A simple max-cut algorithm for planar graphs

F. Liers and G. Pardella

Institut für Informatik, Universität zu Köln, Pohligstraße 1, D-50969 Köln,Germany,
{liers, pardella}@informatik.uni-koeln.de,
http://cophy.informatik.uni-koeln.de

Abstract. The MAX-CUT problem asks for partitioning the nodes V of a graph $G = (V, E)$ into two sets (one of which might be empty), such that the sum of weights of edges joining nodes in different partitions is maximum. Whereas for general instances the MAX-CUT problem is NP-hard, it is polynomially solvable for certain classes of graphs. For planar graphs, there exist several polynomial-time methods determining maximum cuts for arbitrary choice of edge weights. Typically, the problem is solved by computing a minimum-weight perfect matching in some associated graph. In this work, we present a new and simple algorithm for determining maximum cuts for arbitrary weighted planar graphs. Its running time can be bounded by $O(|V|^{\frac{3}{2}} \log |V|)$, similar to the fastest known methods. However, our transformation yields a much smaller associated graph than that of the known methods. Furthermore, it can be computed fast. As the practical running time strongly depends on the size of the associated graph, it can be expected that our algorithm is considerably faster than the methods known in the literature. More specifically, our program can determine maximum cuts in huge realistic and random planar graphs with up to 10^6 nodes.

1 Introduction

Partitioning problems in graphs have many relevant real-world applications. In its most basic version, the problem is to partition the nodes of a graph into two disjoint sets such that the weight of the edges connecting the two sets is either minimum or maximum. The former is denoted by MIN-CUT and the latter by MAX-CUT. Cut problems have many applications, e.g. in VIA minimization in the layout of electronic circuits, [1], in physics of disordered systems [2–4], or in network reliability [5]. Furthermore, the problem is equivalent to unconstrained quadratic 0-1 optimization [6, 7]. Several important combinatorial optimization tasks can naturally be formulated as constrained quadratic optimization problems. Investing knowledge from the unconstrained case often drastically speeds up the solution algorithms [8].

For nonnegative edge weights, the MIN-CUT problem can be solved using network flow techniques due to the famous duality of maximum flows and minimum cuts in networks [9], or by the algorithm proposed in [10].

For general edge weights, the MAX-CUT problem (and by inversion of the signum of the edge weights also the MIN-CUT problem) is NP-hard. We refer to

[11] and the references therein for a detailed study of different classes of instances marking the boundary between easy and hard ones. When restricting to certain graph classes, polynomial-time solution algorithms are known. This is true especially for planar graphs which are the subject of this article. Hadlock's algorithm [12] was the first polynomial-time algorithm for solving the MAX-CUT problem on planar graphs with nonnegative edge weights. In [13, 14] Barahona proposes a MAX-CUT algorithm for (arbitrary weighted) planar grid graphs, focussing on solving the two-dimensional planar Ising spin glass problem from theoretical physics. Furthermore, [15, 16] present a method that reduces the task to the Chinese-Postman problem. In 1990, Mutzel [17] proposed an algorithm using T-joins. In the same year, Shih, Wu, and Kuo [18] presented a mixed MAX-CUT algorithm for arbitrary weighted planar graphs, which generalizes the algorithm for optimal layer assignment of Kuo, Chern, and Shih [19]. It solves the problem in time bounded by $O(|V|^{\frac{3}{2}} \log |V|)$ which is presently the algorithm with the best worst-case running time. The method first constructs the dual graph. It is then expanded such that matchings in the latter correspond to cuts in the former. Moreover, a minimum-weight perfect matching in the latter yields an optimum cut in the original graph. In this work, we follow this general algorithmic scheme which leads to an algorithm with the same asymptotic running time as the one of Shih, Wu, and Kuo. However, in our transformation the expanded dual graph has a simpler structure and contains a considerably smaller number of both nodes and edges. As the bulk of the running time is spent in the matching computation and the latter scales with the size of the graph, our algorithm will be much faster in practice. Our new MAX-CUT algorithm for arbitrary weighted planar graphs is a generalization of the methods proposed in [20, 21] which are based on the work of Kasteleyn [22] from the 1960s.

In the following, we introduce some basic definitions and notations. In Section 3 we introduce and illustrate the algorithm and prove its correctness in Section 4. An analysis of the running time and space demand is presented afterwards in Section 5. In Section 6 possible algorithmic varieties are proposed. Finally, we present running times on realistic and random instances. It turns out that the algorithm can routinely solve the problem for random maximum planar graphs with up to 500,000 nodes and for realistic instances on planar graphs with up to 1,200,000 nodes.

2 Preliminaries

We consider simple, undirected, planar graphs $G = (V, E)$ with node set V and edge set E. We assume G is connected and real-weighted, i.e. each edge $e \in E$ is assigned a weight $w \in \mathbb{R}$. Multiple edges between two nodes can be contracted to a single edge with weight equal to the sum of the weights of these multiple edges. Self-loops can be omitted, as those edges will never be cut-edges. If not stated otherwise, we assume $|V| = n$ and $|E| = m$. Let $deg(v)$ denote the degree of a node $v \in V$, i.e. the number of edges incident to node v. A *path*, $\pi = v_1, v_2, ..., v_k,\ v_i \in V,\ i \in \{1, \ldots, k\},\ k \leq n$, is a sequence $\{v_1, v_2, ..., v_k\}$ of

pairwise different v_i such that $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$ are edges of G. A *closed path* $\pi = v_1, v_2, ..., v_k, v_1$ is called a cycle. A *subgraph* H of G is a graph such that every node of H is a node of G, and every edge of H is an edge in G also. We denote with K_n the complete graph with n nodes. Let $G = (V, E)$ be a weighted graph. For each (possibly empty) subset $Q \subseteq V$, the *cut* $\delta(Q)$ is the set of all edges $e = (u, v)$ with $u \in Q$ and $w \in V \setminus Q$. The weight of a cut is given by $w(\delta(Q)) = \sum_{e \in \delta(Q)} w(e)$. A *minimum cut* (MIN-CUT) asks for a cut $\delta(Q)$ with minimum weight $w(\delta(Q))$. As MAX-CUT is equivalent to MIN-CUT by negating weights, we concentrate on the minimization version of the problem. A connected graph $G = (V, E)$ is called *Eulerian* if and only if E can be partitioned into edge-disjoint cycles which is equivalent to saying that each node of G has even degree. A graph G is *planar* if it can be embedded in the plane in such a way that no two edges meet each other except at a node to which they are both incident. If a graph G is planar, then any embedding of G divides the plane into regions, called *faces*. One of these faces is unbounded, and called the *outer face*. A *geometric dual graph* G_D of a connected planar graph G is a planar graph with the following properties: G_D has a node for each face of G, and an edge for each edge joining two neighboring faces (including self-loops and multiple edges). A *matching* in a graph $G = (V, E)$ is a set of edges $M \subseteq E$ such that no node of G is incident with more than one edge in M. If some edge $m \in M$ is incident with a node $v \in V$, then v is *M-covered*, otherwise v is *M-exposed*. A matching M is *perfect* if every node is M-covered. The weight of M is the sum of weights of the edges in M.

3 The Algorithm

In the following we assume we are given a planar embedding of G. At first, we calculate its dual graph $G_D = (V_D, E_D)$, where the weight of a dual edge is chosen as $w(\tilde{e}) = w(e)$ if $\tilde{e} \in E_D$ is the dual edge crossed by $e \in E$. Subsequently, we split all dual nodes $\tilde{v} \in V_D$ with degree $deg(\tilde{v}) > 4$ into $\lfloor (deg(\tilde{v}) - 1)/2 \rfloor$ nodes and connect the copies by a path of new edges receiving zero weight. Let *split nodes* denote nodes created by a splitting operation. Edges incident to the original node are equally distributed among the split nodes such that the degree of each node is at most four, cf. Figure 1. We denote the resulting graph by $G_t = (V_t, E_t)$.

It is easy to see that after the splitting operations, no node in G_t has a degree smaller than three. Indeed, each face in a planar graph G is bounded by at least two edges. Bounding a face by exactly two edges is only possible if G has multiple edges which contradicts its simplicity. As G_D is the dual of G, we conclude that each node in G_D has degree at least three. Furthermore, a node in the transformed graph G_t has degree three or four.

The connectedness of G and G_D means that G_t is also connected. Moreover, certain structures in G_t can be excluded. For example, degree-four nodes with all edges being self-loops contradict the connectedness of the original graph G.

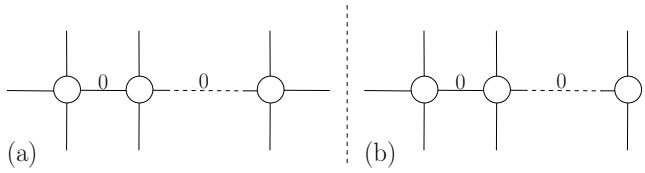

Fig. 1. (a) A node with even degree > 4 is split up in $\lfloor (deg(\tilde{v}) - 1)/2 \rfloor$ nodes. Edges are equally distributed among the split nodes. (b) A node with odd degree > 4 is split into $\lfloor (deg(\tilde{v}) - 1)/2 \rfloor$ nodes, each with degree four, except one receiving degree three.

From now on, we assume that G_t does not contain degree-four nodes with only self-loop edges.

We note that in the special case that G is a path graph of length two or three, P_2 (P_3), a node with degree two having a self-loop (with degree four having two self-loops, respectively) occurs. The algorithm we are going to present works here as well, and so we do not have to take special care of this case.

Next, we expand each node in G_t to a K_4 subgraph (a so-called Kasteleyn city [22]), while keeping the weights of the edges. Newly generated edges again receive zero weight. A node in G_t of degree three is expanded as displayed on the left of Figure 2, a degree-four node as shown on its right. A node with one self-loop is expanded as shown in Figure 3. We denote the resulting graph by G_E.

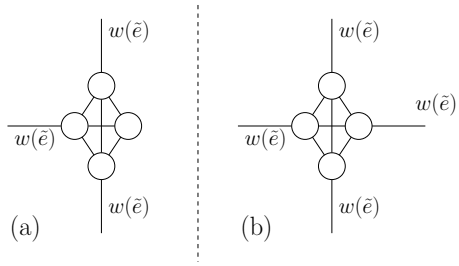

Fig. 2. Expansion of the nodes in G_t to K_4 subgraphs. (a) shows the subgraph for a node with degree three. (b) is generated in case the node has degree four. All edges in K_4 receive zero weight.

Next, we calculate a minimum-weight perfect matching M in G_E. Subsequently, we undo the transformation, i.e., shrink back all K_4 subgraphs and all (possibly created) split nodes, while keeping track of the matched edges. Consider the subgraph induced by the matching edges that are still present in the dual graph after shrinking. We will show in the next section that each node in this subgraph has even degree. This means that it is a minimum weight Eulerian graph which yields a MIN-CUT in the original graph. In the following section we will show the algorithmic flow outlined above on some small example.

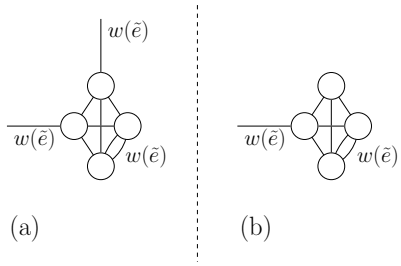

Fig. 3. Expansion for nodes having self-loops. (a) is the subgraph for a node with degree four and one self-loop. (b) is generated in case the node has degree three and one self-loop. All edges in K_4 receive zero weight.

3.1 Example

Consider as an example the planar drawing of the graph in Figure 4.

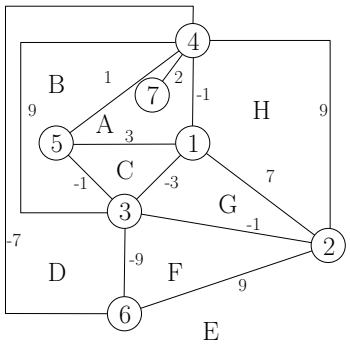

Fig. 4. Planar drawing of some planar graph G. Nodes are labeled by natural numbers. Edge weights are given as subscripts. Capital letters represent faces.

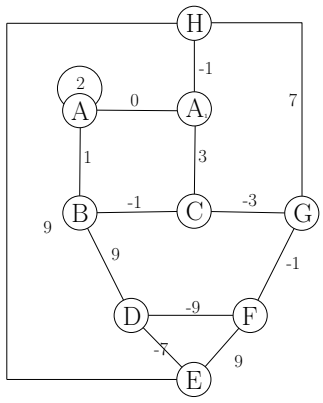

Fig. 5. The transformed dual graph G_t in which node A is split.

G consists of seven nodes and eight faces. Nodes are labeled with numbers and faces with capital letters. Edge weights are given as edge subscripts. The node representing face A is the only one having degree greater than four and is split into two nodes, cf. Figure 5. Figure 6 shows the graph after having expanded each node in G_t to a Kasteleyn subgraph.

On the transformed and expanded graph from Figure 6 we calculate a minimum-weight perfect matching (dotted edges). Shrinking back all artificial nodes yields a minimum-weight Eulerian subgraph of the dual and thus a MIN-CUT of the original graph (Figure 7).

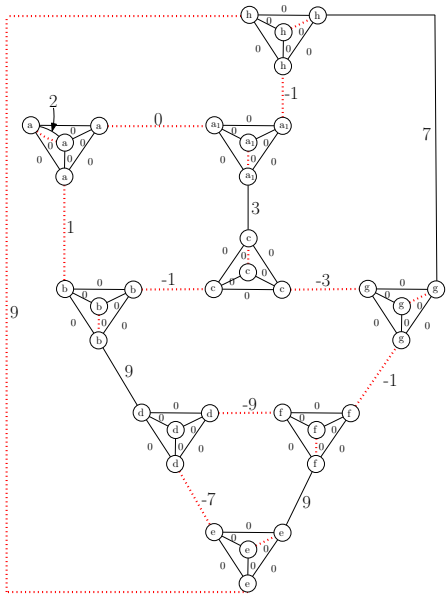

Fig. 6. The transformed dual graph G_E after expansion, together with a minimum-weight matching (dotted (red) edges).

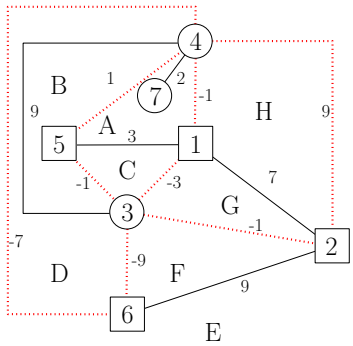

Fig. 7. After shrinking back the node copies, a MIN-CUT $\delta(Q)$ with weight $w(\delta(Q)) = -12$ emerges. Dotted (red) edges are cut-edges. Node partitions are indicated by different node shapes.

4 Correctness of the Algorithm

It is well known that there is a one-to-one correspondence between Eulerian subgraphs in the dual and cuts in its original graph. In this section, we show that the edge set induced by the minimum-weight perfect matching in G_E corresponds to a minimum-weight Eulerian subgraph in the dual and therefore to a minimum cut in the original graph.

To this end, we first need to show that there always exists a perfect matching M in the expanded graph G_E. Then, we need to prove that the constructed perfect matching in G_E induces a subgraph in the dual in which all node degrees are even.

We postpone for a moment the proof that a perfect matching exists. We call edges not contained in a K_4 *outgoing* and count the number of matched outgoing edges on some K_4 for an arbitrary perfect matching in G_E. Modulo analogous cases, we show in Figs. 8 and 9 all different possibilities for a matching covering all nodes of a K_4-subgraph together with its different number of outgoing edges. Analogous cases are those yielding the same number of outgoing matching edges.

Clearly, any possible matching in a K_4 subgraph leads to either zero, two or four outgoing matching edges with all nodes are M-covered. An odd number of outgoing matching edges always leaves an odd number of K_4 nodes unmatched, i.e. M-exposed, which contradicts the matching's perfectness.

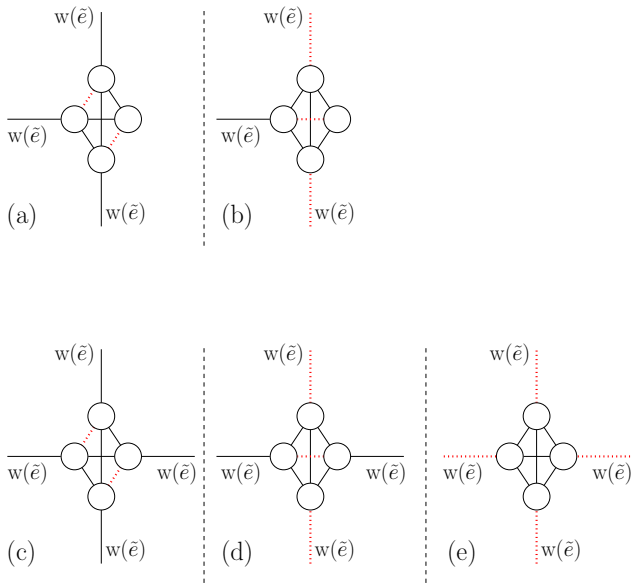

Fig. 8. Different cases for matched edges in a K_4 subgraph together with its outgoing edges, modulo cases in which the same number of outgoing edges is matched. (a) and (b) show possible matchings for subgraphs representing nodes with degree three in G_t. Figures (c), (d) and (e) show the possible matchings for subgraphs representing a node with degree four. Dotted (red) edges are matching edges.

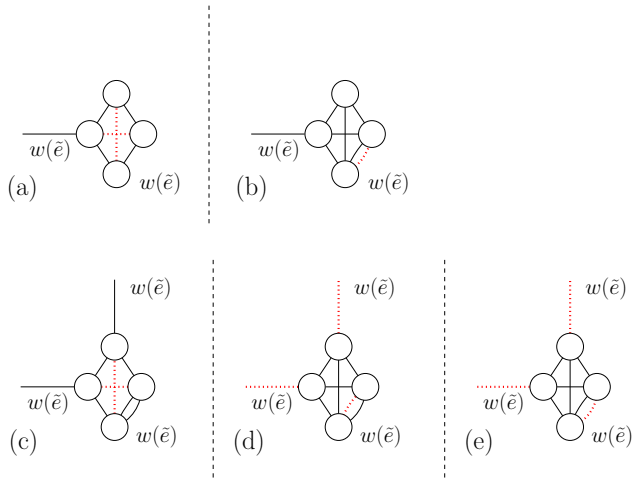

Fig. 9. Different cases for matched edges in a K_4 subgraph representing a node with self-loops (modulo analogous cases), together with its outgoing edges. Dotted (red) edges are matching edges. In Figures (b) and (e) the number of outgoing matching edges is two or four, resp., as the edge representing a former self-loop is matched.

Now we prove that the transformed graph G_E indeed has a perfect matching M. We first note that the graph is connected and has an even number of nodes (due to the inflation of each node to a K_4 subgraph). A trivial perfect matching exists as in each K_4 all nodes can be covered by matching edges contained in the K_4 (cf. Figures 8 (a) and (c) and 9 (a) and (c)). Therefore, a perfect matching in G_E always exists. One might ask whether there also always exists another perfect matching in which not only artificial edges contained in the K_4 subgraphs are matched. Indeed, as we deal with a geometric dual graph G_D, any two adjacent nodes in G_D (i.e., adjacent faces in G) are connected by at least one simple cycle. This cycle is expanded but preserved during the transformation of G_D to G_E. Thus, a possible nontrivial matching in G_t may match the edges in the cycle and additionally in each K_4 subgraph (representing a node on the cycle in G_D) an edge connecting two unmatched K_4 nodes, as shown in Figures 8 (b), (d) and (e). For all other Kasteleyn cities, edges contained in the K_4 can be matched. Each K_4 subgraph then has an even number of possible outgoing matching edges, cf. Figures 8-9.

Shrinking back the artificial nodes to the corresponding split nodes does not affect the number of outgoing matching edges. Consequently, after having collapsed all split nodes back to its dual nodes, each dual node has an even number of adjacent matching edges, too. Hence the matching induced subgraph is Eulerian and therefore defines a cut $\delta(Q)$ in the original graph G.

Yet the minimality of the cut is to be proven. It is

$$\begin{aligned} w(M) &= \sum_{\tilde{e} \in E_D \cap M} w(\tilde{e}) \\ &\underbrace{=}_{w(\tilde{e})=w(e)} \sum_{e \in \delta(Q)} w(e) \\ &= w(\delta(Q)) \end{aligned}$$

As $w(M)$ is the weight of a minimum-weight perfect matching, the weight of the induced Eulerian subgraph is minimum, and thus the weight of the cut $\delta(Q)$, too. We summarize this in the next theorem.

Theorem 1. *The algorithm described above computes a* MIN-CUT *(or* MAX-CUT*) in an arbitrarily weighted planar graph.*

5 Running-Time Analysis

After having shown the correctness of the method, we now concentrate on establishing bounds on its running time. We consider a maximum planar graph with n nodes, i.e. a triangulated planar graph in which each face is enclosed by a simple cycle of three edges. We will argue in the following that among all planar graphs with a specific number of nodes, the transformed graph G_E contains the maximum number of nodes and edges if G is triangulated.

Indeed, among all planar graphs with n nodes triangulated graphs have the maximum number of faces. For this class of graphs, our algorithm does not need

to split any dual node. A triangulation of a face with k nodes leads to $k-2$ new faces (new dual nodes, respectively), whereas a splitting operation yields only $\lfloor (k-1)/2 \rfloor$ nodes for the same face. As splitting operations result in a smaller number of nodes, we also need fewer edges (connecting split nodes) as in the triangulated case.

Obviously, given an embedding of a (maximum) planar graph, one can calculate the geometric dual in time bounded by $O(n)$. Furthermore, the described expansion of the dual graph can be done in time linear in n, (there are at most $3n-6$ edges). Next, the most time consuming step is performed - the calculation of a minimum-weight perfect matching.

Edmonds [23, 24] introduced one of the fundamental results in combinatorial optimization, i.e. the polynomial time blossom algorithm for computing minimum-weight perfect matchings. In its original version the algorithm runs in time bounded by $O(mn^2)$. Improved to $O(n^3)$ by Lawler [25] and Gabow [26] and later on by Gabow to $O(n(m+n\log n)$ [27]. Focusing on planar graphs, Lipton and Tarjan [28] have presented an $O(n^{\frac{3}{2}}\log n)$ divide-and-conquer algorithm for finding maximum-weight matchings using the planar separator theorem.

As our transformed graph G_E is not planar, this algorithm cannot be applied directly. However, a good separator of size $O(\sqrt{n})$ can be found for the planar dual graph G_D which directly implies a good separator of size $O(\sqrt{n})$ for G_t. Additionally, we observe that a matching M is a minimum-weight perfect matching in a weighted graph $G=(V,E)$ with $w: E \to \mathbb{R}$ if and only if M is a maximum-weight perfect matching in G with weight function $\tilde{w}: E \to \mathbb{R},\ \tilde{w}(e) := W - w(e)$ with W being a suitable large constant. With this considerations we can use the algorithm for maximum-weight matchings [29, 26, 23, 24], and are able to calculate a minimum-weight perfect matching in the graph G_E in $O(n^{\frac{3}{2}}\log n)$.

Finally, all nodes blown up in the transformation are shrunk back. Unshrinking can be done again in time $O(n)$. With these considerations, we state the following theorem.

Theorem 2. *Using the method described above, a* MIN-CUT *(or* MAX-CUT*) in a planar graph can be determined in time bounded by* $O(n^{\frac{3}{2}}\log n)$.

We show now that our method is less space demanding than the construction of [18] and leads to an algorithm that is faster in practice. Let F denote the set of faces of a maximum planar graph. Our method constructs a graph G_E with at most $|V_E| = 4|F| = 4(2n-4)$ nodes, as for each dual node we create four nodes, and the number of dual nodes in a maximum planar graph is at most $2n-4$. Its number of edges is $6|F|+|E_D| = 15n-30$, as we need to consider the original dual edges and those edges that are generated by the transformation of each dual node to a K_4 subgraph with six edges.

The transformation by Shih, Wu, and Kuo generates for each dual node a "star" subgraph of seven nodes and nine edges. On this graph a minimum-weight perfect matching is calculated which yields a maximum even-degree edge set of the dual graph, and therefore a MAX-CUT of the original graph. Thus, the method of Shih, Wu, and Kuo [18], yields an expanded dual graph with at least

$7(2n-4)$ nodes and $21n-42$ edges. These bounds are sharp as the first step of Shih, Wu, and Kuo is always a triangulation of the graph. Our transformation, in comparison, computes a matching on a much smaller and sparser graph, even in the case the graph is a triangulation. This makes our method in practice faster than the latter method. Moreover, the practical running time of the method of Shih, Wu, and Kuo might increase, as the matching induced even-degree edge set may be empty in which case an additional $O(n)$ time step is needed to compute a nontrivial even-degree edge set. A modification of the "star" subgraphs is performed, and the matching is recalculated using the planar-separator-theorem [28].

6 Algorithmic variants

Two straightforward algorithmic variants are presented. Let G denote a planar graph meeting the requirements introduced in Section 2. The first variant, which we call FCE algorithm, deals with the possibility to force edges to be in the cut. For example, this makes it possible to calculate a nonempty MIN-CUT in a graph. On the other hand it allows us also to find an $s-t$ cut in the graph, if nodes s and t are connected by an edge or can be connected by an edge without destroying the planarity of the graph.

The second variant, called PCE algorithm, can be seen as the reverse operation to the first variant. Precisely, we can force adjacent nodes to be in the same cut set by excluding the edge connecting these nodes from the set of potential cut-edges. If we want to group nonadjacent nodes, this can be done if again those nodes lie on the same face of G and can be connected by an edge without destroying planarity.

All operations explained in the following can be performed in time linear in n. We start with a detailed look at the first variant followed by a brief explanation of the second one.

Fixed cut edges - fce algorithm In order to force an edge to be in the set of cut-edges $\delta(Q)$ we propose the following FCE algorithm. Let edge $e=(v,w)\in E$ with $e\in\delta(Q)$, and let $e_E=(v_E,w_E)\in E_{G_E}$ be the corresponding edge in the expanded graph G_E. We denote with $G_E\setminus\{v_E,w_E\}$ the graph that arises from G_E by deleting nodes v_E and w_E and all incident edges to v_E and w_E. Clearly, a minimum-weight perfect matching on $G_E\setminus\{v_E,w_E\}$ yields a constrained MIN-CUT $\delta(Q)$ in the primal graph G with nodes v and w belonging to different node sets.

Theorem 3. *A* MIN-CUT $\delta(Q)$ *with the constraint* $e=(v,w)\in\delta(Q)$ *can be calculated with the* FCE *algorithm.*

Proof. Let $e_D\in E_D$ be the edge corresponding to edge e in $G_D=(V_D,E_D)$. As $e\in\delta(Q)$ holds, e_D and thus $e_E=(v_E,w_E)$ are matched. Therefore, we can remove v_E and w_E from G_E as these nodes are M-covered. Their removal yields

still a perfect matching in which each dual node has an even degree of matching edges after the shrink operation. Being more concrete, the following situations can occur in $G_E \setminus \{v_E, w_E\}$: a K_4 subgraph is reduced to a K_3 which has either one or three outgoing matching edges. A K_4 subgraph can be reduced to a K_2, here again there are two possibilities for outgoing matching edges. Either zero or two outgoing edges are matched. Finally, a K_4 subgraph can completely be removed. In all cases the sum of removed edges e_D (forced matched) and matched edges is even at each dual node. The minimality follows directly from Theorem 1. Thus, we arrive at constrained minimum-weight Eulerian subgraphs in the dual and thus at our desired constrained MIN-CUT $e \in \delta(Q)$ in the primal graph. □

As a direct consequence we conclude the following corollary.

Corollary 1. *Running the* FCE *algorithm m times, each time with a different fixed cut-edge, a nonempty* MIN-CUT *in G can be computed in time $O(mn^{\frac{3}{2}} \log n)$.*

Proof. For each edge $e \in E$ fix this edge as cut edge and run the FCE algorithm. In each run of the FCE algorithm a constrained MIN-CUT $\delta(Q)_e$ is calculated. Thus after m runs a set of minimum constrained cuts $Q_{fce} = \cup_{e \in E}\{\delta(Q)_e\}$ is calculated. The minimum nonempty cut $\delta(Q)$ is now given as:

$$\emptyset \neq \delta(Q) = \delta(Q)_e$$

with $w(\delta(Q)_e) = \min\{w(\delta(Q)_e) \mid \delta(Q)_e \in Q_{fce}\}$. □

Moreover we can state

Corollary 2. *Let $G = (V, E)$ be a planar graph and $s, t \in V$ two distinguished nodes. Using the* FCE *algorithm an s-t cut can be calculated iff s and t lie on the same face of G.*

Proof. Let s and t lie on face f of G. If there exists no edge connecting s and t, i.e. $e = (s, t) \notin E$, then insert such an edge e with weight zero to G and denote the resulting graph with $G \cup \{e\}$. Use the FCE algorithm with edge e as fixed cut-edge on $G \cup \{e\}$. The resulting cut $\delta(Q)$ separates s and t and is minimum among all those nonempty cuts. □

Prohibited cut edges - pce algorithm This variant works almost in the same manner. We want to prohibit edges to be in the cut, i.e. we want to specify adjacent nodes which are in the same partition.

We do not remove nodes and edges from the expanded graph G_E, like in the FCE algorithm, but only the edge connecting those nodes we want to be in the same partition. Let $e = (u, r) \in E$ with $\delta(Q) \cap e = \emptyset$. Now, let $e_E = (u_E, r_E) \in E_{G_E}$ be the corresponding edge in the expanded graph G_E. We consider the graph $G_E \setminus e_E$ that arises from G_E by removing edge e_E. Apparently, a minimum-weight perfect matching will never match this edge, thus it will never be a cut-edge. This can be seen as follows. The removal of edge e does not modify the internal structure of the K_4 subgraphs. This means - recall that we seek for

a perfect matching - there exists one node in each two K_4 subgraphs with no outgoing edge, i.e. this node only has incident edges to other nodes in this K_4 subgraph. This are exactly the nodes that are endnodes of edge e which has been removed. Each of these nodes must be matched to one other node in this K_4 subgraph. Again there are only two possibilities for outgoing matching edges, either zero or two, but the edge $e = (u, r)$ cannot be an outgoing matching edge. Therefore we achieved a grouping of node u and r.

7 Experiments

We implemented the algorithm from Section 3 using the OGDF library [30], and tested our implementation on a variety of problem instances, both realistic and randomly generated. All computational tests were carried out on Intel® Xeon© CPU E5410 2.33GHz (running under Debian Linux 4.1.1-21).

As the publicly available blossom program by Cook and Rohe [31] is one of the fastest state-of-the-art implementations for matching problems, we used their implementation as a black box for the matching part in our implementation.

The randomly generated instances are triangulated graphs with either uniformly distributed or Gaussian distributed edge weights, created using the Standford Graph Base [32] as part of the graph generator rudy [33]. It has the nice property that it produces identical random graphs for given seeds on most machine types.

% $(w(e) < 0)$ \ $\lvert V \rvert$	1000	2000	5000	15,000	50,000	80,000	100,000	500,000
10	0.07	0.14	0.36	2.92	16.18	27.47	34.36	127.37
30	0.14	0.28	0.73	8.91	49.46	88.58	110.82	425.42
50	0.15	0.29	0.78	9.83	54.03	95.96	120.64	483.78
70	0.15	0.29	0.78	9.80	53.86	95.02	120.66	484.79
100	0.13	0.25	0.67	8.52	47.13	82.28	103.15	438.97
Gaussian	0.19	0.39	1.28	13.86	77.25	137.89	178.79	715.08

Table 1. Random instances. Running times (in sec.) for various large maximum planar graphs. For the uniform edge weights (top), "% $(w(e) < 0)$" indicates the percentage of edges with negative weights. The number of edges is given by $3|V| - 6$.

In Table 1 average running times (in sec.) for minimum cut computations are given for various large random maximum planar graphs. We studied 100 instances for each choice of parameters (size and percentage of negative weights). We show results for triangulated graphs with up to 500,000 nodes. Even for the largest sizes studied, one min-cut computation only takes up to several minutes.

We also applied the algorithm to a special class of planar graphs that are often studied in the physics application, i.e. two-dimensional grid graphs. Determining maximum cuts in grid graphs is relevant for the study of so-called Ising

spin glasses. The results for grid graphs with edge weights following a bimodal distribution are given in Table 2, again always averaged over 100 instances.

$\|V\|$	average time (sec)
100^2	0.26
500^2	83.10
1000^2	1487.90
2000^2	16586.95
3000^2	59051.47

Table 2. Grid graph instances. Average running times (in sec.) over 100 instances. Each instance with $\frac{|E|}{2}$ edges having weight < 0.

The results of our experiments for random graphs show that the presented algorithm works very well. Even for large random maximum planar graphs the performance is good. Results for grid graphs obtained by using heuristic but high-quality variants of some MIN-CUT procedures are presented in the physics literature with sizes up to 480^2 [34] or 1801^2 grids [35]. With our algorithm, the study of considerably larger sizes is possible and allow a deeper insight in the physics of those systems.

We are not aware of any benchmarks or testsuites for cut algorithms on planar graphs. In order to study realistic instances, we took the TSPLIB library, maintained by Gerhard Reinelt [36]. As the realistic instances all have non-negative edge weights, we compute MAX-CUTs. For all geometric TSPLIB instances with at least $1,000$ nodes we computed Delaunay triangulations (using the LEDA library [37]) and set the Euclidean distance as edge weights. Results can be found in Table 3. Even for the largest instances the computation does not take longer than about 3 hours, often optimum cuts can be determined within seconds.

We also studied road network maps of the USA, taken from the 9th DIMACS Implementation Challenge (Shortest Paths) [38]. From the library, we took all instances with up to $1,200,000$ nodes. The largest instance took around 4.5 days to compute. The road network instances seem to be a bit harder computationally as they take longer to solve than random instances of comparable size, cf. Table 3. This is not surprising as for several applications one finds the same behavior that random instances are easier to solve than real-world problems. We are not aware of any available MAX-CUT or MIN-CUT implementation or experimental study of algorithms for planar graphs. However, as argued above, it is reasonable to expect that the new implementation is faster than other methods presented earlier in the literature.

The Delaunay triangulated TSPLIB instances, as well as the random instances (cf. Table 1), are computed fast. The construction of the transformed dual graph is done within a few seconds, and the bulk of the computation time is spent in the matching computation.

instance name ($\|E\|$)	average time (sec)
pla85900 (257604)	10248.70
pla33810 (101367)	390.50
usa13509 (40503)	169.73
brd14051 (42128)	140.49
d18512 (55510)	86.50
pla7397 (21865)	15.11
rl11849 (35532)	9.87
rl5934 (17770)	4.56
fnl4461 (13359)	3.21
rl5915 (17728)	2.84
pr2392 (7125), dsj1000 (2981), vm1748 (4784), rl1889 (5631)	< 2.00
rl1323 (3950), fl3795 (11326), u1060 (3153), rl1304 (3879), pcb3038 (9101), vm1084 (2869)	< 1.00
pr1002 (2972), u1432 (4204), d2103 (6290), u2319 (6869), u2152 (6312), d1291 (3845), u1817 (5386), d1655 (4890)	< 0.40
fl1577 (4643), nrw1379 (4115), fl1400 (4138), pcb1173 (3501)	< 0.20
instance name ($\|V\|$, $\|E\|$)	average time (sec)
USA-road-d.FLA (1,070,376 nodes, 2,712,798 edges)	394937.00
USA-road-d.NW (1,207,945 nodes, 2,840,208 edges)	168239.00
USA-road-d.NY (264,346 nodes, 793,002 edges)	117997.27
USA-road-d.BAY (321,270 nodes, 800,172 edges)	90486.00
USA-road-d.COL (435,666 nodes, 1,057,066 edges)	32227.10

Table 3. Realistic instances. Running times (in sec.) for MAX-CUT computation on different realistic instances. For the TSPLIB instances the number of nodes is encoded in the instance name, and the number of edges is given explicitly.

8 Conclusion

We presented a new MAX-CUT algorithm for arbitrary weighted planar graphs. Our approach is nifty and simpler than the methods presented earlier. Moreover, it is easy to implement. Its worst-case asymptotic running time is similar to the fastest algorithm used today. Furthermore, we showed in the computational experiments that it is very fast in practice and can compute optimum cuts in graphs with up to a million nodes. An interesting question is to explore whether the usage of planar separator strategies for the matching part could further reduce the computation time in practice.

Acknowledgments

We are grateful to Frank Baumann for verifying some of the computational results. Financial support from the German Science Foundation is acknowledged under contract Li 1675/1-1.

References

1. Barahona, F., Grötschel, M., Jünger, M., Reinelt, G.: An application of combinatorial optimization to statistical physics and circuit layout design. Operations Research **36**(3) (1988) 493–513
2. Hartmann, A.K., Rieger, H.: Optimization Algorithms in Physics. Wiley-VCH (2002)
3. Harary, F., ed.: Graph theory and theoretical physics. Academic Press, London (1967)
4. Liers, F., Jünger, M.M., Reinelt, G., Rinaldi, G. New Optimization Algorithms in Physics. In: Computing Exact Ground States of Hard Ising Spin Glass Problems by Branch-and-Cut. Wiley-VCH (2004) 47–68
5. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows. Prentice Hall Inc., Englewood Cliffs, NJ (1993) Theory, algorithms, and applications.
6. Boros, E., Hammer, P.L.: The max-cut problem and quadratic 0-1 optimization; polyhedral aspects, relaxations and bounds. Ann. Oper. Res. **33**(1-4) (1991) 151–180 Topological network design (Copenhagen, 1989).
7. de Simone, C.: The cut polytope and the boolean quadric polytope. Discrete Mathematics **79** (1989) 71–75
8. Buchheim, C., Liers, F., Oswald, M.: A basic toolbox for constrained quadratic 0/1 optimization. In: WEA. (2008) 249–262
9. Ford, Jr., L.R., Fulkerson, D.R.: Maximal flow through a network. Canad. J. Math. **8** (1956) 399–404
10. Stoer, M., Wagner, F.: A simple min-cut algorithm. J. ACM **44**(4) (1997) 585–591
11. McCormick, S.T., Rao, M.R., Rinaldi, G.: Easy and difficult objective functions for Max Cut. Math. Program. **94**(2-3, Ser. B) (2003) 459–466 The Aussois 2000 Workshop in Combinatorial Optimization.
12. Hadlock, F.: Finding a maximum cut of a planar graph in polynomial time. SIAM J. Comput. **4**(3) (1975) 221–225
13. Barahona, F.: On the computational complexity of Ising spin glass models. J. Physics A: Mathematical and General 15 **10** (1982) 3241–3253
14. Barahona, F., Maynard, R., Rammal, R., Uhry, J.P.: Morphology of ground states of two-dimensional frustration model . Journal of Physics A Mathematical General **15** (Feb 1982) 673–699
15. Barahona, F.: The max-cut problem on graphs not contractible to K_5. Oper. Res. Lett. **2**(3) (1983) 107–111
16. Barahona, F.: Planar multicommodity flows, max cut, and the Chinese-Postman problem. In: Polyhedral combinatorics (Morristown, NJ, 1989). Volume 1 of DIMACS Ser. Discrete Math. Theoret. Comput. Sci. Amer. Math. Soc., Providence, RI (1990) 189–202
17. Mutzel, P.: Implementierung und Analyse eines Max-Cut Algorithmus für planare Graphen (diploma thesis) (1990)
18. Shih, W.K., Wu, S., Kuo, Y.S.: Unifying maximum cut and minimum cut of a planar graph. IEEE Trans. Comput. **39**(5) (1990) 694–697
19. Kuo, Y.S., Chern, T.C., kuan Shih, W.: Fast algorithm for optimal layer assignment. In: DAC '88: Proceedings of the 25th ACM/IEEE conference on Design automation, Los Alamitos, CA, USA, IEEE Computer Society Press (1988) 554–559
20. Thomas, C.K., Middleton, A.A.: Matching Kasteleyn cities for spin glass ground states. Physical Review B (Condensed Matter and Materials Physics) **76**(22) (2007) 220406

21. Pardella, G., Liers, F.: Exact ground states of huge two-dimensional planar ising spin glasses. Technical report, Combinatorial Optimization in Physics (COPhy) (January 2008) submitted, preprint http://lanl.arxiv.org/abs/0801.3143v2.
22. Kasteleyn, P.W.: Dimer statistics and phase transitions. Journal of Mathematical Physics **4**(2) (1963) 287–293
23. Edmonds, J.: Paths, trees, and flowers. Can. J. Math. **17** (1965) 449–467
24. Edmonds, J.: Maximum matching and a polyhedron with 0-1 vertices. Journal of Research at the National Bureau of Standards **69B** (1965) 125–130
25. Lawler, E.L.: Combinatorial optimization: networks and matroids. Holt, Rinehart and Winston, New York (1976)
26. Gabow, H.N.: An efficient implementation of edmonds' algorithm for maximum matching on graphs. J. ACM **23**(2) (1976) 221–234
27. Gabow, H.N.: Data structures for weighted matching and nearest common ancestors with linking. In: SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (1990) 434–443
28. Lipton, R.J., Tarjan, R.E.: A separator theorem for planar graphs. SIAM J. Appl. Math. **36**(2) (1979) 177–189
29. Lipton, R.J., Tarjan, R.E.: Applications of a planar separator theorem. SIAM J. Comput. **9**(3) (1980) 615–627
30. OGDF: Open Graph Drawing Framework. http://www.ogdf.net (2007)
31. Cook, W., Rohe, A.: Computing minimum-weight perfect matchings. INFORMS Journal on Computing **11** (1999) 138–148
32. Knuth, D.E.: The Stanford GraphBase: a platform for combinatorial computing. ACM, New York, NY, USA (1993)
33. Rinaldi, G.: rudy – A rudimentary graph generator. https://www-user.tu-chemnitz.de/ helmberg/rudy.tar.gz (1998)
34. A.K.Hartmann, A.P.Young: Lower critical dimension of ising spin glasses
35. R.G.Palmer, J.A.: Ground states for large samples of two-dimensional ising spin glasses
36. Reinelt, G.: TSPLIB – A Traveling Salesman Problem Library. ORSA Journal on Computing **3** (1991) 376–384
37. Mehlhorn, K., Näher, S.: LEDA: A Platform for Combinatorial and Geometric Computing. Cambridge University Press (1999)
38. 9th DIMACS: Implementation challenge - shortest paths. http://www.dis.uniroma1.it/ challenge9/download.shtml (2005)