ARTICLE IN PRESS

European Journal of Combinatorics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

European Journal of Combinatorics

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

A fast scaling algorithm for the weighted triangle-free 2-matching problem[☆]

S. Artamonov [a], M. Babenko [b]

[a] *Moscow State University, Russian Federation*

[b] *National Research University Higher School of Economics (HSE), Russian Federation*

ARTICLE INFO

Article history:
Available online xxxx

ABSTRACT

A *perfect 2-matching* in an undirected graph $G = (V, E)$ is a function $x : E \to \{0, 1, 2\}$ such that for each node $v \in V$ the sum of values $x(e)$ on all edges e incident to v equals 2. If $\mathrm{supp}(x) = \{e \in E \mid x(e) \neq 0\}$ contains no triangles then x is called *triangle-free*.

Polyhedrally speaking, triangle-free 2-matchings are harder than 2-matchings, but easier than usual 1-matchings.

Given edge costs $c : E \to \mathbb{R}_+$, a natural combinatorial problem consists in finding a perfect triangle-free matching of minimum total cost. For this problem, Cornuéjols and Pulleyblank devised a combinatorial strongly-polynomial algorithm, which can be implemented to run in $O(VE \log V)$ time. (Here we write V, E to indicate their cardinalities $|V|$, $|E|$.)

If edge costs are integers in range $[0, C]$ then for both 1- and 2-matchings some faster scaling algorithms are known that find optimal solutions within $O(\sqrt{V\alpha(E, V) \log V}E \log(VC))$ and $O(\sqrt{V}E \log(VC))$ time, respectively, where α denotes the inverse Ackermann function. So far, no efficient cost-scaling algorithm is known for finding a minimum-cost perfect triangle-free 2-matching. The present paper fills this gap by presenting such an algorithm with time complexity of $O(\sqrt{V}E \log V \log(VC))$.

☆ This is an extended version of a conference paper Artamonov and Babenko (2016) [1].

E-mail addresses: stiartamonov@gmail.com (S. Artamonov), maxim.babenko@gmail.com (M. Babenko).

http://dx.doi.org/10.1016/j.ejc.2017.07.008

1. Introduction

1.1. Basic notation and definitions

We shall use some standard graph-theoretic notation throughout the paper. For an undirected graph G, we denote its sets of vertices and edges by $V(G)$ and $E(G)$, respectively. Unless stated otherwise, we allow parallel edges and loops in graphs. A subgraph of G induced by a subset $U \subseteq V(G)$ is denoted by $G[U]$. For $U \subseteq V(G)$, the set of edges with one end in U and the other in $V(G) - U$ is denoted by $\delta_G(U)$; for $U = \{u\}$, the latter notation is shortened to $\delta_G(u)$. Also, $\gamma_G(U)$ denotes the set of edges with both endpoints in U. When G is clear from the context, it is omitted from notation.

For a path $P = v_0 e_0 v_1 e_1 \dots v_k e_k v_{k+1}$ viewed as an alternating sequence of vertices and edges, we denote its reverse by $\overline{P} = v_{k+1} e_k v_k \dots e_1 v_1 e_0 v_0$; for two paths P_1, P_2 such that the last vertex of P_1 matches the first vertex of P_2, $P_1 \circ P_2$ stands for their concatenation. For an arbitrary set W and a function $f : W \to \mathbb{R}$, we denote its *support set* by $\mathrm{supp}(f) = \{w \in W \mid f(w) \neq 0\}$. For an arbitrary subset $W' \subseteq W$, we write $f(W')$ to denote $\sum_{w \in W'} f(w)$.

The following objects will be of primary interest throughout the paper:

Definition 1. Given an undirected graph G, a *2-matching* in G is a function $x : E(G) \to \{0, 1, 2\}$ such that $x(\delta(v)) \leq 2$ for all $v \in V(G)$. If $x(\delta(v)) = 2$ for all $v \in V(G)$ then x is called *perfect*. A vertex v is called *free* from x if $x(\delta(v)) = 0$. If $\mathrm{supp}(x)$ contains no triangles then x is called *triangle-free*.

Consider some non-negative real valued edge costs $c : E(G) \to \mathbb{R}_+$. Then a natural combinatorial problem consists in finding a perfect triangle-free 2-matching x of minimum total cost $c \cdot x$. For this problem, Cornuéjols and Pulleyblank [3] devised a combinatorial polynomial algorithm. While they were not aiming for the best time bound, it is not difficult to implement their algorithm to run in $O(VE \log V)$ time (hereinafter in complexity bounds we identify sets with their cardinalities).

1.2. Related work and our contribution

Now let edge costs be integers in $[0, C]$. The problem of finding a perfect triangle-free 2-matching of minimum cost is closely related to other problems in matching theory, for which some faster cost-scaling algorithms are known.

First, we may allow triangles in $\mathrm{supp}(x)$ and ask for a perfect 2-matching of minimum cost. This problem is trivially reducible to minimum cost perfect bipartite matching. (Indeed, we create two vertices v_1, v_2 for each vertex v and add two edges $e_1 = \{u_1, v_2\}$, $e_2 = \{u_2, v_1\}$ with $c(e_1) = c(e_2) = c(e)$ for each edge $e = \{u, v\}$.) A classical algorithm [7] based on cost scaling and blocking augmentations solves this problem in $O(\sqrt{V}E \log(VC))$ time.

Second, in Definition 1 we may replace $x(\delta(v)) \leq 2$ by $x(\delta(v)) \leq 1$ and get the usual notion of *1-matchings*. For general graphs G, a sophisticated algorithm from [8] solves the minimum-cost perfect matching problem within $O(\sqrt{V\alpha(E, V) \log V}E \log(VC))$ time.

For a related but somewhat harder case of *simple* triangle-free 2-matchings (where x is only allowed to take values 0 and 1), a good survey was done by Kobayashi [12].

Some relevant prior art also exists for the unweighted case, where the goal is to find a matching with maximum *size* $x(E(G))$. For unweighted 2-matchings (or, equivalently, 1-matchings in bipartite graphs), Hopcroft and Karp devised an $O(\sqrt{V}E)$ time algorithm [11] (by use of *clique compression*, the latter bound was improved to $O(\sqrt{V}E \log_V(V^2/E))$ in [6]). Later, a conceptually similar but much more involved $O(\sqrt{V}E)$-time algorithm [13] for matchings in general graphs was devised (and its running time was similarly improved to $O(\sqrt{V}E \log_V(V^2/E))$ in [9]).

Concerning unweighted triangle-free 2-matchings, Cornuéjols and Pulleyblank [4] gave a natural augmenting path algorithm; with a suitable implementation, its time complexity is $O(VE)$. To match the latter with the complexity of 1- and 2-matchings, [2] proposed a method that reduces the problem to a pair of maximum 1-matching computations. Unfortunately, this approach does not seem to extend to weighted problems.

Apart from the primal–dual algorithm given in [3], no other methods for solving the weighted perfect triangle-free 2-matching problem are known. In particular, no efficient cost scaling algorithm