Available online at www.sciencedirect.com
SCIENCE DIRECT®
Operations Research Letters 33 (2005) 625–628
Operations Research Letters
www.elsevier.com/locate/orl

A fast algorithm for minimum weight odd circuits and cuts in planar graphs

Adam N. Letchford*, Nicholas A. Pearson

Department of Management Science, The Management School, Lancaster University, Lancaster LA1 4YX, UK

Received 13 September 2004; accepted 3 December 2004
Available online 8 January 2005

Abstract

We give a simple $O(n^{3/2} \log n)$ algorithm for finding a minimum weight odd circuit in planar graphs. By geometric duality, the same algorithm can be used to find minimum weight odd cuts. For general sparse graphs, the fastest known algorithms for these two problems take $O(n^2 \log n)$ time and $O(n^3 \log n)$ time, respectively.

Keywords: Planar graphs; Algorithms; Combinatorial optimization

1. Introduction

It has been known for a long time that various optimization problems defined on graphs can be solved much more quickly if the underlying graph is planar. For example, the fastest known deterministic algorithm for finding a *minimum weight cut* in a general graph with n vertices and m edges, due to Nagamochi et al. [17], runs in $\mathcal{O}(n(m+n \log n))$ time in the worst case, which reduces to $\mathcal{O}(n^2 \log n)$ when the graph is sparse. For planar graphs, however, Chalermsook et al. [6] recently presented an $\mathcal{O}(n \log^2 n)$ algorithm. Similarly, for the *maximum flow* problem, the fastest known (strongly polynomial) deterministic algorithm, due to Golberg and Tarjan [9], runs in $\mathcal{O}(nm \log(n^2/m))$, or $\mathcal{O}(n^2 \log n)$ for sparse graphs. For planar graphs, there exists an $\mathcal{O}(n \log n)$ algorithm; see Weihe [20]. (Actually, the article of Weihe contains some mistakes; see Brejova and Vinar [4] for a corrected algorithm.)

Most of these results for planar graphs rely on the famous *separator theorem* of Lipton and Tarjan [16]:

Theorem 1 (*Lipton and Tarjan [16]*). *Given a planar graph $G = (V, E)$, with n vertices, there exists a set of vertices $S \subset V$, called a separator, such that*:

- $|S|$ *is* $\mathcal{O}(\sqrt{n})$,
- *the removal of S disconnects the graph into two or more components*,
- *each such component contains at most* $2n/3$ *vertices*.

* Corresponding author. Tel.: +44-1524 594719; fax: +44-1524 844885.
E-mail address: a.n.letchford@lancaster.ac.uk (A.N. Letchford).

0167-6377/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.orl.2004.12.001

Moreover, a separator can be found in $\mathcal{O}(n)$ time.

The separator theorem, or generalizations of it, leads naturally to a variety of divide-and-conquer algorithms for various optimization problems on planar graphs.

Our goal in this paper is to give a fast and simple algorithm for the planar versions of two other well-known optimization problems: the *minimum odd cut* and *minimum odd circuit* problems. In Section 2 we briefly review the literature on these problems. In Section 3 we give the new algorithm and show that it runs in $O(n^{3/2} \log n)$ time. Some concluding remarks are made in Section 4.

2. Literature review

First we give a formal definition of our problems. Let $G = (V, E)$ be an undirected graph. A set of edges $C \subseteq E$ is called a *circuit* if it induces a connected subgraph of G, in which all vertices have degree two. A set of edges $F \subseteq E$ is called an *edge cutset* or *cut* if there exists a set of vertices $S \subseteq V$ such that $e \in F$ if and only if e has exactly one end-vertex in S.

Now assume that each edge $e \in E$ is labelled either *odd* or *even*. A circuit or cut is called *odd* if it contains an odd number of odd edges. Now assume that, in addition, each edge $e \in E$ is assigned a *weight* $w_e \geqslant 0$. The *minimum weight odd circuit* problem is that of finding an odd circuit of minimum total edge weight; the *minimum weight odd cut* problem is defined similarly. As well as being interesting combinatorial optimization problems in their own right, these two problems have applications in the polyhedral approach to various other combinatorial problems, including the *max-cut*, *stable set*, *matching* and *travelling salesman* problems; see Grötschel and Pulleyblank [11], Grötschel et al. [10], Padberg and Rao [19] and Padberg and Grötschel [18]. They also arise in heuristic methods for the separation of cutting planes for general integer programs (see, for example, Caprara and Fischetti [5] and Borndörfer and Weismantel [3]), and also appear in the literature on binary matroids and clutters (e.g., Grötschel and Truemper [12]).

We remark that, in some of these applications, it may be worthwhile allowing parallel edges in G (i.e., edges which share the same end-vertices). Note that instances with many parallel edges can be easily preprocessed so that there are at most two parallel edges between any given pair of end-vertices, namely, an odd-labelled edge and an even-labelled one.

To our knowledge, the first known algorithm for the minimum weight odd circuit problem is the one described by Grötschel and Pulleyblank [11], who attributed it to 'Waterloo folklore'. It involves a reduction of the problem to a sequence of m minimum weight perfect matching problems. Using the algorithm of Gabow [8] to solve these matching problems, this leads to an overall running time of $\mathcal{O}(nm(m + n \log n))$, or $\mathcal{O}(n^3 \log n)$ if the graph is sparse.

A simpler and faster approach was discovered independently by several authors (e.g., Barahona and Mahjoub [2], Grötschel et al. [10]). It works as follows. We create a new graph G', with twice as many vertices and edges, in the following way. For each $i \in V$, we have two vertices i', i'' in G'. For each even edge $\{i, j\}$ in G, we put two edges $\{i', j'\}$ and $\{i'', j''\}$ in G'; for each odd edge $\{i, j\}$ in G, we put two edges $\{i', j''\}$ and $\{i'', j'\}$ in G'. Then, by construction, a minimum weight odd circuit in G passing through any given vertex i corresponds to a shortest (i', i'')-path in G'. Hence, we can find the overall best circuit by solving n shortest-path problems in G'. Using, for example the fibonacci heap version of Dijkstra's shortest-path method (see Fredman and Tarjan [7]), this leads to a time of only $\mathcal{O}(n(m + n \log n))$ overall, or $\mathcal{O}(n^2 \log n)$ in sparse graphs.

Actually, although not mentioned in [2,10], there is a minor complication. It is possible that a shortest (i', i'')-path in G' yields, not an odd circuit in G, but an odd *cycle* which visits some vertices more than once. (By a cycle we mean a set of edges which induces a connected subgraph of G in which all vertices have even degree.) However, any such cycle will consist of the union of an odd circuit not passing through i and one or more even circuits, one of which will indeed pass through i. (Fig. 1 illustrates this possibility. The solid lines represent even edges and the dotted lines represent odd edges. The odd cycle shown contains three circuits, of which only the one on the right is odd.) Fortunately, we can ignore any such odd cycle generated as the algorithm proceeds, because (since edge-weights are non-negative) the odd circuit contained in the odd cycle will have the same or smaller weight than the odd cycle itself, and therefore we are

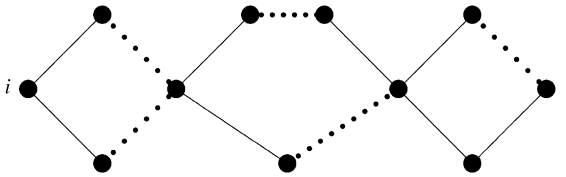

Fig. 1. A shortest odd cycle always contains one odd circuit.

guaranteed to find either that odd circuit or a better one later on.

Now we move on to briefly consider the minimum weight odd *cut* problem. The first, and essentially only, algorithm for this problem is due to Padberg and Rao [19]. It involves the solution of a sequence of up to $n-1$ max-flow problems. Using the pre-flow push algorithm of Goldberg and Tarjan [9] to solve these max-flow problems, the Padberg–Rao algorithm can be implemented to run in $\mathcal{O}(n^2 m \log(n^2/m))$ time, or $\mathcal{O}(n^3 \log n)$ if the graph is sparse. When the graph is planar, we can solve these max-flow problems using the Weihe algorithm, leading to $\mathcal{O}(n^2 \log n)$ overall.

We will give a simple $\mathcal{O}(n^{3/2} \log n)$ algorithm for both problems in the planar case.

3. The algorithms

It is well known (see, e.g., Harary [13]) that given any planar graph G, there is another planar graph, called the (geometric or combinatorial) dual, such that every minimal cut in G corresponds to a circuit in the dual and vice-versa. There is a one-to-one correspondence between edges in G and edges in the dual graph, and therefore every odd cut in G corresponds to an odd circuit in the dual. Moreover, constructing the dual takes $\mathcal{O}(n)$ time. Therefore, for our purposes it suffices to present an algorithm for the odd circuit case only.

In our context, the key is the trivial observation that the minimum weight odd circuit either passes through at least one vertex in the separator, or it does not. This leads to the following general approach to our problem:

0. Let $G=(V,E)$, $w \in \mathbb{R}^m_+$ and the parity labels be the input.
1. Find a separator S and let $G^1, \ldots, G^p$ be the associated components.
2. For each vertex i in the separator, find the minimum weight odd cycle passing through i. If this cycle is a circuit, and its weight is less than that of the best odd circuit found so far, store it in memory. (Otherwise, ignore it.)
3. Call this algorithm recursively with $G^1, \ldots, G^p$ as input.

The correctness of the algorithm is obvious. From Theorem 1, step 1 takes $\mathcal{O}(n)$ time. To find the minimum weight odd cycle passing through i in step 2, it suffices to find the shortest path from i' to i'' in the graph G' defined above. Note that G' is not necessarily planar, but it is sparse. Hence, using a good implementation of Dijkstra's method, each such shortest path computation takes $\mathcal{O}(n \log n)$ time. Since there are $\mathcal{O}(\sqrt{n})$ vertices in the separator, step 2 can be performed in $\mathcal{O}(n^{3/2} \log n)$ time. It then remains to be shown that subsequent recursive calls to the algorithm contribute a negligible amount to the overall running time, so that the entire minimum weight odd circuit problem can be solved in $\mathcal{O}(n^{3/2} \log n)$ time. This can be shown using a standard inductive argument.

Note that it is not actually necessary to use the fibonacci heap version of Dijkstra's method, since the simpler binary heap version [21] also runs in $\mathcal{O}(n \log n)$ time on sparse graphs. The resulting algorithm still has a time of $\mathcal{O}(n^{3/2} \log n)$ overall, but is much simpler to implement.

4. Concluding remarks

We have shown that the use of planarity enables us to save an $\mathcal{O}(\sqrt{n})$ factor in the case of odd circuits and an $\mathcal{O}(n^{3/2})$ factor in the case of odd cuts. It is natural to ask whether this running time can be further improved. We believe that it probably can be, for the following two reasons. First, we note that there is a (very complicated) linear-time algorithm due to Henzinger et al. [14] to solve planar shortest path problems. If we could adapt this algorithm to find shortest paths in G' in linear time, then we could save a $\log n$ factor. Second, it might be possible to adapt some of the ideas in the recent paper by Chalermsook et al. [6] to somehow perform step 2 of the above algorithm

more directly, without solving a series of shortest-path problems in an auxiliary graph.

It is also natural to ask what happens if we remove the restriction that edge-weights must be non-negative. For general real weights, it is easy to show that the planar minimum weight odd circuit problem is strongly $\mathcal{NP}$-hard, for example by reduction to Hamiltonian circuit on grid graphs [15]. By planar duality, the same holds for the planar minimum weight odd cut problem with general real weights. However, in the case of *conservative* weights (i.e., the case in which there may exist negative weights but no negative weight circuits), our algorithm can be modified to yield an $\mathcal{O}(n^{5/2})$ algorithm. The idea is simply to replace each call of Dijkstra's method with a call to the Moore–Bellman–Ford shortest-path algorithm, which can handle conservative weights. (The increased running time is due to the fact that the Moore–Bellman–Ford algorithm runs in $\mathcal{O}(n^2)$ time.)

Finally, we point out that our algorithm can also be used to solve the minimum weight odd circuit problem in more general classes of graphs for which separators of size $\mathcal{O}(\sqrt{n})$ exist and can be found efficiently, such as graphs which are not contractible to K_5 (see, for example, Barahona [1]). However, the algorithm cannot be used to solve the minimum weight odd *cut* problem in such graph classes, since the duality between cuts and circuits no longer holds.

References

[1] F. Barahona, The max-cut problem on graphs not contractible to K_5, Oper. Res. Lett. 2 (1983) 107–111.
[2] F. Barahona, A.R. Mahjoub, On the cut polytope, Math. Program. 36 (1986) 157–173.
[3] R. Borndörfer, R. Weismantel, Set packing relaxations of some integer programs, Math. Program. 88 (2000) 425–450.
[4] B. Brejova, T. Vinar, Weihe's algorithm for maximum flow in planar graphs, University of Waterloo, unpublished manuscript, 1999.
[5] A. Caprara, M. Fischetti, $\{0, 1/2\}$-Chvátal–Gomory cuts, Math. Program. 74 (1996) 221–235.
[6] P. Chalermsook, J. Fakcharoenphol, D. Nanongkai, A deterministic near-linear time algorithm for finding minimum cuts in planar graphs, in: J.I. Munro (Ed.), Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2004), Siam, Philadelphia, 2004.
[7] M.L. Fredman, R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, J. ACM 34 (1987) 596–615.
[8] H.N. Gabow, Data structures for weighted matching and nearest common ancestors with linking, in: Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, Siam, Philadelphia, 1990, pp. 434–443.
[9] A.V. Goldberg, R.E. Tarjan, A new approach to the maximum flow problem, J. ACM 35 (1988) 921–940.
[10] M. Grötschel, L. Lovász, A.J. Schrijver, Geometric Algorithms and Combinatorial Optimization, Springer, Berlin, 1988.
[11] M. Grötschel, W.R. Pulleyblank, Weakly bipartite graphs, Oper. Res. Lett. 1 (1981) 23–27.
[12] M. Grötschel, K. Truemper, Decomposition and optimization over cycles in binary matroids, J. Comb. Th. (B) 46 (1989) 306–337.
[13] F. Harary, Graph Theory, Addison–Wesley, Reading, MA, 1994.
[14] M.R. Henzinger, P. Klein, S. Rao, S. Subramanian, Faster shortest-path algorithms for planar graphs, J. Comput. System Sci. 55 (1997) 3–23.
[15] A. Itai, C.H. Papadimitriou, J. Szwarcfiter, Hamiltonian paths in grid graphs, SIAM J. Comput. 11 (1982) 676–686.
[16] R.J. Lipton, R.E. Tarjan, A separator theorem for planar graphs, SIAM J. Appl. Math. 36 (1979) 177–189.
[17] H. Nagamochi, T. Ono, T. Ibaraki, Implementing an efficient minimum cut algorithm, Math. Program. 67 (1994) 325–341.
[18] M.W. Padberg, M. Grötschel, Polyhedral computations, in: E. Lawler, J. Lenstra, A. Rinnooy Kan, D. Shmoys (Eds.), The Traveling Salesman Problem, Wiley, Chichester, 1985, pp. 307–360.
[19] M.W. Padberg, M.R. Rao, Odd minimum cut-sets and *b*-matchings, Math. Oper. Res. 7 (1982) 67–80.
[20] K. Weihe, Maximum (s, t)-flows in planar networks in $O(|V| \log |V|)$-time, J. Comp. Sys. Sci. 55 (1997) 454–476.
[21] J.W.J. Williams, Algorithm 232: Heapsort, ACM Commun. 7 (1964) 347–348.