptable solutions in polynomial time.

In this paper, we compare the performance of several greedy construction heuristics for the MAX-CUT problem. In particular, we present and study a new "worst-out" construction approach for the MAX-CUT problem, the *edge contraction heuristic*. We show that the proposed algorithm has the approximation ratio of at least 1/3. We also present an experimental comparison of solutions obtained using the edge contraction heuristic, the classical "best-in" $\frac{1}{2}$-approximation algorithm of Sahni and Gonzales [27], and modifications for both.

The remainder of this paper is organized as follows. Section 2 briefly surveys the known construction algorithms for MAX-CUT problem. The edge contraction heuristic is proposed and analyzed in Section 3. Some modified versions of the Sahni-Gonzalez algorithm are presented in Section 4. Section 5 presents the results

of experimental comparison of several greedy construction heuristics for the problem of interest in terms of the solution quality. Finally, some concluding remarks are given in Section 6.

2 Construction algorithms and their approximation ratios

Most heuristic approaches to the MAX-CUT problem consist of a construction algorithm, which builds a feasible solution from scratch, and a local search procedure that attempts to iteratively improve the current solution until a local maximum with respect to a given neighborhood is reached. In this paper we deal only with construction algorithms. More specifically, we are interested in simple greedy construction approaches and their variations. There are two major types of greedy construction algorithms for discrete optimization problems on graphs: "best in" and "worst out". A best-in algorithm typically starts with an empty graph (or a very small subgraph of the input graph), while various subgraphs of the original graph (which may be a vertex or an edge, depending on the problem) are considered to be candidates for inclusion in the constructed feasible solution. Then the algorithm successively adds a candidate, which provides the "best" contribution to the objective function value, and removes the candidates that become ineligible for inclusion from the list of candidates. The procedure is repeated until a feasible solution is constructed (for minimization problem) or the candidate list is empty. Alternatively, a worst-out algorithm usually starts with the input graph and on each step removes the part of the graph (such as a vertex or an edge), which, if included in the solution, would provide a "worst" contribution to the objective function value compared to all other candidates for removal. The algorithm stops when the remaining graph constitutes a feasible solution (for maximization problem) or any additional step would make otherwise feasible graph infeasible. Note that this description of best-in and worst-out algorithms aims to provide a general idea behind such algorithms and does not intend to be restrictive, as numerous variations of the outlined greedy approaches can still be called best-in or worst-out algorithms. This paper was partially motivated by observation that, while there are several well-known best-in algorithms for MAX-CUT problem, no results on worst-out approaches have been published to the best of our knowledge. In particular, we are interested in comparing the the proposed worst-out and the known best-in algorithms in terms of their approximation ratios and quality of the solutions obtained in numerical experiments. Next we define the concept of approximation ratio for a MAX-CUT algorithm and mention some of the known algorithms for MAX-CUT and their approximation ratios.

Let $W_\mathcal{A}(G)$ be the cut size generated by an approximation algorithm $\mathcal{A}$ for MAX-CUT problem on a graph G. The approximation ratio of the algorithm $\mathcal{A}$ is defined as the largest $R_\mathcal{A}$ for which

$$W_\mathcal{A}(G)/W^*(G) \geq R_\mathcal{A} \text{ for any graph } G,$$

where $W^*(G)$ is the optimal cut value of G. Note that $W^*(G) \leq W(G)$, where $W(G) = \sum_{i<j} w_{ij}$ is the sum of all edge weights of the graph. Thus, any R such that $W_\mathcal{A}(G)/W(G) \geq R$ for any G provides a lower bound on $R_\mathcal{A}$. Since finding

a better than $W(G)$ upper bound on the size of maximum cut may be nontrivial, this bound is frequently used to estimate an algorithm's approximation ratio.

In 1976, Sahni and Gonzalez [27] presented an algorithm that constructs an approximate solution to MAX-CUT with the approximation ratio of $1/2$. The time complexity of this algorithm is $O(|V| + |E|)$. Their algorithm starts by placing one vertex to each partition, and the remaining $|V| - 2$ vertices are examined one by one. A vertex j is assigned to a partition if the total weight of the edges in between vertex j and the vertices in that partition is minimal. This algorithm, which is perhaps the first known approximation algorithm for MAX-CUT, is still quite popular due to its simplicity and reasonably good quality of solution it guarantees. Recently, Festa et al. [15] implemented and tested a greedy randomized adaptive search procedure, a variable neighborhood search and a path-relinking intensification heuristic for MAX-CUT. In the construction phase, at each iteration, an element is randomly selected from a restricted candidate list, whose elements are ranked according to the main idea that Sahni and Gonzalez used.

Haglin and Venkatesan [20] proposed an algorithm that guarantees an approximation ratio of at least of $1/2 + \frac{1}{2|V|}$ starting with a matching of size $|E|/|V|$. Their algorithm runs in $O((|E| \log |E| + |V| \log |V|)/p + \log p \log |V|)$ parallel time using $1 <= p <= |E| + |V|$ processors. They also showed that it is NP-Complete to decide if a given graph has a maximum cut with at least a fraction $1/2 + \varepsilon$ of the sum of weights of its edges, where ε is a positive constant. Cho et al. [11] proposed an improved approximation algorithm running in $O(|E| + |V|)$ sequential time yielding a node-balanced maximum cut with size at least $W(G)(1/2 + 1/2|V|)$. Although the approximation ratio is the same as Haglin and Venkatesan's, their algorithms is better in terms of time complexity. They initialize the partitions to be empty and find a matching M of size $|E|/|V|$ to be included in the final cut. Then, they assign the vertices to partitions considering a vertex pair at a time such that the cut value in between the partitions is maximized. Kajitani et al. [22] modified the Haglin and Venkatesan's approach by using a matching with $|E|/(|V| - 1)$ edges in G, which they computed in $O(|E| + |V|)$ time. This allowed them to obtain an approximation ratio of $1/2 + \frac{1}{2(|V|-1)}$ which is a slight (but not asymptotic) improvement.

The most remarkable approximation results for MAX-CUT problem are associated with using semidefinite programming relaxations of MAX-CUT formulations. In their breakthrough paper [17], Goemans and Williamson used semidefinite programming to develop an algorithm for MAX-CUT that always delivers solutions of expected value at least 0.87856 times the optimal value. Feige et al. [24] improved the last step of the Goemans-Williamson algorithm to obtain an approximation ratio of at least 0.921 for graphs of maximum degree three. Liu et al. [25] proposed a tighter semidefinite relaxation of MAX-CUT. For cubic graphs, *i.e.*, graphs in which the degree of all vertices is three, Halperin et al. [21] presented an improved semidefinite programming based approximation algorithm, which has an approximation ratio of 0.9326. Semidefinite programming approaches yield algorithms with the best known approximation ratios, however, the time and space requirements limit their applicability in practice.

In 2002, Alperin and Nowak presented a smoothing heuristic based on Lagrangian relaxation [3]. The heuristic is based on a parametric optimization problem defined as a convex combination between a Lagrangian relaxation and the original problem. Starting from the Lagrangian relaxation, a path following method is

applied to obtain good solutions while gradually transforming the relaxed problem into the original problem formulated with an exact penalty function.

Although researchers found improvements, especially by making use of semidefinite programming, since the publication of the very basic 0.5-approximation algorithm of Sahni and Gonzalez, there has not been much progress in developing algorithms with a constant approximation ratio that would be fast, simple and effective in practice. This is especially important for the cases where one needs to solve MAX-CUT as a subroutine many times. It is also important to note that as long as total weight $W(G)$ of all edges in the graph is used instead of the optimal cut value in derivation of the approximation ratio result, we cannot prove that the approximation ratio is better than $1/2$ for any algorithm. This is because there exist graphs on which the MAX-CUT value is very close to $W(G)/2$. At the same time, finding a tighter upper bound for the MAX-CUT value is not trivial. In fact, this bound is sharp in some sense, since a bipartite graph has a MAX-CUT value which is exactly $W(G)$. The need for improved simple algorithms which would outperform the algorithm proposed by Sahni and Gonzalez in terms of solution quality in practical applications is another motivation to consider several alternative greedy approaches which we present and analyze in the following two sections.

3 The edge contraction heuristic

Let $e = (x, y)$ be an edge of a graph $G = (V, E)$. Contracting an edge e means forming a new vertex $v = v_{xy}$ out of e, which becomes adjacent to all the former neighbors of x and y. The edge contraction heuristic takes the graph $G = (V, E)$ as an input. If the graph is not complete, add all missing edges with weight zero to the original graph. The minimum weighted edge of G is contracted and the graph is updated. Each time an edge e is contracted, the number of the vertices in the graph decreases by one. This procedure is done repeatedly until the number of vertices remaining in the graph becomes two. This heuristic is a worst-out greedy method. The motivation for this method comes from the fact that at each iteration, contracting the minimum weighted edge corresponds to removing this edge from the final solution by assigning the adjacent vertices to the same partition.

In the algorithm, whenever an edge e, whose endpoints are the vertices x and y, is contracted, we form the edges adjacent to the new vertex $v = v_{xy}$. Let i be a vertex distinct from x and y. Then the weight of the new edge between the vertices v_{xy} and i is obtained by adding the weights of the edges (x, i) and (y, i): $w_{vi} = w_{xi} + w_{yi}$. At each step, each vertex v has a *contraction list* which contains all the vertices adjacent to edges that were contracted in previous steps of the algorithm to form vertex v. When the algorithm stops, we have two vertices whose *contraction lists* give us the output cut partition. The steps of the algorithm are summarized in Algorithm 1. It is easy to see that the time complexity of this algorithm is $O(|V|^3)$.

Note that if we use $|V| - k$ steps instead of $|V| - 2$ steps in the main **for**-loop of this algorithm, then we obtain a heuristic for the maximum k-cut problem, which is to partition all vertices into k disjoint sets so that the some of the weights of all edges with endpoints in different parts is maximized. The $ContractionList(i)$ set of each remaining vertex i would correspond to the partitions, while the objective

---

Algorithm 1: The Edge Contraction Heuristic

> **Input**: A complete graph $G(V, E)$ with edge weights $w_{ij}, \forall i, j \in V, \; i \neq j$
>
> **Output**: A cut $S_1, S_2 : S_1 \cup S_2 = V, S_1 \cap S_2 = \emptyset$ and the cut value $cut(S_1, S_2)$
>
> **for** $j = 1 : |V|$ **do**
>
> $ContractionList(j) = \{j\}$
>
> **end**
>
> **for** $j = 1 : |V| - 2$ **do**
>
>   Find a minimum weight edge (x, y) in G
>
> $v = contract(x, y)$
>
> $V = V \cup \{v\} \setminus \{x, y\}$
>
>   **for** $i \in V \setminus \{v\}$ **do**
>
> $w_{vi} = w_{xi} + w_{yi}$
>
>   **end**
>
> $ContractionList(v) = ContractionList(x) \cup ContractionList(y)$
>
> **end**
>
> Denote by x and y the only 2 vertices in V
>
> $S_1 = ContractionList(x);$
>
> $S_2 = ContractionList(y);$
>
> $cut(S_1, S_2) = w_{xy}$

---

function value would be the sum of weights of all remaining edges.

The following lemma will be used to prove an approximation ratio result for the contraction algorithm for the MAX-CUT problem.

Lemma 1. *Let W_k denote the total weight of the first k edges contracted by the edge contraction heuristic and let W be the total weight of all edges in the graph. Then for any $1 \leq k \leq n - 2$*

$$W_k \leq \frac{2kW}{(n-1)(n-k+1)}$$

where n is the number of vertices in the input graph.

Proof. The proof is based on the fact that the weight of a contracted edge is no greater than the average edge weight in the current graph at each iteration. This is true since the procedure always contracts an edge of the smallest weight. So, the weight of the first contracted edge satisfies

$$W_1 \leq \frac{W}{\binom{n}{2}} = \frac{2W}{n(n-1)}.$$

We will use induction on k. We have already shown that the lemma is valid for $k = 1$. Assume it is correct for all integer $k \leq \kappa$. We need to derive the inequality in lemma for $k = \kappa + 1$. Expressing the upper bound on $W_{\kappa+1}$ through W_κ and W, and using the induction assumption for W_κ, we obtain:

$$W_{\kappa+1} \leq W_{\kappa} + \frac{W - W_{\kappa}}{\binom{n-\kappa}{2}} = \frac{(n-\kappa)(n-\kappa-1) - 2}{(n-\kappa)(n-\kappa-1)} W_{\kappa} + \frac{2W}{(n-\kappa)(n-\kappa-1)}$$

$$\leq \frac{2W}{(n-\kappa)(n-\kappa-1)} \left(\frac{((n-\kappa)(n-\kappa-1) - 2)\kappa}{(n-1)(n-\kappa+1)} + 1 \right)$$

$$= \frac{2(\kappa+1)W(n-\kappa+1)(n-\kappa-1)}{(n-\kappa)(n-\kappa-1)(n-1)(n-\kappa+1)}$$

$$= \frac{2(\kappa+1)W}{(n-1)(n-\kappa).}$$

Thus,

$$W_{\kappa+1} \leq \frac{2(\kappa+1)W}{(n-1)(n-\kappa)}$$

and by induction the lemma is correct. □

Theorem 1. *Denote by W_c the value of the cut obtained using the edge contraction heuristic and by W^* the weight of an optimal cut. Then*

$$W_c \geq \frac{1}{3} \left(1 + \frac{2}{n-1} \right) W$$

and, in particular,

$$W_c > \frac{1}{3} W^*.$$

Here, as before, W denotes the total weight of all edges in the graph.

Proof. The proof follows from Lemma 1. Indeed, $W_c = W - W_{n-2}$, thus from the lemma

$$W_c \geq W - \frac{2(n-2)W}{(n-1)(n-(n-2)+1)} = \frac{1}{3} \left(1 + \frac{2}{n-1} \right) W,$$

and since $W \geq W^*$, we obtain $W_c > \frac{1}{3} W^*$. □

4 Modifications to the edge contraction heuristic and Sahni-Gonzalez algorithm

In this section, we discuss four algorithms which are obtained by modifying the edge contraction heuristic and the Sahni-Gonzalez algorithm. The algorithms SG1, SG2 and SG3 are variations of Sahni-Gonzalez algorithm where the order to consider the vertices depends on a score function.

The compromise heuristic: This heuristic is a combination of the edge contraction heuristic and the Sahni-Gonzalez algorithm. We first apply the edge contraction heuristic until the weight of the minimum weighted edge on updated graph becomes greater than or equal to the average edge weight of the input graph. The intuition behind this idea is that if the edge weights in a

graph are much smaller than the average edge weight, then it is more likely that the endpoints of these type of edges will be in the same partition of the MAX-CUT. We will call them "light edges" and the edges that are not light will be called "heavy". After all light edges are contracted, we apply the Sahni-Gonzalez algorithm on the updated graph.

SG1: This is a best-in algorithm, which is a modification of the Sahni-Gonzalez approach. Its steps are summarized in Algorithm 2. Here $w(i, S_j)$ denotes the total weight of the edges in between vertex i and the vertices in the partition V_j, $j = 1, 2$. At each iteration, the SG1 algorithm considers all the remaining vertices and picks the one which will contribute the most to the current cut value at that iteration.

Algorithm 2: The SG1 Algorithm

Input: A complete graph $G(V, E)$ with edge weights $w_{ij}, \forall i, j \in V, i \neq j$

Output: A cut $S_1, S_2 : S_1 \cup S_2 = V, S_1 \cap S_2 = \emptyset$ and the cut value $cut(S_1, S_2)$

$V' = V$

Pick the maximum weighted edge (x, y)

$cut(S_1, S_2) = w_{xy}$

$V' = V' \setminus \{x, y\}$

$S_1 = \{x\}; \quad S_2 = \{y\}$

for $j = 1 : n - 2$ **do**

    for $i \in V'$ **do**

        $score(i) = \max\{w(i, S_1), w(i, S_2)\}$

    end

    Choose the vertex i^* with the maximum score

    If $w(i^*, S_1) > w(i^*, S_2)$ then add i^* to S_2 else add it to S_1

    $V' = V' \setminus \{i\}$

    $cut(S_1, S_2) = cut(S_1, S_2) + \max\{w(i^*, S_1), w(i^*, S_2)\}$

end

SG2: The SG2 algorithm is very similar to SG1. The differences are in the definition of the *score* function and in the choice of the next vertex to be included in one of the partitions:

- $score(i) = \min\{w(i, S_1), w(i, S_2)\}$
- Choose the vertex i^* with the minimum score.

This algorithm can be thought of as a best-in algorithm for the minimum 2 set partitioning problem, which is to partition all vertices into two disjoint subsets so that the sum of weights of edges with both endpoints in the same partition is minimized. Obviously, this problem is equivalent to the MAX-CUT problem. Indeed, at each step a vertex that is the best in terms of contributing to the goal of minimizing the current partitions weight is chosen.

SG3: This algorithm is also very similar to SG1, the only difference being the score function. In this case the score of each remaining vertex is calculated as follows:

$$score(i) = |w(i, S_1) - w(i, S_2)|$$

In fact, SG3 can be viewed as a clever combination of SG1 and SG2. For all the vertices, it takes into account the contribution to the minimization of the current partition weight and at the same time the contribution to the current cut value in that iteration by simply looking at the absolute difference.

In the next section, we present the results of the numerical experiments.

5   Numerical results

This section presents the results of the numerical experiments to compare the performances of the algorithms introduced in Sections 3 and 4, and also the basic Sahni-Gonzalez [27] algorithm. We tested these algorithms on several randomly generated graphs and some TSP instances from TSPLIB. These graphs are all weighted and complete. In addition, some instances were taken from Resende et al. [15] (G1, G2, G15, G17 and G53 are all 0, 1 weighted). The size of the TSP instances are already specified in their names and for an instance named rxx, yy, xx denotes the number of the vertices and yy denotes the maximum weight of edges. G1,G2, G15 and G17 are of size 800 while G53 is of size 1000. We first compare the results for Sahni-Gonzalez algorithm and the edge contraction heuristic in Table 1. In this table, W_{SG} and W_C represent the value of the cut obtained using the Sahni-Gonzalez and the contraction algorithm, respectively, while W stands for the sum of weights of all edges in the graph.

From Table 1, the Sahni-Gonzalez algorithm outperforms the edge contraction heuristic by giving better approximation in general. In particular, on three instances ($r25, 30$; $r45, 50$; and $r150, 3$) the ratio W_C/W is less than $1/2$, which is the approximation ratio of the Sahni-Gonzalez algorithm. Thus, the approximation ratio of the edge contraction heuristic is less than that of the Sahni-Gonzalez algorithm. The $1/3$ bound derived in Theorem 1 may be tight, but has not been proven so. However, on 5 out of 9 TSP instances, regarded as "dense" instances, the edge contraction heuristic does better than the Sahni-Gonzalez algorithm. This observation enhances our motivation for the compromise heuristic. The Sahni-Gonzalez algorithm is also better than the edge contraction heuristic in terms of running time. The time complexity of the edge contraction heuristic is $O(|V|^3)$. Since we are considering the complete graphs, the time complexity of Sahni-Gonzalez algorithm is $O(|V|^2)$.

Next, we present the results obtained by the edge contraction heuristic (C), the Sahni-Gonzalez algorithm (SG), the compromise heuristic (CSG) and the modified versions, $SG1$, $SG2$ and $SG3$ of SG in Table 2. Here the results are presented as the ratios of the cut value of the solution obtained using a given algorithm to the total edge weight W. We see that the compromise heuristic (CSG) does not perform better than the contraction and Sahni-Gonzalez algorithms. But the results we obtained from the algorithms $SG1$, $SG2$ and $SG3$ are more encouraging. Among these $SG3$ is the best overall. As it was explained in the previous section, $SG3$ chooses the "best" candidate at each iteration by considering the absolute contribution in terms of both the increase in the cut value and the increase in the partition weight. Figure 1 illustrates the results graphically for the SG, C and $SG3$ algorithms. It clearly shows that $SG3$ outperforms the contraction and Sahni-Gonzalez algorithms in all considered instances.

Table 1   Comparative results of the algorithm of Sahni and Gonzalez [27] and the edge contraction heuristic

| Instance		Sahni-Gonzalez		Edge Contraction	
Name	W	W_{SG}	W_{SG}/W	W_C	W_C/W
$Burma14$	355	193	0.543662	254	0.715493
$gr17$	37346	23354	0.625341	24986	0.669041
$bayg29$	66313	36939	0.55704	35058	0.528675
$bays29$	370530	214339	0.578466	226102	0.610212
$dantzig42$	59574	30508	0.512103	36023	0.604677
$att48$	3.74E+06	2.49E+06	0.665833	2.25E+06	0.602836
$hk48$	1.15E+06	712355	0.617408	732706	0.635046
$berlin52$	762783	453174	0.594106	445739	0.584359
$brazil58$	3.52E+06	1.92E+06	0.543782	1.83E+06	0.519003
$r15, 30$	2679	1504	0.561404	1504	0.561404
$r20, 30$	8723	4675	0.535939	4522	0.5184
$r25, 30$	6115	3384	0.553393	3038	0.496811
$r25, 40$	12161	6603	0.542965	6186	0.508675
$r30, 40$	15202	8153	0.536311	8073	0.531049
$r35, 40$	18926	10293	0.543855	10122	0.53482
$r45, 50$	44705	23708	0.530321	21840	0.488536
$r55, 40$	29487	16306	0.552989	15827	0.536745
$r55, 50$	36889	20375	0.552333	20041	0.543278
$r63, 75$	72801	39604	0.544004	39163	0.537946
$r75, 130$	177855	96491	0.542526	92445	0.519777
$r80, 10$	15737	8492	0.53962	7873	0.50028595
$r82, 130$	214256	115466	0.538916	108347	0.505689
$r82, 20$	33461	17973	0.537133	16868	0.504109
$r100, 50$	122178	65834	0.538837	61851	0.506237
$r150, 3$	16863	9041	0.536144	8426	0.499674
$r150, 4$	22064	11785	0.534128	11066	0.501541
$r250, 100$	1.57E+06	813292	0.519272	796739	0.508703
$r500, 101$	6.35E+06	3.27E+06	0.514598	3.23E+06	0.508212
$G1$	19176	10949	0.570974	9794	0.510743
$G2$	19176	11050	0.576241	9955	0.519139
$G15$	4661	2865	0.614675	2645	0.567475
$G17$	4667	2883	0.617742	2656	0.569102
$G53$	5914	3642	0.615827	3352	0.566791

Our experimental analysis shows that $SG3$ is the best choice although it has the same worst-case approximation ratio of $1/2$ as the original Sahni-Gonzalez algorithm. It is important to note that experimental analysis has a crucial role especially in comparison of construction algorithms for MAX-CUT since theoretically it is not possible to obtain an approximation ratio greater than $1/2$ as long as the total edge weight is used instead of the optimal cut value in the approximation ratio derivation.

Table 2 Comparative results of the ratios of the cut value to the graph's total edge weight achieved by SG, C, CSG, $SG1$, $SG2$ and $SG3$

Name	SG	C	CSG	$SG1$	$SG2$	$SG3$
$burma14$	0.543662	0.715493	0.695775	0.721127	0.715493	0.721127
$gr17$	0.625341	0.669041	0.555669	0.669041	0.669041	0.669041
$bayg29$	0.55704	0.528675	0.539306	0.559242	0.569632	0.564173
$bays29$	0.578466	0.610212	0.57330	0.642666	0.617070	0.642666
$dantzig42$	0.512103	0.604677	0.509659	0.677024	0.677024	0.677024
$att48$	0.665833	0.602836	0.672023	0.674483	0.674661	0.674661
$hk48$	0.617408	0.635046	0.614240	0.668855	0.668855	0.668855
$berlin52$	0.594106	0.584359	0.572952	0.616434	0.614527	0.617117
$brazil58$	0.543782	0.519003	0.543635	0.546723	0.556287	0.563592
$r15,_3 0$	0.561404	0.561404	0.567376	0.565136	0.578201	0.583053
$r20, 30$	0.535939	0.518400	0.529634	0.544079	0.548321	0.550384
$r25, 30$	0.553393	0.496811	0.575470	0.573508	0.569583	0.573181
$r25, 40$	0.542965	0.508675	0.532522	0.555464	0.553326	0.554889
$r30, 40$	0.536311	0.531049	0.543810	0.552296	0.548349	0.552033
$r35, 40$	0.543855	0.534820	0.528796	0.551358	0.550988	0.557117
$r45, 50$	0.530321	0.488536	0.523722	0.542624	0.541461	0.541640
$r55, 40$	0.552989	0.536745	0.541832	0.556856	0.548377	0.563401
$r55, 50$	0.552333	0.543278	0.539863	0.558107	0.548619	0.559869
$r63, 75$	0.544004	0.537946	0.538770	0.552932	0.548797	0.555308
$r75, 130$	0.542526	0.519777	0.539872	0.548160	0.541705	0.555149
$r80, 10$	0.539620	0.500286	0.534981	0.549215	0.543750	0.554299
$r82, 130$	0.538916	0.505689	0.529096	0.539850	0.543285	0.549833
$r82, 20$	0.537133	0.504109	0.533815	0.542004	0.545710	0.548549
$r100, 50$	0.538837	0.506237	0.526781	0.539606	0.539639	0.542299
$r150, 3$	0.536144	0.499674	0.530688	0.538813	0.541481	0.544565
$r150, 4$	0.534128	0.501541	0.522117	0.538751	0.537255	0.540790
$G1$	0.570974	0.510743	0.531237	0.584220	0.580622	0.591834
$G2$	0.576241	0.519139	0.522685	0.586045	0.578796	0.594180
$G15$	0.614675	0.567475	0.541729	0.628835	0.597726	0.642137
$G17$	0.617742	0.569102	0.537819	0.628455	0.599529	0.638097
$G53$	0.615827	0.566791	0.542273	0.629523	0.599087	0.638147

6   Conclusion

In this paper, a greedy worst-out construction heuristic for the MAX-CUT problem called the edge contraction heuristic was introduced. We have shown that it has an approximation ratio of at least $1/3$ and a time complexity of $O(|V|^3)$. To the best of our knowledge, this is the first time a worst-out greedy approach has been applied to approximate the MAX-CUT problem. We also proposed several best-in $\frac{1}{2}$-approximation algorithms for the same problem that are modifications of a well-known heuristic introduced by Sahni and Gonzalez [27]. We carried out some numerical experiments to compare the performance of these heuristics. Our experiments showed that the edge contraction heuristic is outperformed by Sahni and
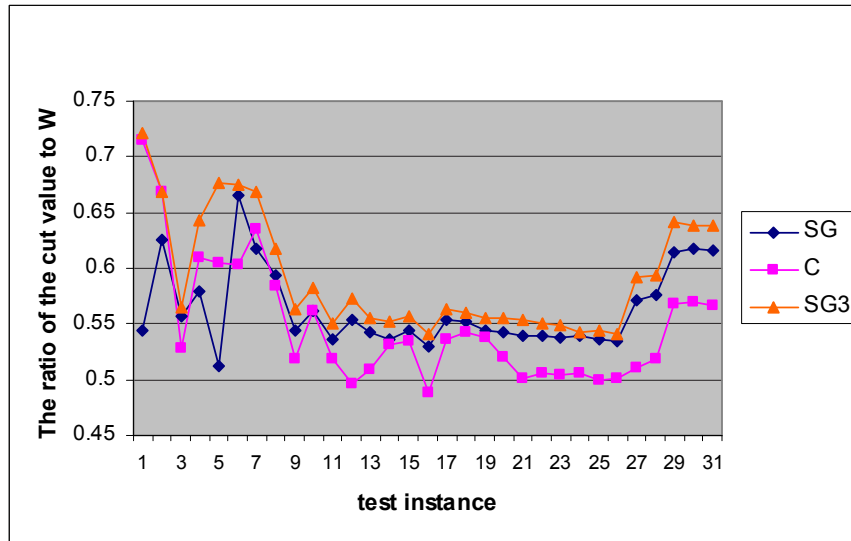
Figure 1   Comparison of results for *SG*, *C*, and *SG*3 algorithms. The instances are numbered in the order they are presented in Tables 1 and 2.

Gonzalez approach. Moreover, we observed that its approximation ratio is worse than that of the Sahni-Gonzalez algorithm. We also observed that the modified versions of Sahni-Gonzalez heuristics, where a score function is used to determine the best candidate at each iteration, outperform the Sahni-Gonzalez heuristic. Based on our experiments, we concluded that this approach is outperformed by best-in algorithms, while the results obtained from modified versions of the Sahni-Gonzalez approach are quite encouraging. Recall that these results are for construction algorithms only and can be further improved by applying improvement heuristics, such as local search and advanced metaheuristic strategies [1, 16].

Acknowledgements

References and Notes

1 E. H. L. Aarts and J. K. Lenstra, editors. *Local Search in Combinatorial Optimization*. Princeton University Press, 2003.

2 R. Agrawal, S. Rajagopalan, R. Srikant and Y. Xu. Mining Newsgroups Using Networks Arising From Social Behavior. *Proceedings of the Twelfth International World Wide Web Conference*, Hungary, May 2003.

3 H. Alperin and I. Nowak. Lagrangian Smoothing Heuristics for Max-Cut. *Journal of Heuristics*, 11: 447–463, 2005.

4 B. Balasundaram and S. Butenko. Constructing Test Functions for Global Opti-

mization Using Continuous Formulations of Optimization Problems on Graphs. *Optimization Methods and Software*, 20:1-14, 2005.

5 F. Barahona. The max-cut problem in graphs not contractible to K_5. *Operations Research Letters*, 2:107–111, 1983.

6 F. Barahona, M. Grotschel, M. Junger, and G. Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36: 493–513, 1988.

7 Y. Bramoullé. Anti-Coordination and Social Interactions. *Games and Economic Behavior*, forthcoming

8 S. Burer and R.D.C. Monteiro. A projected gradient algorithm for solving the Max-Cut SDP relaxation. *Optimization Methods and Software*, 15: 175-200, 2001.

9 S. Burer, R.D.C. Monteiro, and Y. Zhang. Rank-two relaxation heuristics for MAX-CUT and other binary quadratic programs. *SIAM J. on Optimization*, 12: 503-521, 2001.

10 Y. Chen, M.M. Crawford, and J. Ghosh. Integrating Support Vector Machines in a Hierarchical Output Decomposition Framework. *Proceedings of the International Geoscience and Remote Sensing Symposium*, Anchorage, Alaska, pages 949-953, September 20-24, 2004.

11 J. D. Cho, S. Raje, and M. Sarrafzadeh. Fast approximation algorithms on max-cut, k-coloring, and k-color ordering for VLSI applications. *IEEE Transactions on Computers*, 47(11), 1998.

12 M. Deza and M. Laurent. *Geometry of Cuts and Metrics*. Springer Verlag, 1997.

13 W. Fernandez de la Vega. Max-cut has a randomized approximation scheme in dense graphs. *Random Structures and Algorithms*, 8(3):187–198, 1996.

14 W. Fernandez de la Vega and C. Kenyon. A randomized approximation scheme for metric max-cut. In *Proceedings of the 39th Anuall Symposium on Foundations of Computer Science*, page 468, 1998.

15 P. Festa, P. M. Pardalos, M. G. C. Resende, and C. C. Ribeiro. Randomized heuristics for the max-cut problem. *Optimization Methods and Software*, 7:1033–1058, 2002.

16 F. Glover and G. Kochenberger. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003.

17 M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *ACM-SIAM Symp. On Theory of Computing*, pages 431–442, 1994.

18 M. Grotschel and W.R. Pulleyblank. Weakly bipartite graphs and the max-cut problem. *Operations Research Letters*, 1:23–27, 1981.

19 F. O. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4:221–225, 1975.

20 D. J. Haglin and S. M. Venkatesan. Approximation and intractability results for the maximum cut problem and its variants. *IEEE Transactions on Computers*, 40:110–113, 1991.

21 E. Halperin, D. Livnat, and U. Zwick. MAX-CUT in cubic graphs. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 506–513, 2002.

22 Y. Kajitani, J. D. Cho, and M. Sarrafzadeh. New approximation results on graph matching and related problems. In *Graph-Theoretic Concepts in Computer Science, International Workshop (WG), LNCS (903)*, pages 343–358. Springer-Verlag, Berin, 1994.

23 R. M. Karp.   Reducibility among combinatorial problems.   In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

24 M. Karpinski, U. Feige, and M.Landberg. Improved approximation of max-cut on graphs of bounded degree. *Journal of Algorithms*, 43:201–219, 2002.

25 H. Liu, S. Liu, and F. Xu. A tight semidefnite relaxation of the MAX-CUT problem. *Journal of Combinatorial Optimization*, 7:237–245, 2003.

26 C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Comput. System Science*, 43(3):425–440, 1991.

27 S. Sahni and T. Gonzales. P-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.

28 L. Trevisan, G. B. Sorkin, M. Sudan, and D. P. Williamson. Gadgets, approximation, and linear programming. *SIAM Journal on Computing*, 29(6):2074– 2097, 2000.