ARTICLE IN PRESS
Discrete Applied Mathematics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Practical algorithms for branch-decompositions of planar graphs ☆

Zhengbing Bian [a], Qian-Ping Gu [b,*], Mingzhe Zhu [b]

[a] D-Wave Systems Inc., Canada
[b] School of Computing Science, Simon Fraser University Canada, Canada

ARTICLE INFO

Article history:
Received 11 March 2014
Received in revised form 18 October 2014
Accepted 17 December 2014
Available online xxxx

Keywords:
Graph algorithms
Branch-decomposition
Planar graphs
Computational study

ABSTRACT

Branch-decompositions of graphs have important algorithmic applications. A graph G of small branchwidth admits efficient algorithms for many NP-hard problems in G. These algorithms usually run in exponential time in the branchwidth and polynomial time in the size of G. It is critical to compute the branchwidth and a branch-decomposition of small width for a given graph in practical applications of these algorithms. It is known that given a planar graph G and an integer β, whether the branchwidth of G is at most β can be decided in $O(n^2)$ time, and an optimal branch-decomposition of G can be computed in $O(n^3)$ time. In this paper, we report the practical performance of the algorithms for computing the branchwidth/branch-decomposition of planar G and the heuristics for improving the algorithms.

1. Introduction

The notions of branchwidth and branch-decomposition are introduced by Robertson and Seymour [23] and have important algorithmic applications. Informally, a *branch-decomposition* of a graph G is a system of vertex cut sets of G represented as links of a tree whose leaves are the edges of G. The width of a branch-decomposition is the maximum cardinality of the vertex cut sets in the system. The *branchwidth* of G, denoted by $\mathrm{bw}(G)$, is the minimum width of all possible branch-decompositions of G. The branchwidth $\mathrm{bw}(G)$ and the *treewidth* $\mathrm{tw}(G)$ of graph G are linearly related: $\max\{\mathrm{bw}(G), 2\} \le \mathrm{tw}(G) + 1 \le \max\{\lfloor \frac{3}{2}\mathrm{bw}(G)\rfloor, 2\}$ for every G with more than one edge. A graph G of small $\mathrm{bw}(G)$ ($\mathrm{tw}(G)$) admits efficient dynamic programming algorithms for many NP-hard problems [2,5]. These algorithms have two major steps: (1) compute a branch-/tree-decomposition of G and (2) apply a dynamic programming algorithm based on the decomposition to solve the problem. Step (2) usually runs in polynomial time in the size of G and exponential time in the width of the decomposition computed in Step (1). To apply branch-decomposition based algorithms in practice, it is important to compute a branch-decomposition of small width efficiently.

Deciding the branchwidth/treewidth and computing a branch-/tree-decomposition of minimum width have received much attention. Given an arbitrary graph G of n vertices and an integer β, it is NP-complete to decide whether $\mathrm{bw}(G) \le \beta$ [26] ($\mathrm{tw}(G) \le \beta$ [1]). If the branchwidth (treewidth) is upper-bounded by a constant then both the decision problem and the optimal decomposition problem can be solved in $O(n)$ time [6,7]. However, the constants behind the Big-Oh are huge

☆ Part of the work appeared in the Proc. of the 9th SIAM Workshop on Algorithm Engineering and Experiments (ALENEX2008) Bian et al. (2008) [4] and the Proc. of the 7th International Workshop on Experimental Algorithms (WEA 2008) Bian and Gu (2008) [3].

* Corresponding author.
E-mail addresses: zbian@dwavesys.com (Z. Bian), qgu@sfu.ca (Q.-P. Gu), mingzhez@sfu.ca (M. Zhu).

http://dx.doi.org/10.1016/j.dam.2014.12.017
0166-218X/© 2015 Published by Elsevier B.V.

and the linear time algorithms are mainly theoretically interesting. For arbitrary graphs, the best known approximation factor in polynomial time is $O(\sqrt{\log \beta})$, where β is $\mathrm{bw}(G)$ or $\mathrm{tw}(G)$ [10]. Constant-factor $2^{O(\mathrm{bw}(G))}n^{O(1)}$ ($2^{O(\mathrm{tw}(G))}n^{O(1)}$) time algorithms for branchwidth/treewidth are given in [18,24].

For a planar graph G, the branchwidth and optimal branch-decomposition can be computed efficiently. Seymour and Thomas give a rat-catching algorithm which decides whether $\mathrm{bw}(G) \leq \beta$ in $O(n^2)$ time and an edge-contraction algorithm which computes an optimal branch-decomposition in $O(n^4)$ time [26]. An attractive feature of these algorithms is that there is no huge hidden constant in the running time. Notice that it is open whether computing the treewidth is NP-complete or not and the above algorithms are 1.5-approximation algorithms for the treewidth and optimal tree-decompositions of planar graphs. Motivated by the results of Seymour and Thomas, branch-decompositions of planar graphs and their algorithmic applications have received much attention. Readers may refer to the papers of [5,9,8,16] for extensive literature in the theory and application of branch/tree-decompositions.

Computing an optimal branch-decomposition is a major step in branch-decomposition based algorithms for NP-hard problems in planar graphs. The edge-contraction algorithm calls the rat-catching algorithm as a sub-routine $O(n^2)$ times to construct an optimal branch-decomposition of a planar graph in $O(n^4)$ time. Gu and Tamaki give an improved edge-contraction algorithm which calls the rat-catching algorithm $O(n)$ times to construct an optimal branch-decomposition of a planar graph in $O(n^3)$ time [13]. Hicks proposes a divide-and-conquer heuristic (called cycle method) to reduce the calls of the rat-catching algorithm for computing optimal branch-decompositions of planar graphs [14,15]. In the worst case, the cycle method has $O(n^4)$ running time, but in practice it is faster than the edge-contraction algorithm.[1] Hicks reports that the edge-contraction algorithm and the cycle method can solve instances of about 2000 edges and 7000 edges, respectively, in a practical time [14,15].

In this paper, we report practical progresses in computing optimal branch-decompositions of planar graphs. Because all known algorithms for optimal branch-decompositions of planar graphs use the rat-catching algorithm as a subroutine, the efficiency of the rat-catching algorithm is a key for computing optimal branch-decompositions of large planar graphs. We report efficient implementations of the rat-catching algorithm. These implementations can compute the branchwidth of planar graphs of size up to tens of thousand edges in a practical time. Using the efficient implementations of the rat-catching algorithm, the edge-contraction algorithms can compute optimal branch-decompositions of planar graphs with up to 7,000 edges in a practical time by a computer with a CPU of about 3 GHz and a memory of 2 GByte.

One progress for improving the edge-contraction algorithms is a multiple edge-contraction heuristic to reduce the calls of the rat-catching algorithm. The edge-contraction algorithms construct an optimal branch-decomposition of a planar graph G by finding a sequence of contractible edges in the medial graph $M(G)$ of G (the definition of $M(G)$ is given in the next section). Informally, an edge e of $M(G)$ is contractible if the contraction of e in $M(G)$ will produce a part of an optimal branch-decomposition of G. The edge-contraction algorithms contract an edge e and then uses the rat-catching algorithm to check if e is contractible. The multiple edge-contraction heuristic contracts multiple edges $e_1, \ldots, e_k$ then check if all $e_1, \ldots, e_k$ are contractible by one call of the rat-catching algorithm. In the worst case, the multiple edge-contraction heuristic has running time $O(n^3)$, but is faster than the edge-contraction algorithms in practice. Using efficient implementations of the rat-catching algorithm, the multiple edge-contraction heuristic can compute optimal branch-decompositions for some instances of size up to 20,000 edges in a practical time.

We also introduce an improved cycle method. The cycle method proposed in [14,15] partitions G into subgraphs, finds an optimal branch-decomposition of each subgraph recursively and combines the solutions of subgraphs into an optimal branch-decomposition of G. The cycle method calls the rat-catching algorithm to check the branchwidth of each subgraph at each recursive step. The cycle method is reported faster in practice than the edge-contraction algorithm of [26] by a factor of 10–30 in average for a class of planar graphs (Delaunay triangulation instances) [15]. The improved cycle method uses a better strategy to partition G into subgraphs and is faster than the edge-contraction algorithms of [26,13] by a factor of 200–300 for Delaunay triangulation instances of large size. Using efficient implementations of the rat-catching algorithm, the improved cycle method can compute optimal branch-decompositions for some instances of size up to 50,000 edges in a practical time.

The rest of the paper is organized as follows. Section 2 gives the preliminaries. We introduce the efficient implementations of the rat-catching algorithm, the multiple edge-contraction heuristic, and the improved cycle method in Sections 3–5, respectively. The final section concludes the paper.

2. Preliminaries

A graph G consists of a set $V(G)$ of vertices and a multi-set $E(G)$ of edges, where each edge e of $E(G)$ is a subset of $V(G)$ with at most two elements. For a set $A \subseteq E(G)$ of edges let $V(A) = \cup_{e \in A} e$ be the set of vertices in edges of A. We say a vertex v and an edge e are incident to each other if $v \in e$. We denote by $\deg_G(v)$ the number of edges in G incident to v and by $\Delta(G)$ the largest $\deg_G(v)$ for all $v \in V(G)$. For a subset $X \subseteq V(G)$, we denote by $\deg_G(X)$ the number of edges in G incident

[1] The cycle method uses the edge-contraction algorithm as a supplementary step. If the improved edge-contraction algorithm is used then the cycle method has $O(n^3)$ running time.