Combinatorial 5/6-approximation of Max Cut in graphs of maximum degree 3

Cristina Bazgan [a] and Zsolt Tuza [b,c,d]

[a] LAMSADE, Université Paris-Dauphine, Place du Marechal de Lattre de Tassigny, F-75775 Paris Cedex 16, France. Email: bazgan@lamsade.dauphine.fr

[b] Computer and Automation Institute, Hungarian Academy of Sciences, H-1111 Budapest, Kende u. 13-17, Hungary. Email: tuza@sztaki.hu

[c] Department of Computer Science, University of Pannonia, H-8200 Veszprém, Egyetem u. 10, Hungary.

[d] Corresponding author. Fax number: +36-1-4667503

Version of 2007–1–5

Abstract

The best approximation algorithm for Max Cut in graphs of maximum degree 3 uses semidefinite programming, has approximation ratio 0.9326, and its running time is $\Theta(n^{3.5}\log n)$; but the best *combinatorial* algorithms have approximation ratio 4/5 only, achieved in $O(n^2)$ time [Bondy and Locke, J. Graph Theory **10** (1986), 477–504; and Halperin et al., J. Algorithms **53** (2004), 169–185]. Here we present an improved combinatorial approximation, which is a 5/6-approximation algorithm that runs in $O(n^2)$ time, perhaps improvable even to $O(n)$. Our main tool is a new type of vertex decomposition for graphs of maximum degree 3.

Keywords: maximum cut, cubic graph, approximation algorithm, vertex decomposition, unicyclic graph.

Mathematics 2000 Subject Classification: 05C35, 05C85, 68W25, 68R10.

1 Introduction

MAX CUT is one of the most studied combinatorial optimization problems. Besides its theoretical importance, it has applications in circuit layout design, statistical physics, and many other fields; see, e.g., [5] or Section 6 of [11]. Given a graph $G = (V, E)$, the search version of MAX CUT consists of finding a partition (V_1, V_2) of the vertex set V, which maximizes the number of edges with one endpoint in V_1 and the other in V_2. Let us denote the maximum by $mc(G)$. For a constant c $(0 < c < 1)$ an algorithm is said to be a *c-approximation* if, for each input graph G, it finds a cut with at least $c \cdot mc(G)$ edges.

MAX CUT was proved to be *NP*-hard, even for graphs of maximum degree 3 [12]. The best approximation algorithm for general graphs was given by Goemans and Williamson [7], using semidefinite programming, and has approximation ratio 0.87856. For graphs of maximum degree 3, Halperin, Livnat and Zwick [8] adopted the same technique and established a 0.9326-approximation. Berman and Karpinski [2] showed that no polynomial-time algorithm can achieve the approximation ratio of 0.997 for MAX CUT in 3-regular graphs, unless *P=NP*.

The heavy tool of semidefinite programming, as an approach to MAX CUT, not only needs a phase of derandomization—that results in a large multiplicative constant as the precision of approximation increases—but also its fastest known implementation requires $\Theta(n^{3.5} \log n)$ time [1]. For this reason, it is of interest to investigate, how strong approximation can be achieved by faster algorithms. What is more, algorithms of a combinatorial nature usually give more insight to the structure of the problem.

Large cuts of *3-regular* graphs were studied by Hopkins and Staton [9]. Bondy and Locke [4], and later Halperin et al. [8], too, established combinatorial 4/5-approximation algorithms for MAX CUT in graphs of *maximum degree 3*. For the more restricted class of 3-regular graphs, also a 22/27-approximation algorithm is presented in [8] (only in the journal version). Both algorithms run in $O(n^2)$ time. More detailed comments on those results will be given in Section 4.

In this paper, we develop a different approach to MAX CUT in graphs of maximum degree 3. Our algorithm is the first combinatorial one after more than two decades that beats the 4/5 bound of [9] and [4] without assuming 3-regularity. In Section 3 we prove the following result.

Theorem 1 *There is an algorithm with running time $O(n^2)$ that establishes a 5/6-approximation for* MAX CUT *in any graph of maximum degree 3.*

Our main tool is a novel type of vertex decomposition for graphs of maximum degree 3, described in Corollary 2. As it will be proved at the end of Section 3, the structural properties of those decompositions allow us to select a cut in each partition class in such a way that all, or all but one, edges induced by the class belong to its cut. Moreover, those cuts can be combined to a cut F of the entire graph such that F contains the majority of edges joining distinct partition classes, and if some edge inside a class is not in F, then an extra gain can be guaranteed for F from the edges incident with that class. These properties, achieved for the partition classes sequentially, are extracted in Lemmas 3 and 4.

In this context, it is worth noting that other kinds of decompositions have already been used for lower bounds on Max Cut, for instance vertex partitions into bipartite induced subgraphs [10].

It remains an open problem whether our decomposition lemmas (Corollaries 1 and 2 of Section 2) can be established by algorithms faster than in quadratic time. Such improvements would yield better bounds in Theorem 1 as well, since the additional steps finding a large cut can be executed in *linear* time. Some comments concerning a way towards more efficient implementation are given in the concluding section.

Although the algorithm presented here performs better than the combinatorial ones known so far for Max Cut in graphs of maximum degree 3, it still remains an open problem to design a combinatorial approximation that beats the ratio obtained via semidefinite programming.

1.1 Terminology and notation

We consider graphs without loops and multiple edges. The following notation and terminology will be used:

$V(G)$ and $E(G)$: the vertex set and edge set of graph G

$G[S]$: the subgraph of graph G induced by vertex subset S

$\mathrm{e}(A,B)$: the number of edges with one endpoint in A and the other in B, where A and B are disjoint vertex sets

$G-H$: the subgraph of graph G induced by $V(G)\setminus V(H)$, where H is a subgraph of G

cubic graph: 3-regular graph

subcubic graph: graph of maximum degree at most 3

unicyclic graph: graph that is *connected* and has precisely one cycle

2 Unicyclic decompositions

In this section we prove some lemmas concerning vertex decompositions, that will play a central role in our Max Cut algorithm on subcubic graphs.

Lemma 1 *If $G=(V,E)$ is a connected subcubic graph with $|E|>|V|$, then there exists an induced subgraph H of G with the following properties:*

(i) H is unicyclic,

(ii) $G-H$ is connected,

(iii) Each edge joining H with $G-H$ is incident with the cycle of H.

Moreover, an induced subgraph H with these properties can be found in $O(|V|)$ time.

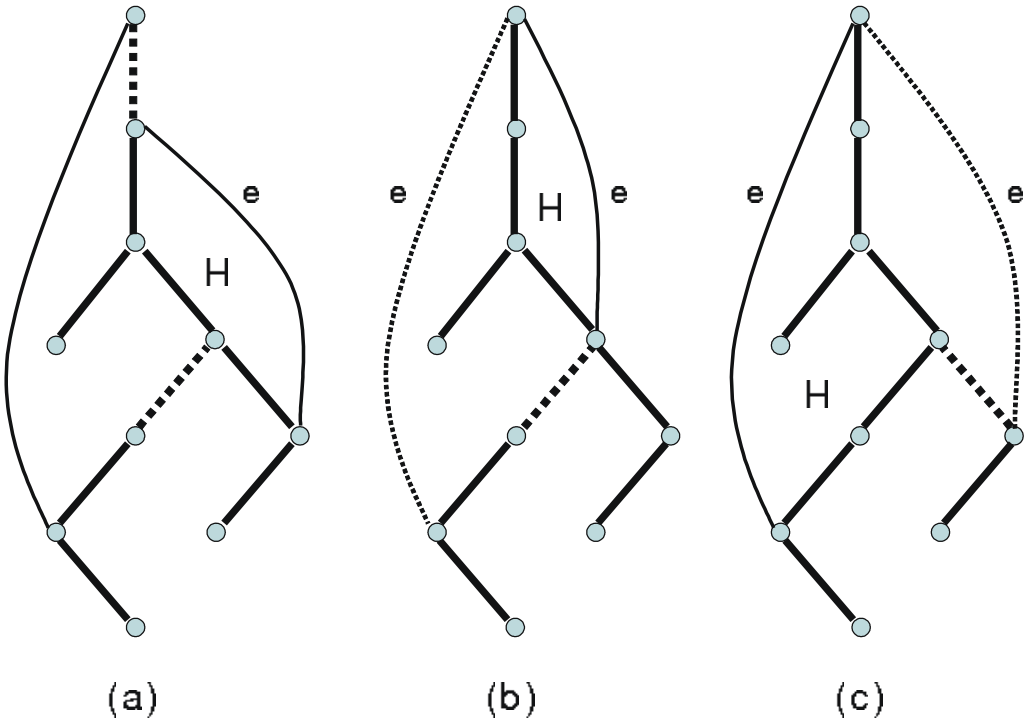

Figure 1: Construction of unicyclic subgraph H. Thick lines: tree edges; thin lines: non-tree edges; solid lines: edges in $E(H) \cup E(G - H)$; dotted lines: deleted edges, to detach H from $G - H$. (a) Unique edge with largest $m(e)$. (b) Two edges with $m(e) = m(e') = 1$, the endpoints of e and e' on the same rooted path. (c) Two edges with $m(e) = m(e') = 1$, the endpoints of e and e' on different branches of the spanning tree.

Proof Let us choose an arbitrary starting vertex r, and run Depth-First Search to find a spanning tree of G, say $T = (V, F)$, rooted at r. We label the vertices v with the natural numbers $1, 2, \ldots, |V|$, in the order they get visited during DFS. Denoting those labels with $a(v)$, we associate with each edge $e = uv \in E \setminus F$ the value $m(e) = m(uv) = \min\{a(u), a(v)\}$. When an edge of $E \setminus F$ is discovered during DFS, we register it at its endpoint of value $\max\{a(u), a(v)\}$. Moreover, a buffer is maintained for storing the non-tree edge with *currently largest* $m(e)$. Since G has maximum degree 3, each non-leaf vertex different from r is incident with at most one edge not in F; hence, the content of the buffer becomes uniquely determined, once a non-tree edge not incident with the root has been found. The case of $m(e) = m(e') = 1$ with two edges $e, e' \in E \setminus F$ will require a separate analysis, that we postpone until the end of the proof.

The way we find the subgraph H satisfying the requirements is illustrated in Figure 1. Having DFS terminated, we identify the edge $e = uv$ in the buffer. Assume $a(u) < a(v)$, without loss of generality. The subgraph H to be constructed will contain the edge uv and the u–v path P_{uv} of T. Moreover, for each $x \in V(P_{uv})$ we check whether x has a child outside P_{uv}. There can be at most one such child, because $P_{uv} \cup \{e\}$ is a cycle and

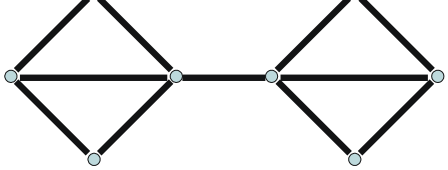

Figure 2: Graph with maximum degree 4 that becomes disconnected after the removal of any unicyclic induced subgraph.

G is subcubic.

If some x has a child x' outside P_{uv}, then we consider the subtree $T_{x'}$ rooted at x'. If $T_{x'}$ contains a vertex incident with a non-tree edge $e_x \in E \setminus F$, then we delete the edge xx' from T and select e_x instead, along which the subtree will be attached to the subgraph $G - H$ under construction. (Just one e_x is selected for each $T_{x'}$. Recall that $m(e_x) < m(e)$ necessarily holds, by the choice of e.)

The subtrees $T_{x'}$ not incident with any non-tree edge remain attached to P_{uv}, and their union together with P_{uv} induces the subgraph H. Now, any edge $e' \in E \setminus F$ distinct from e has $m(e') < m(e)$, therefore e' can have at most one vertex in H. For this reason, the unique non-tree edge induced by $V(H)$ is $e = uv$, and it is also clear that $E(H) \setminus \{e\}$ is a subtree of T, implying property (i). Also, (ii) is ensured by the choice of the edges e_x, whose insertion modifies T to a spanning tree of $G - H$. To verify (iii), we observe that a non-tree edge sharing a vertex with a subtree $T_{x'}$ either has been taken as e_x, or has become a non-tree edge in $G - H$ if another e_x has been chosen for T_x. Thus, each edge from H to $G - H$ is of the form xx', i.e. that joins some $T_{x'}$ with a vertex x of P_{uv}.

Finally, suppose that there are precisely two non-tree edges, say $e = rv$ and $e' = rv'$, both incident with the root r. Assuming $a(v) < a(v')$, one easy way to complete the construction is to remove e' and the last edge, say e'', on the path from v' to the cycle $P_{rv} \cup \{e\}$. What remains is a unicyclic graph containing e, and a tree component containing v'. The two edges between these subgraphs clearly satisfy (iii).

As regards complexity, our algorithm finding H is linear because DFS and also the additional operations described above can be implemented in $O(|V(G)|)$ time. □

Remark 1 The conclusion of Lemma 1 is not valid for graphs of higher degrees. A small counterexample is the graph shown in Figure 2, on 8 vertices, with degree sequence $4, 4, 3, 3, 2, 2, 2, 2$.

Remark 2 Although it does not lead to a simpler proof, let us note that the last case in the proof of Lemma 1—i.e., when there are just two non-tree edges and both of them are incident with the root—can be avoided by choosing r as a vertex of minimum degree. Indeed, if r has degree less than 3, then it can be incident with at most one non-tree edge; and if G is 3-regular, then each vertex, including r as well, is disjoint from at least one non-tree edge. The latter is true because there are $n - 1$ tree edges, plus at most two non-tree edges containing r, while the number of edges is $3n/2 > n + 1$ for $n \geq 4$. Alternatively, any leaf of the spanning tree is incident with two non-tree edges, and their other endpoints must be distinct.

Repeated application of Lemma 1 yields:

Corollary 1 *Every connected subcubic graph $G = (V, E)$ has a decomposition into vertex-disjoint connected graphs $H_1, \ldots, H_t$ for some natural number t, such that*

- *H_i is unicyclic for all $1 \leq i < t$,*
- *H_t is either unicyclic, or a tree joined to H_{t-1} with at least two edges (unless $t = 1$), and*
- *if an edge e joins some unicyclic H_i ($1 \leq i < t$) with any H_j ($i < j \leq t$), then e is incident with the cycle of H_i.*

Moreover, such a decomposition can be found in $O(|V|^2)$ time.

As it will turn out in the proof of Theorem 1, the last subgraph H_t is critical in some sense, and it is less favorable with respect to the computation if H_t is a tree or a unicyclic graph whose cycle is odd. In such a situation we need to impose a further condition on H_{t-1} or on H_t. For this purpose, the next assertion will be useful. The constant 126 comes from a very rough estimate, we have not made any efforts to optimize it.

Lemma 2 *Let $G = (V, E)$ be a connected subcubic graph, and let $H_1, \ldots, H_t$ be a decomposition determined in Corollary 1. If there are more than 126 edges between H_{t-1} and H_t, then the decomposition can be modified to $H_1, \ldots, H_{t-2}, H, H'$ in $O(|V|)$ time, such that H is unicyclic and satisfies condition (iii) of Lemma 1, H' is connected, and moreover the number of disjoint cycles in H' is larger than that in H_t.*

Proof We shall make use of the following results that are valid for any graph, not only for those with maximum degree 3. Erdős and Pósa [6] proved that, for some constant c, every graph with n' vertices and at least $n' + c\,k \log k$ edges contains k edge-disjoint cycles. They also showed that for two disjoint cycles, $n' + 4$ edges suffice. Moreover, Bodlaender [3] described a linear-time algorithm that either finds k vertex-disjoint cycles, or outputs a feedback vertex set (i.e., a set meeting all cycles) of cardinality at most $12k^2 - 27k + 15$. We shall apply these facts for $k = 2$ and $k = 3$. The number 126 (easily reducible to at most 86) comes from an estimated worst case for $k = 3$, as we shall explain in Case 2 below.

Let us emphasize that in subcubic graphs, edge-disjoint cycles necessarily are vertex-disjoint, too. In particular, this can make the formulation of the Erdős–Pósa theorem stronger when we apply it to G.

It will be convenient to distinguish between the two types of H_t, despite the arguments are fairly similar for them.

Case 1: H_t is a tree

We select a set E' of arbitrary five edges that join H_{t-1} with H_t. By the result of [6], the subgraph formed by $E' \cup E(H_{t-1}) \cup E(H_t)$ contains two vertex-disjoint cycles C, C'. The following simple argument shows that they can be found in linear time.

By assumption, the edges in E' are incident with the cycle of H_{t-1}, and their endpoints in H_{t-1} are distinct because G is subcubic. Hence, each pair of them is connected by two paths along the cycle of H_{t-1}, and by a unique path in the tree H_t. In this way,

we obtain 20 cycles, each containing just two edges of E'. Let G' denote the subgraph formed by the union of their edge sets. We claim that the two disjoint cycles C and C' must be among those 20 cycles. Indeed, they cannot coincide with the cycle of H_{t-1}, because removing the latter we obtain a forest (since H_t is a tree). And, any other cycle of G' contains four edges of E', therefore such a cycle must share an edge with each cycle of G'.

Given H_{t-1}, H_t, and the five edges of E', it is a trivial task to generate the 20 cycles mentioned above in linear time, and then it is easy to check which two of them are disjoint.

Since G' is connected, there is a path P between C and C'. Let us assume a vertex labeling as follows: $C = u_1, u_2, \ldots, u_p$; $P = v_1, v_2, \ldots, v_q$; $C' = z_1, z_2, \ldots, z_r$; $u_p = v_1$, $v_q = z_1$. Then, the path $P^* = u_1, \ldots, u_p/v_1, \ldots, v_q/z_1, \ldots, z_r$ (where the symbol '/' indicates identical vertices) is a feasible start of a DFS algorithm. Observe that in this subtree, $z_1 z_r$ is a non-tree edge that lies completely under C. Thus, applying the construction of Lemma 1 in the way that the first $p+q+r-3$ edge-selections in DFS are driven by P^*, the unicyclic subgraph H determined by the subroutine has no vertices above z_1. Consequently, no matter how DFS has terminated on G', the entire C remains in $G' - H$. This fact completes the proof of Case 1, as H_t was supposed to be a tree.

Case 2: H_t is unicyclic

Removing $42 = 12 \cdot 9 - 27 \cdot 3 + 15$ or fewer vertices from the induced subgraph $G' := G[V(H_{t-1}) \cup V(H_t)]$, the number of edges would decrease by at most 126, hence we would obtain a graph that still contains more edges than vertices. Thus, the algorithm of [3] cannot terminate with a feedback vertex set of cardinality at most 42. Consequently, three disjoint cycles can be found in G' in linear time. Let us denote them by C, C', C''.

We may assume without loss of generality that C and C' are connected by a path P that is vertex-disjoint from C''. We adopt the notation of Case 1 for the vertices of $C \cup P \cup C'$. Now again, we begin DFS with the path $u_1, \ldots, u_p/v_1, \ldots, v_q/z_1, \ldots, z_r$; moreover, when a vertex of C'' is reached for the first time, we continue DFS all around C''. In this way, the last edge of C'' becomes a non-tree edge, lying completely under $C \cup C'$. Thus, by an argument similar to the one given in Case 1, the algorithm of Lemma 1 for G' will leave the entire $C \cup C'$ in $G' - H$. This fact completes the proof of Case 2, and hence also of Lemma 2. □

Corollary 2 *Every connected subcubic graph $G = (V, E)$ has a decomposition into vertex-disjoint connected graphs $H_1, \ldots, H_t$ for some natural number t, such that*

- *H_i is unicyclic for all $1 \leq i < t$,*
- *if an edge e joins some H_i ($1 \leq i < t$) with any H_j ($i < j \leq t$), then e is incident with the cycle of H_i, and*
- *H_t has one of the following types:*
  - *(a) unicyclic, with an even cycle*
  - *(b) unicyclic, with an odd cycle, in which case also the cycle of H_{t-1} is odd*
  - *(c) $|V(H_t)| + 1 \leq |E(H_t)| \leq |V(H_t)| + 127$, with at least one even cycle in H_t.*

Moreover, such a decomposition can be found in $O(|V|^2)$ time.

Proof Repeatedly applying Lemma 1, we first generate the vertex-disjoint unicyclic graphs $H_1, H_2, \ldots, H_{i-1}$, as long as the graph $G_i = G - \left(\bigcup_{j=1}^{i-1} H_j\right)$ contains at least one cycle. From the Erdős–Pósa theorem we know that $|V(G_i)| \leq |E(G_i)| \leq |V(G_i)| + 3$, for otherwise we can find two further vertex-disjoint cycles and a refined decomposition with a larger value of i. Now, we first check the number, say s, of edges with one endpoint in H_{i-1} and the other in G_i. For $s > 126$, H_{i-1} gets re-defined along the lines of Lemma 2 and the algorithm continues, while the case of $2 \leq s \leq 126$ terminates with type (c), by re-defining $H_{i-1} := G[V(H_{i-1}) \cup V(G_i)]$ and setting $t = i - 1$; we shall argue at the end of the proof why an even cycle necessarily occurs in it. Similarly, if $s = 1$ and G_i is *not* unicyclic, we get type (c) by uniting H_{i-1} with G_i. Finally, if $s = 1$ and G_i is unicyclic, we set $t = i$, $H_t := G_t$, and check the parity of the unique cycle in H_{t-1} and in H_t. If the latter is even, then type (a) has been obtained, while if both are odd, then we have reached type (b). If the cycle in H_{t-1} is even and in H_t is odd, we switch H_{t-1} with H_t, also attaching their unique connecting subtree to the updated H_t. This yields type (a).

The existence of an even cycle in case (c) is implied by the facts that all the—at least three—cycles of H_t cannot be mutually disjoint, and that the union of two cycles sharing a nontrivial path always contains an even cycle. □

3 Finding a large cut

In this section we prove Theorem 1. The proof is based on Corollary 2, but we shall need some further assertions. The first one is quite simple, nevertheless we haven't found it published elsewhere. Allowing multiple edges in its formulation not only makes it slightly stronger but also simplifies the proof.

Proposition 1 *For every natural number k, there is a constant c_k with the following property: In each connected multigraph $G = (V, E)$ with $|E| < |V| + k$, the value of $mc(G)$ can be determined in at most $c\,|V| + c_k$ steps, for some small absolute constant c.*

Proof If G has a pendant vertex or triangle, it can be removed and then $mc(G)$ decreases by 1 or 2, and so does $|V|$ as well, while $|E| - |V|$ remains the same or decreases by 1. Otherwise, if there are two adjacent vertices of degree 2 which have distinct neighbors, then they are internal vertices of a (not necessarily induced) P_4, which can be contracted to an edge; this decreases both $|V|$ and $mc(G)$ by precisely 2 (also if a multiple edge is created), and keeps $|E| - |V|$ unchanged. These reductions can be executed in constant time and the graph remains connected after them, hence the proof can be completed by induction on $|V|$, with reference to smaller cases already verified.

If these transformations do not apply, then G is a connected multigraph of minimum degree at least 2, such that each edge is incident with a vertex of degree at least 3. Let us denote by x and y the numbers of vertices whose degree is greater than 2 and is equal to 2, respectively. The 2-edge stars on the y vertices are edge-disjoint, therefore

$$y \leq \frac{1}{2}\,|E|\,.$$

Summing up the degrees, we obtain

$$3x + 2y \leq 2|E|\,,$$

and so

$$3|V| = 3x + 3y \leq 2|E| + y \leq \frac{5}{2}\,|E| < \frac{5}{2}\,(|V| + k)\,,$$

from what we see $|V| < 5k$.

Consequently, $mc(G)$ can be determined by brute force in constant time whenever k is fixed. □

Remark 3 The formulation of Proposition 1—instead of writing simply $O(|V|)$—is intended to express the substantial difference between the two constants involved. While c_k may depend exponentially on k, the value of c is obtained from just what is needed for recognizing small-degree vertices and performing the local reductions above.

The next lemma is crucial for the proof of our main result.

Lemma 3 *Let $G = (V, E)$ be a connected subcubic graph, $V' \cup V'' = V$ a vertex partition, and $A' \cup B' = V'$ a partition inside V'. Let us assume:*

(1) $G[V'']$ is unicyclic, and its cycle is odd.

(2) All edges joining V' with V'' are incident with the cycle of $G[V'']$.

Then a partition $A'' \cup B'' = V''$ can be found in $O(|V''| + \mathrm{e}(V', V''))$ time, such that

(3) $E(G[A'']) \cup E(G[B''])$ consists of just one edge, and

(4) $\mathrm{e}(A', B'') + \mathrm{e}(B', A'') > \mathrm{e}(A', A'') + \mathrm{e}(B', B'')$.

We note that it will be essential to have strict inequality '$>$' instead of '$\geq$' in property (4).

Proof Let C be the cycle of $G[V'']$. For each edge e of C, the graph $G_e = G[V''] \setminus \{e\}$ is a tree, therefore it has a unique partition into two independent sets apart from their order; we denote them by A_e and B_e. Then both ordered partitions (A_e, B_e) and (B_e, A_e) satisfy property (3).

If there is an odd number of edges between V' and V'', then one of (A_e, B_e) and (B_e, A_e) is a proper choice for (A'', B'') satisfying property (4) as well, for any e.

Suppose that $\mathrm{e}(V', V'')$ is even. The proof will be done if we can identify an edge $e \in E(C)$ such that

$$\mathrm{e}(A_e, B') + \mathrm{e}(B_e, A') \;\neq\; \mathrm{e}(A_e, A') + \mathrm{e}(B_e, B')\,. \qquad (\star)$$

Since G is connected, there is a V'–V'' edge f incident with some vertex $v \in V''$. We denote by e and e' the two edges of the cycle in V'' that are incident with v. Note that e, e', f are all the edges on v, as G is subcubic. Since C is an *odd* cycle, we have $(A_{e'}, B_{e'}) = (A_e \setminus \{v\}, B_e \cup \{v\})$. Consequently, f is the only one edge of G that is counted for e and e' on different sides of $(\star)$. Thus, non-equality must hold for at least one of e and e'. □

For a cycle of even length, the conclusion is slightly different:

Lemma 4 *Let $G = (V, E)$ be a connected subcubic graph, $V' \cup V'' = V$ a vertex partition, and $A' \cup B' = V'$ a partition inside V'. If $G[V'']$ is unicyclic, and its cycle is even, then a bipartition $A'' \cup B'' = V''$ into independent sets can be found in $O(|V''| + \mathrm{e}(V', V''))$ time, such that $\mathrm{e}(A', B'') + \mathrm{e}(B', A'') \geq \mathrm{e}(A', A'') + \mathrm{e}(B', B'')$.*

Proof The bipartite connected graph $G[V'']$ has a unique partition into two independent sets, apart from their order. At least one of the two possible orders satisfies the requirement. □

Now we are in a position to prove 5/6-approximability.

Proof of Theorem 1 Let $H_1, H_2, \ldots, H_t$ be a decomposition of G guaranteed by Corollary 2. We are going to determine a vertex bipartition in reverse order for $H_t, H_{t-1}, \ldots, H_1$ as follows. For H_t, we find a cut that is maximum, on applying Proposition 1 if H_t is of type (c), or putting all of its edges into the cut if it is unicyclic with an even cycle (type (a)). For type (b), an arbitrary edge is deleted from the cycle of H_t; in this case, the computation will be slightly different—and simpler—but the construction for the H_i ($t > i \geq 1$) will be the same.

Assume that H_{i+1} has already been processed, for some $i > 1$. To generate a vertex bipartition of H_i, we set

$$V' \;=\; \bigcup_{j=i+1}^{t} V(H_j)\,, \qquad V'' \;=\; V(H_i)\,.$$

A partition on V' has been determined in the previous steps. We now apply Lemma 3 or 4, according as the cycle of H_i is odd or even, respectively. In this way, eventually a cut (A, B) of G is obtained; we will prove that it is a 5/6-approximation.

Let us denote by m_i the number of edges in H_i ($1 \leq i \leq t$), by ℓ the number of unicyclic H_i ($1 \leq i \leq t-1$) whose cycle is odd, and by n_t the number of vertices in H_t. Moreover, we write the value of maximum cut in H_t in the form $m_t - s$.

Case 1: H_t is of type (a) or (c)

In this case we have

$$m_t - s - n_t \;\geq\; 0$$

since H_t contains an even cycle that can be extended to a bipartite, connected, spanning subgraph of H_t.

Let m denote the number of edges in G. Since the H_i are vertex-disjoint, and every cut omits at least one edge from each odd cycle, we surely have

$$mc(G) \leq m - \ell - s\,.$$

On the other hand, according to Lemma 4 and condition (4) of Lemma 3, the algorithm produces a cut F of size at least

$$[(n - n_t - \ell) + (m_t - s)] + \frac{1}{2}\,[\,m - (m_1 + \cdots + m_t) + \ell\,]\,,$$

the first term counting the edges of F within the H_i and the second term estimating $\left|F \setminus \left(\bigcup_{i=1}^{t} E(H_i)\right)\right|$ from below. Here

$$m_1 + \cdots + m_t \;=\; n - n_t + m_t\,,$$

therefore the solution generated by the algorithm has value at least

$$\frac{1}{2}\,(n - n_t - \ell + m_t + m - 2s)\,.$$

Thus, the performance of the algorithm is at least as good as

$$\frac{1}{2}\,\frac{m-\ell-s+n-n_t+m_t-s}{m-\ell-s} \;=\; \frac{1}{2}+\frac{1}{2}\,\frac{n+m_t-n_t-s}{m-\ell-s} \;\geq\; \frac{1}{2}+\frac{n}{2m} \;\geq\; \frac{5}{6}$$

since both $m_t - n_t - s$ and $\ell + s$ are nonnegative, and $m \leq 3n/2$ in every subcubic graph.

Case 2: H_t is of type (b)

Assuming now that ℓ of the subgraphs H_i $(1 \leq i \leq t)$ have odd cycles, we see that the largest cut cannot exceed $m - \ell$, while the algorithm finds a cut of size at least

$$(n-\ell) + \frac{1}{2}\,(m - n + \ell - 1)\,.$$

The 'loss' of -1 in the last term may occur because we cannot benefit from Lemma 3 for the last subgraph H_t. Nevertheless, since the inequalities $\ell \geq 2$ and $n \geq 2m/3$ are valid, we obtain that the performance of the algorithm is at least as good as

$$\frac{1}{2}\,\frac{m+n-\ell-1}{m-\ell} \;=\; \frac{1}{2}+\frac{1}{2}\,\frac{n-1}{m-\ell} \;\geq\; \frac{1}{2}+\frac{1}{2}\,\frac{\frac{2}{3}m-1}{m-2} \;>\; \frac{5}{6}\,.$$

The decomposition in Corollary 2 can be found in $O(|V|^2)$ time. Then, the repeated application of Lemmas 3 and 4 takes

$$\sum_{1\leq i\leq t-1} O(|V_i|) + \sum_{1\leq i<j\leq t} O(\mathrm{e}(V_i,V_j)) = O(|V|) + O(|E|) = O(|V_i|)$$

steps. Thus, the algorithm can be implemented in $O(|V|^2)$ time. □

4 Comparison with earlier algorithms

In this concluding section we compare our results with those in [9], [4], and [8]—only the second one contains a time analysis among those three—and we also indicate how the time bound $O(n^2)$ may perhaps be improved in our case. In order to simplify notation, in the discussion below we assume throughout that $G = (V, E)$ is a subcubic graph with n vertices and $m \leq 3n/2$ edges.

Generally speaking, all the algorithms cited here apply the technique of *local switching*. It means that an initial (possibly partial) vertex partition is iteratively improved by changing the position of one or more vertices with respect to the cut.

- [9]: Hopkins and Staton only consider *3-regular* triangle-free graphs. Their argument for $mc(G) \geq 4m/5$ is presented as a proof by contradiction. Nevertheless, it can be converted to a *linear-time* algorithm. The obvious way of implementing the algorithm in [9] would be of order $O(n^2)$, and the price of going down to $O(n)$ is a more complicated book-keeping that not only registers vertex types—there are six of them—but also keeps track of the second neighborhood of each vertex.

- [4]: Bondy and Locke extend the inequality $mc(G) \geq 4m/5$ to *subcubic* triangle-free graphs, and also describe an efficient algorithm achieving this bound. As stated explicitly at the end of their Section 6 ([4, p. 490]), the time bound is $O(n^2)$. Although it is not impossible that a stronger worst-case bound might be proved, the difficulty is that in some situations it already takes $\Theta(n)$ steps just to find an improving switch, and in principle as many as $\Theta(n)$ of such iterations may occur.

  Of course, the algorithmic proof of the lower bound $mc(G) \geq 4m/5$ immediately implies a 4/5-approximation on triangle-free subcubic graphs. Moreover, perhaps a slightly stronger result might be obtained for regular graphs, since it is shown in [4] that the only two *3-regular* connected triangle-free graphs attaining $4m/5$ are the Petersen and the dodecahedron graphs. Nevertheless, this fact would probably not lead to a substantial improvement.

- [8]: Halperin et al. present two combinatorial algorithms, which we have to discuss one by one.

  Concerning subcubic graphs, it is observed in [8] that removing all triangles and their incident edges, then running the 4/5-approximation algorithm on the triangle-free subgraph obtained, and finally putting back the triangles one by one, can preserve the lower bound $4\,mc(G)/5$ for the size of the edge cut found. Although the steps handling triangles at the beginning and at the end can easily be organized in linear time, this approach uses the $O(n^2)$ algorithm of [4] as a subroutine, and hence in the worst case it cannot be faster than the one in [4] on triangle-free input graphs.

  On the other hand, the 22/27-approximation algorithm on *3-regular* graphs is self-contained, and it may be the case that it runs in $O(n)$ worst-case time. The difficulty in deciding this question lies in the fact that one of the seven local improvement steps applied in the algorithm consists of finding a cycle in an induced subgraph and switching some (about half) of its vertices; and in some situations it clearly may take $\Theta(n)$ time to find a cycle with as few as $O(1)$ vertices. So, to go down from $O(n^2)$ to $O(n)$—if this is possible at all—would require a careful way of organizing iterated improvements.

Summarizing, the algorithm designed in this paper not only beats all the previous ones with its 5/6-approximation ratio, but also it runs in $O(n^2)$ time, which is not worse than any of the time bounds proved for subcubic graphs in [4] and [8].

We do not know whether our decomposition lemmas (Corollaries 1 and 2) can be established by algorithms in fewer than $O(n^2)$ steps. Although the first unicyclic graph H_1 is found in linear time, and we do not need to search a spanning tree again from the

very beginning, the removal of H_1 may leave some branches of the remaining spanning tree in a non-DFS position. It is not clear how efficiently the rearrangement of those branches and the book-keeping of incident non-tree edges can be organized, and whether the total time of those additional steps becomes $o(n^2)$, or even as small as $O(n)$.

Assuming that a time bound $f(n)$ gets proved for Corollaries 1 and 2, the upper bound $f(n)+O(n) = O(f(n))$ would follow for Theorem 1 as well, since all the additional steps to construct a large edge cut from a unicyclic decomposition can be executed in linear time.

Acknowledgements. This research was supported by the bilateral research cooperation Balaton between EGIDE (France) and the Ministry of Education (Hungary) under grant numbers 07244RJ and F-29/2003. The second author was also supported in part by the Hungarian Scientific Research Fund, grant OTKA T-049613. We would like to express our thanks to the referees for careful reading and for constructive suggestions, based on which we have extended the explanation of several ideas in the proofs.

References

[1] F. Alizadeh, *Interior point methods in semidefinite programming with applications to combinatorial optimization*, SIAM Journal on Optimization, 5(1) (1995), 13–51.

[2] P. Berman and M. Karpinski, *On some tighter inapproximability results*, Proceedings of the 26th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science Vol. 1644, 1999, 200–209.

[3] H. L. Bodlaender, *On disjoint cycles*, International Journal of Foundations of Computer Science, 5(1) (1994), 59–68.

[4] J. A. Bondy and S. C. Locke, *Largest bipartite subgraphs in triangle-free graphs with maximum degree three*, Journal of Graph Theory, 10 (1986), 477–504.

[5] M. Deza and M. Laurent, *Applications of cut polyhedra. I, II*, Journal of Computational and Applied Mathematics, 55(2) (1994), 191–216, 217–247.

[6] P. Erdős and L. Pósa, *On the maximal number of disjoint circuits of a graph*, Publicationes Mathematicae Debrecen, 9 (1962), 3–12. MR0150756 (27#743)

[7] M. X. Goemans and D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, Journal of the ACM, 42 (1995), 1115–1145.

[8] E. Halperin, D. Livnat and U. Zwick, *MAX CUT in cubic graphs*, Journal of Algorithms, 53 (2004), 169–185. Preliminary version in : Proceedings of SODA 2002, 506–513.

[9] G. Hopkins and W. Staton, *Extremal bipartite subgraphs of cubic triangle-free graphs*, Journal of Graph Theory, 6 (1982), 115–121.

[10] N. V. Ngoc and Zs. Tuza, *Linear-time approximation algorithms for the max-cut problem*, Combinatorics, Probability and Computing, 2(2) (1993), 201–210.

[11] S. Poljak and Zs. Tuza, *Maximum cuts and large bipartite subgraphs*, Combinatorial Optimization (W. Cook et al., eds.), DIMACS series in Discrete Mathematics and Theoretical Computer Science Vol. 20, American Mathematical Society, 1995, 181–244.

[12] M. Yannakakis, *Node- and edge-deletion NP-complete problems*, Proceedings of the 10th ACM Symposium on the Theory of Computing, 1978, 253–264.