An exact algorithm for MAX-CUT in sparse graphs

Federico Della Croce, Marcin J. Kaminski, Vangelis Th. Paschos

An exact algorithm for MAX-CUT in sparse graphs[1]

Federico Della Croce[*], Marcin. J. Kaminski[†], Vangelis. Th. Paschos[‡]

Abstract

The MAX-CUT problem consists in partitioning the vertex set of a weighted graph into two subsets. The objective is to maximize the sum of weights of those edges that have their endpoints in two different parts of the partition. MAX-CUT is a well known **NP**-hard problem and it remains **NP**-hard even if restricted to the class of graphs with bounded maximum degree Δ (for $\Delta \geq 3$). In this paper we study exact algorithms for the MAX-CUT problem. Introducing a new technique, we present an algorithmic scheme that computes maximum cut in weighted graphs with bounded maximum degree. Our algorithm runs in time $O^*(2^{(1-(2/\Delta))n})$. We also describe a MAX-CUT algorithm for general weighted graphs. Its time complexity is $O^*(2^{mn/(m+n)})$. Both algorithms use polynomial space.

1 Background

There has recently been a growing interest in analysis of the worst-case complexity of many **NP**-hard problems. Unless **P** $\neq$ **NP**, solving such problems requires super-polynomial time. Each problem in **NP** can be solved by a naive algorithm that exhaustively searches the solution space. However, for most of the problems more refined algorithms with better, but still exponential-time complexity, are known.

[1]The work was done while the first author was visiting LAMSADE on a research position funded by CNRS and the second author was visiting LAMSADE being supported by DIMACS under the NSF grant INT03-39067 to Rutgers University.

[*]D.A.I., Politecnico di Torino, Italy, federico.dellacroce@polito.it

[†]RUTCOR, Rutgers University, USA, mkaminski@rutcor.rutgers.edu

[‡]LAMSADE, CNRS UMR 7024 and Université Paris-Dauphine, France paschos@lamsade.dauphine.fr

Development of exact algorithms is mainly of theoretical interest but existence of fast exponential algorithms may also have practical importance. Today's computers are able to handle moderate size instances of **NP**-hard problems. However, even though one can afford to run an exponential-time algorithm, polynomial-space complexity is a must.

Satisfiability, graph coloring and maximum independent set problem are among the problems that received much attention in the context of exact algorithms. In this paper we study another well-known **NP**-hard problem. Given an arbitrary graph with real weights assigned to its edges, the MAX-CUT problem asks to find a partition of vertices into two subsets such that the sum of weights of all the edges that have endpoints in two different parts of the partition is maximized. In unweighted case (i.e. all weights are positive and equal) the problem is often referred to as SIMPLE MAX-CUT.

1.1 Previous work

SIMPLE MAX-CUT was one of the first problems whose **NP**-hardness was established. However, there are classes of graphs as planar graphs, graphs with large girth ([5]), cographs and graphs with bounded treewidth ([1]), that admit polynomial-time solution of this problem.

On the other hand, SIMPLE MAX-CUT (and therefore MAX-CUT) remains **NP**-hard even if restricted to such classes as chordal, split, or 3-colorable graphs ([1]). As shown in [11], the problem is **NP**-hard also in the class of graphs with bounded maximum degree Δ, if $\Delta \geq 3$ (for $\Delta \leq 2$ the problem becomes trivial).

The worst-case complexity of the MAX-CUT problem has been studied in few papers. The fastest algorithm for MAX-CUT in arbitrary graphs was proposed by Williams in [9]. Unfortunately it requires exponential space. The algorithm runs in time $O^*(2^{\omega n/3})$ and uses $O^*(2^{\omega n/3})$ space, where in notation $O^*(\cdot)$ polynomial multiplicative terms are omitted and $\omega < 2.376$ is the matrix multiplication exponent (the product of two $k \times k$ matrices can be computed in time $O(k^\omega)$). Whether exists a polynomial-space algorithm that computes SIMPLE MAX-CUT and runs faster than the naive one of time complexity $O^*(2^n)$ is an open question listed in [10].

More algorithms has been developed for sparse graphs. In [3] the bound of $O^*(2^{m/3})$ was obtained and then improved to $O^*(2^{m/4})$ in [2]. In [7] (see also [8]) an algorithm running in time $O^*(2^{\min((m-n)/2, m/5)})$ and polynomial space was proposed.

1.2 Our contribution

In this paper we develop a technique that seems to be a new approach to the MAX-CUT problem. The method consists in enumerating cuts in a subgraph H of G and then

extending them in an optimal way to cuts in G. The technique is applied to graphs with bounded maximum degree and to general graphs. In both cases, we obtain an exponential-time algorithm that uses polynomial space.

For weighted graphs with bounded maximum degree Δ, we present an algorithmic scheme that computes a maximum cut. For fixed Δ, the algorithm runs in time $O^*(2^{(1-(2/\Delta))n})$ and polynomial space. The algorithm is faster than [3, 2], however for $\Delta \leq 7$, the running time of our algorithm is dominated by the running time of [7]. It is also slower than the exponential-space [9] for $\Delta < 10$.

For general weighted graphs, we obtain an algorithm that computes a maximum cut and runs in time $2^{mn/(m+n)}$. The running time of our algorithm dominates the running times of [3] and [2] for $m > 2n$ and $m > 3n$, respectively. If $m < 7n/5$ the algorithm is faster than [7] and faster than the exponential-space [9] for $m < \omega n/(3-\omega) < 3.808n$.

The organization of the paper is as follows. The next section is a formal introduction and contains definitions used later. In Section 3 we study a modification of the MAX-CUT problem and develop our technique which is applied in Section 4 to graphs with bounded maximum degree and to general graphs in Section 5.

2 Introduction

We consider weighted, undirected, loopless graphs without multiple edges. In a graph $G = (V, E, w)$, V is the vertex set of cardinality $|V| = n$, E is the edge set of cardinality $|E| = m$, and $w : E \to \mathbb{R}$ is a weight function that assigns a real number w_{ij} to each edge ij of G.

The number of edges incident to a vertex in a graph is called the *degree* of the vertex. The maximum degree of all the vertices of a graph is called the *maximum degree* of the graph and denoted by Δ. The *average degree* of a graph is the sum of degrees of all vertices of the graph divided by the number of vertices. The average degree is denoted by d. Notice that $d = 2m/n$. Given a subset U of vertices of V, the subgraph induced by the vertices in U is denoted by $G[U]$.

A *cut* $C = (V_0, V_1)$ in a graph is a partition of its vertex set V into two disjoint subsets V_0 and V_1. The weight $w(C)$ of cut C is the sum of weights of all the edges that have their endpoints in two different parts of the cut. Notice that the characteristic vector of one of the parts, say V_0, uniquely determines the partition.

For the purpose of this paper, we will think of a partition as an assignment of $0-1$ values to the vertices of the graph. Let x_i be a Boolean variable which takes value 0, if $v_i \in V_0$, and 1, if $v_i \in V_1$. The weight of a cut in a graph $G = (V, E, w)$ can be

expressed as a pseudo-boolean function,

$$w(C) = \sum_{ij \in E} w_{ij} \left(x_i \overline{x_j} + \overline{x_i} x_j \right) = \sum_{i \in V} w_i x_i - 2 \sum_{ij \in E} w_{ij} x_i x_j, \tag{1}$$

where $w_i = \sum_{\{i,j\} \in E} w_{ij}$. A *maximum cut* in a graph G is a cut of maximum weight.

Given a graph G as an input, the MAX-CUT problem asks to compute a cut in G that maximizes (1).

Notice that it is enough to consider only connected graphs as if the graph is not connected the MAX-CUT problem can be solved for each of its connected components separately.

It is easy to see that if the weights are restricted to be nonnegative real numbers, the MAX-CUT problem can be solved in polynomial time for the class of bipartite graphs.

3 Extending partial partition of vertices

In this section we consider a modification of the MAX-CUT problem. Suppose some of the vertices have already been partitioned into two subsets and now the problem is to find an optimal cut in the graph with respect to that pre-partition. We prove that if the graph induced by the vertices that have not yet been partitioned is bipartite, then the problem of finding an optimal extension of the partial partition can be solved in polynomial time. The algorithms presented in the next sections are based on this result.

Let $U \subset V$ be a subset of vertices of G such that the subgraph $G' = G[U']$ induced by the vertices in $U' = V \setminus U$ is bipartite. Also, let (U_0, U_1) be a partition of U into two subsets. Consider the problem of finding a partition (V_0, V_1) of V with $U_0 \subset V_0$ and $U_1 \subset V_1$ that maximizes (1).

The vertices in U have already been assigned to parts of the cut, thus variables x_i, for $i \in U$, have their values fixed. There are four possible types of edges in the cut: with both endpoints in U, from U_0 to U', from U_1 to U', and with both endpoints in U'. The problem of finding an optimal extension of pre-partition is now equivalent to maximizing the following pseudo-boolean function,

$$\sum_{\substack{i \in U_0 \\ j \in U_1}} w_{ij} + \sum_{\substack{i \in U_0 \\ j \in U'}} w_{ij} x_j + \sum_{\substack{i \in U_1 \\ j \in U'}} w_{ij} \overline{x_j} + \sum_{\substack{i \in U' \\ j \in U'}} w_{ij} \left(x_i \overline{x_j} + \overline{x_i} x_j \right) \tag{2}$$

where all sums are taken over edges $ij \in E$ of the graph G. Putting,

$$c_{ij} = \sum_{i \in U_0} w_{ij} - \sum_{i \in U_1} w_{ij} + \sum_{i \in U'} w_{ij}$$

where all sums are again taken over edges $ij \in E$, and omitting the constant term, the problem is equivalent to finding a maximum of the function,

$$\sum_{j \in U'} c_{ij} x_j - 2 \sum_{ij \in E'} w_{ij} x_i x_j \tag{3}$$

where E' is the edge set of the bipartite graph G'. In other words, the problem of finding an optimal extension of pre-partition can be stated as the following integer quadratic program:

$$\begin{array}{ll} \max & \displaystyle\sum_{i \in U'} c_i x_i - 2 \sum_{ij \in E'} w_{ij} x_i x_j \\ \text{s.t.} & x_i \in \{0, 1\} \end{array} \tag{4}$$

The standard linearization technique applied to (4) by introducing $y_{ij} = x_i x_j$, yields the following linear program:

$$\begin{array}{lrcl} \max & \displaystyle\sum_{i \in U'} c_i x_i - 2 \sum_{ij \in E'} w_{ij} y_{ij} & & \\ \text{s.t.} & y_{ij} & \geq & x_i + x_j - 1 \\ & y_{ij} & \geq & 0 \\ & x_i & \in & \{0, 1\} \\ & y_{ij} & \in & \{0, 1\} \end{array} \tag{5}$$

It is easy to see that (4) and (5) are equivalent. They have the same optimal value and there is an easy correspondence between their solutions, namely $y_{ij} = x_i x_j$.

Having modelled the original quadratic problem (4) as an integer linear program, let us study the continuous relaxation of (5):

$$\begin{array}{lrcl} \max & \displaystyle\sum_{i \in U'} c_i x_i - 2 \sum_{ij \in E'} w_{ij} y_{ij} & & \\ \text{s.t.} & y_{ij} & \geq & x_i + x_j - 1 \\ & x_i & \geq & 0 \\ & x_j & \leq & 1 \\ & y_{ij} & \geq & 0 \\ & y_{ij} & \leq & 1 \end{array} \tag{6}$$

Lemma 1. *The constraint matrix of the linear program (6) is totally unimodular, i.e., determinant of every square submatrix of it equals* 0 *or* ± 1.

Proof. Let A be the constraint matrix of (6). It has $|U'|+|E'|$ columns and $2|U'|+3|E'|$ rows and all its entries are either 0 or ± 1. Let B be a edge-vertex incidence matrix of G'. Notice that B is a submatrix of A. Moreover, any submatrix of A that has two non-zero entries in every row and every column has to be a submatrix of B.

Take any square $k \times k$ submatrix of A. We will prove the lemma by induction on k. Clearly, the result holds for $k = 1$.

Now assume that all $(k-1) \times (k-1)$ submatrices of A are totally unimodular and consider matrix M which is a $k \times k$ submatrix of A.

If all entries of any row or column of M are 0, then $\det(M) = 0$ and M is totally unimodular. If any row or column of M has a single non-zero element (± 1), then using the expansion method for calculating determinant and the induction hypothesis, it is easy to see that $\det(M)$ is either 0 or ± 1, and A is totally unimodular.

Suppose that each row and each column of M has at least two non-zero entries. Hence, M must be a submatrix of B but since B is an incidence matrix of a bipartite graph so is M. It is possible to partition columns of M into two parts, according to the partition of vertices of bipartite graph. Sum of the columns in each part yields a unit vector (each edge of the bipartite subgraph has one endpoint in each part) and that implies linear dependence of M, therefore $\det(M) = 0$ and M is totally unimodular. □

Theorem 2. *Let $U \subset V$ be such that the subgraph $G' = G[U']$ induced by the vertices in $U' = V \setminus U$ is bipartite and (U_0, U_1) be a partition of U into two subsets, then the problem of finding a partition (V_0, V_1) of V with $U_0 \subset V_0$ and $U_1 \subset V_1$ that maximizes (1) is polynomial-time solvable.*

Proof. The problem of finding a partition (V_0, V_1) of V with $U_0 \subset V_0$ and $U_1 \subset V_1$ that maximizes (1) can be modelled as the integer quadratic program (4) which is equivalent to (5). Total unimodularity of the constraint matrix of (6) (by Lemma 1) implies the existence of an optimal $0-1$ solution of (6). Such solution can be found in polynomial time. Since the relaxation (6) of (5) has an optimal $0-1$ solution, therefore (4) can be solved in polynomial time. □

In the previous section we mentioned that the MAX-CUT problem is polynomial-time solvable in the class of bipartite graphs if the weights are restricted to be nonnegative real numbers. Note that from Theorem 2 follows that the MAX-CUT problem can be solved in polynomial time in the class of bipartite graphs (possibly with negative weights).

Before we proceed to the next section, let us briefly describe the algorithmic technique we are going to apply. Given an induced bipartite subgraph $G[B]$ of B, one can enumerate all partitions of $V \setminus B$ and find an optimal extension of each such partition in polynomial time (by Theorem 2). The complexity of such technique is $O^*(2^{|V \setminus B|})$ and it strongly depends on the size of the bipartite subgraph that has to be constructed.

4 Algorithm for graphs with bounded maximum degree

In this section we present and analyze an algorithmic scheme A(Δ). For a fixed integer Δ ($\Delta \geq 3$), the scheme yields an algorithm whose input is a weighted graph $G = (V, E, w)$ of maximum degree Δ and the output a maximum cut in G with respect to the weight function w.

`Step 1.` If G is isomorphic to the complete graph on $\Delta + 1$ vertices, then let B be any pair of vertices and go to `Step 3`.

`Step 2.` Δ-color G. Let B be the union of 2 largest color classes of the coloring.

`Step 3.` Enumerate all partitions of elements of $V \setminus B$ into two subsets (all $0 - 1$ assignments) and for each find an optimal extension of the partial partition.

`Step 4.` Find a cut C that has the largest weight among all checked in `Step 3`. Return the cut C.

Theorem 3. *For a fixed integer Δ ($\Delta \geq 3$), Algorithm* A(Δ) *computes* MAX-CUT *in a graph G in time $O^*(2^{(1-(2/\Delta))n})$ and polynomial space.*

Proof. Let us notice first that the algorithm indeed finds a maximum cut. It is clear that the induced subgraph $G[B]$ is bipartite. Therefore, any partition of $V \setminus B$ into two subsets can be extended to an optimal partition of V in polynomial time by Theorem 2. Clearly, by enumerating all partitions of $V \setminus B$ and then extending each in optimal way, one finds a maximum cut in G.

The enumeration of partitions in `Step 3` is the bottleneck of the algorithm; it needs exponential time $O^*(2^{|V \setminus B|})$. Other steps can be performed in linear time. It is clear for `Steps 1` and `4`, and the linear time algorithm for `Step 2` is given in [4]. Notice, that the algorithm can be implemented in such a way that it uses only polynomial space.

Suppose that the input graph is isomorphic to the complete graph on $\Delta + 1$ vertices. The number of partitions that are enumerated in `Step 3` is 2^{n-2} but since $\Delta = O(n)$ the claimed running time follows.

Now suppose that the input graph G is not isomorphic to the complete graph on $\Delta + 1$ vertices. Then, by the Brooks' Theorem G is Δ colorable. Clearly, the union of two largest color classes have size at least $2n/\Delta$ and $|V \setminus B| \leq n(1 - (2/\Delta))$. The number of partitions that are enumerated in `Step 3` is $2^{(1-(2/\Delta))n}$ and the claimed running time follows. □

5 Algorithm for general graphs

Let us notice that in the algorithm presented in the previous section, the assumption of bounded maximum degree is needed only to obtain an induced bipartite graph. Now we relax this assumption and study the complexity of the method in general graphs. Let us formalize that as an algorithm B. The input of B is a weighted graph $G = (V, E, w)$ and the output is a maximum cut in G with respect to the weight function w.

`Step 1.` Find a maximal independent set I_0 in G.

`Step 2.` Find a maximal independent set I_1 in $G[V \setminus I_0]$. Let B be the union of I_0 and I_1.

`Step 3.` Enumerate all partitions of elements of $V \setminus B$ into two subsets (all $0-1$ assignments) and for each find an optimal extension of the partial partition.

`Step 4.` Find a cut C that has the largest weight among all checked in `Step 3`. Return the cut C.

To complete the description of the algorithm, we need to provide a procedure that finds an induced bipartite subgraphs in `Steps 1` and `2`.

From Turan's theorem follows that the size of a maximum independent set is at least $n/(d+1)$ and as shown in [6], there is a linear-time algorithm that constructs an independent set of at least that size. As the time complexity of B depends on $|B|$, we need to give a lower bound on the size of bipartite subgraph B.

Claim 4. *The set B of vertices constructed in `Step 2` of Algorithm* B *has size at least* $2/(d+2)$.

Proof. Let $i = |I_0|$ and m' be the number of edges of the subgraph $G[I_0 \cup I_1]$. If $i \geq 2n/(d+2)$, then $|B| \geq 2n/(d+2)$ and the claim follows. Suppose $i < 2n/(d+2)$. The average degree d' in the graph $G[V - I_0]$ is $d' = 2(m-m')/(n-i)$. Notice that $m' \geq n-i$ since I_0 is an independent set. Hence, $d' \leq 2n/(n-i) - 2$ and since $i < 2n/(d+2)$, then $d' < d$. From that, follows that $|B| = i + (n-i)/(d'+1) \geq 2n/(d+2)$. □

Having established the lower-bound on the size of B, we can claim the running time of Algorithm B. Notice that $2/(d+2) = n/(n+m)$ and $n - |B| \leq mn/(m+n)$.

Theorem 5. *Algorithm* B *computes* MAX-CUT *in a graph G with n vertices and m edges in time $O^*(2^{mn/(m+n)})$, and polynomial space.*

The proof of this theorem is similar to the proof of Theorem 3 and will be omitted.

References

[1] H. L. Bodlaender and K. Jansen. On the complexity of the maximum cut problem. *Nordic Journal of Computing*, 7(1):14–31, 2000.

[2] S. S. Fedin and A. S. Kulikov. A $2^{|E|/4}$-time algorithm for max-cut. *Journal of Mathematical Sciences*. To appear. Available at `http://logic.pdmi.ras.ru/~kulikov/maxcut_e.ps.gz`.

[3] J. Gramm, E. A. Hirsch, R. Niedermaier, and P. Rossmanith. Worst-case upper bounds for Max2Sat with an application to MaxCut. *Discrete Appl. Math.*, 130:139–155, 2003.

[4] L. Lovász. Three short proofs in graph theory. *J. Combin. Theory Ser. B*, 19:269–271, 1975.

[5] S. Poljak and Z. Tuza. Maximum cuts and large bipartite subgraphs. In W. Cook, L. Lovász, and P. Seymour, editors, *Combinatorial Optimization. Papers from the DIMACS Special Year*, pages 181–224. AMS, 1995.

[6] S. Sakai, M. Togasaki, and K. Yamazaki. A note on greedy algorithms for the maximum weighted indepedent set problem. *Discrete Appl. Math.*, 126:313–322, 2003.

[7] A. D. Scott and G. B. Sorkin. Faster algorithms for MAX CUT and MAX CSP, with polynomial expected time for sparse instances. In *Proc. RANDOM'03*, volume 2764 of *Lecture Notes in Computer Science*, pages 382–395. Spinger-Verlag, 2003.

[8] A. D. Scott and G. B. Sorkin. Solving sparse semi-random instances of Max-Cut and Max-CSP in linear expected time. Research Report 23417 (W0411-056), IBM Research division, Thomas J. Watson Researcc Center, 2004.

[9] R. Williams. A new algorithm for optimal constraint satisfaction and its implications. Report 32, Electronic Colloquium on Computational Complexity, 2004.

[10] G. J. Wœginger. Space and time complexity of exact algorithms: some open problems. In *Proc. IWPEC'04*, volume 3162 of *Lecture Notes in Computer Science*, pages 281–290. Springer-Verlag, 2004.

[11] M. Yannakakis. Node- and edge-deletion NP-complete problems. In *Proc. STOC'78*, pages 253–264, 1978.