Amber Q. Chen, Kevin K. H. Cheung, P. Michael Kielstra & Avery D. Winn

Revisiting a Cutting-Plane Method for Perfect Matchings

Volume 1 (2020), article no. 2 (15 pages)

<http://ojmo.centre-mersenne.org/item/OJMO_2020__1__A2_0>

Article submitted on June 15, 2020, revised on September 24, 2020, accepted on November 30, 2020.

Revisiting a Cutting-Plane Method for Perfect Matchings

Amber Q. Chen
University of Waterloo, 200 University Ave. W., Waterloo, ON N2L 3G1, Canada
q283chen@uwaterloo.ca

Kevin K. H. Cheung
Carleton University, 1125 Colonel By Dr., Ottawa, ON K1S 5B6, Canada
kcheung@math.carleton.ca

P. Michael Kielstra
Harvard University, 1 Oxford Street, Cambridge, MA 02138, USA
pmkielstra@college.harvard.edu

Avery D. Winn
University of Michigan, 530 S State St., Ann Arbor, MI 48109, USA
arwinn@umich.edu

Abstract

In 2016, Chandrasekaran, Végh, and Vempala (*Mathematics of Operations Research*, 41(1):23–48) published a method to solve the minimum-cost perfect matching problem on an arbitrary graph by solving a strictly polynomial number of linear programs. However, their method requires a strong uniqueness condition, which they imposed by using perturbations of the form $c(i) = c_0(i) + 2^{-i}$. On large graphs (roughly $m > 100$), these perturbations lead to cost values that exceed the precision of floating-point formats used by typical linear programming solvers for numerical calculations. We demonstrate, by a sequence of counterexamples, that perturbations are required for the algorithm to work, motivating our formulation of a general method that arrives at the same solution to the problem as Chandrasekaran *et al.* but overcomes the limitations described above by solving multiple linear programs without using perturbations. The key ingredient of our method is an adaptation of an algorithm for lexicographic linear goal programming due to Ignizio (*Journal of the Operational Research Society*, 36(6):507–515, 1985). We then give an explicit algorithm that exploits our method, and show that this new algorithm still runs in strongly polynomial time.

Digital Object Identifier 10.5802/ojmo.2

Keywords perfect matching, uniqueness, perturbation, lexicographic linear goal programming, cutting-plane.

1 Introduction

Given a graph $G = (V, E)$ with edge cost function c, the minimum-cost (or minimum-weight) perfect matching problem is to find a perfect matching $E' \subseteq E$ (a subset such that every vertex $v \in V$ is covered by exactly one $uv \in E'$) so that the sum of the costs of E' is minimized. As mentioned in [4], the minimum-cost perfect matching problem is a classical problem in combinatorial optimization with numerous and varied applications.

Since Edmonds [8] introduced the blossom algorithm (a polynomial-time combinatorial method of solving the problem), a number of efficient implementations have been developed over the years, with Kolmogorov's Blossom V [14] being a recent notable version.

The problem can also be formulated as a binary integer program:

$$\begin{aligned} \min \sum_{e \in E} c(e)x(e) & \\ \text{s.t.} \sum_{uv \in E} x(uv) = 1 & \qquad \forall\, v \in V \\ x(e) \in \{0, 1\} & \qquad \forall\, e \in E. \end{aligned}$$

To use linear programming (LP) techniques to solve the problem, the constraints $x(e) \in \{0, 1\}$ are first relaxed to $x(e) \in [0, 1]$ and then to $x(e) \geq 0$ since the upper bounds are then implied. The linear program that results turns out to be exact for bipartite graphs in the sense that a basic optimal solution is the incidence vector of a minimum-weight perfect matching. Edmonds [7] provides an LP formulation for non-bipartite graphs that has the same property. It requires the addition of "blossom inequalities":

$$\sum_{\substack{uv\in E\\ u\in S, v\notin S}} x(uv) \geq 1 \quad \forall\, S \subseteq V,\ |S| \text{ odd},\ 3 \leq |S| \leq |V| - 3$$

Unfortunately, the presence of an exponential number of constraints in this formulation precludes polynomial-time solvability via a generic LP solver. As a result, researchers in the past have experimented with a cutting-plane approach, solving the relaxation first without the blossom inequalities, then iteratively finding and adding violated inequalities until the problem has an integral solution. A polynomial-time (though impractical) algorithm follows using the equivalence of separation and optimization via the ellipsoid method (see Grötschel *et al.* [10]) and the polynomial-time identification of violated blossom inequalities due to Padberg and Rao [15]. The existence of a practical LP-based cutting plane method for the minimum-weight perfect matching remained uncertain until 2016, when Chandrasekaran *et al.* [2] gave a cutting-plane algorithm which uses only a polynomial number of linear programs.

Their approach involves carefully selecting the blossom inequalities to be included and dropping ones that are not helpful at each iteration and requires that the optimal solution to the linear program be unique. As this uniqueness property does not always hold in general, their method introduces an edge ordering and a perturbation on the edge costs. (The edge costs are assumed to be integers.) In particular, if $c_0(i)$ is the original cost for the i-th edge, then the perturbed cost is $c(i) = c_0(i) + 2^{-i}$. Such a perturbation turns out to be sufficient for providing the required uniqueness property. Even though the increase in size in representing the perturbed costs is polynomial, when the graph is large (say with hundreds of edges), the precision required to represent the perturbed costs exceeds what is typical of the floating-point formats used by most LP solvers [6]. (For example, $4 + 2^{-100} = \frac{5070602400912917605986812821505}{1267650600228229401496703205376}$ requires a mantissa of over 100 bits.)

To overcome the potential numerical difficulties caused by perturbation, we present a variant of the algorithm which does not require an explicit perturbation to ensure uniqueness. We make the observation that by leaving the perturbation unspecified yet sufficiently small, the optimization problem is a lexicographic linear goal program. The algorithm is an adaptation of the sequential method for lexicographic linear goal programming due to Ignizio [12]. It works by solving a sequence of linear programs for each single linear program that the original algorithm would solve. We show that, given the solutions to these programs, we can derive the optimal solution to a hypothetical perturbed linear program without any explicit calculations on perturbed costs. After this, the rest of the proof follows just as it did for the original algorithm.

The trade-off is that our algorithm has a worse runtime than that of Chandrasekaran *et al.* Theirs requires solving $O(n \log n)$ linear programs, while ours solves $O(mn \log n)$. This is, however, still polynomial. Whether or not our method is preferable in practice to implementing the algorithm of Chandrasekaran *et al.* using a high- or arbitrary-precision solver is a question for future research.

The rest of this paper is organized as follows. After defining some terms (Section 2) and summarizing the algorithm from [2] (Section 3), we give examples of graphs which show that this algorithm requires some form of perturbation in both the primal and dual problems. In particular, without perturbing the edge costs, we cannot guarantee that the intermediate solutions will always be half-integral (Section 4) or that the algorithm will terminate (Section 5). This occurs even if we force the primal solution to be the same as it would have been with perturbations. This motivates our new method, which uses multiple linear programs to accurately emulate the perturbations. We first explain this in a general case and point out its connection with lexicographic linear goal programming (Section 6) and then apply it to the specific problem of finding perfect matchings (Section 7).

2 Notation and definitions

The set of $m \times n$ matrices with real entries is denoted by $\mathbb{R}^{m\times n}$. For a matrix $A \in \mathbb{R}^{m\times n}$, $A_{i,j}$ denotes the (i,j)-entry of A; that is, the entry of A at the intersection of the i-th row and the j-th column. $A_{:,j}$ denotes the j-th column of A and $A_{i,:}$ the i-th row. The transpose of A is denoted by A^T.

Following common usage in combinatorics, for a finite set E, let $\mathbb{R}^E$ denote the set of tuples of real numbers indexed by elements of E. For $y \in \mathbb{R}^E$, let $y(i)$ denote the entry indexed by $i \in E$. For a positive integer n, we use $\mathbb{R}^n$ an abbreviation for $\mathbb{R}^{\{1,\ldots,n\}}$. Depending on the context, elements of $\mathbb{R}^n$ are treated as if they were elements of $\mathbb{R}^{n\times 1}$.

We assume familiarity with basic terminology related to matchings and linear programming. A refresher of the former can be found in [5, Chapter 5], and of the latter in [16]. We next recall some definitions in Chandrasekaran *et al.* [2] to facilitate discussion of their minimum-cost perfect matching algorithm.

Let $G = (V, E)$ be a simple undirected graph with integer edge costs given by $c \in \mathbb{Z}^E$. A family $\mathscr{F}$ of subsets of V is said to be *laminar* if for all $U, W \in \mathscr{F}$, $U \cap W = \emptyset$ or $U \subseteq W$ or $W \subseteq U$. For a set $S \subseteq V$, $\delta(S)$ denotes the set of edges incident to one vertex in S and one vertex not in S. For a vertex u, $\delta(u)$ denotes $\delta(\{u\})$. For $x \in \mathbb{R}^E$ and $T \subseteq E$, $x(T)$ denotes the sum $\sum_{e \in T} x(e)$.

Let M be a matching of G. Let $U \subseteq V$, and let $\mathscr{F}$ be a laminar family of subsets of V. Then M is a $(U, \mathscr{F})$-*perfect-matching* if $|\delta(S) \cap M| \leq 1$ for every $S \in \mathscr{F}$ and M covers exactly the vertex set U. A set of vertices $S \in \mathscr{F}$ is said to be $(G, \mathscr{F})$*-factor-critical* for G if, for every $u \in S$, there exists an $(S \setminus \{u\}, \mathscr{F})$-perfect-matching using the edges of G.

For a laminar family $\mathscr{F}$ of odd subsets of V, define the following primal-dual pair of linear programming problems:

$$\begin{aligned} \min \sum_{uv \in E} & c(uv)x(uv) && && (P_{\mathscr{F}}(G,c)) \\ \text{s.t. } x(\delta(u)) &= 1 && \forall\, u \in V \\ x(\delta(S)) &\geq 1 && \forall\, S \in \mathscr{F} \\ x &\geq 0, \end{aligned}$$

$$\begin{aligned} \max \sum_{S \in V \cup \mathscr{F}} & \Pi(S) && && (D_{\mathscr{F}}(G,c)) \\ \text{s.t.} \sum_{S \in V \cup \mathscr{F} : uv \in \delta(S)} \Pi(S) &\leq c(uv) && \forall\, uv \in E \\ \Pi(S) &\geq 0 && \forall\, S \in \mathscr{F}. \end{aligned}$$

Let Π be a feasible solution to $D_{\mathscr{F}}(G,c)$. G_Π denotes the graph (V, E_Π) where

$$E_\Pi = \left\{ uv \in E : \sum_{S \in V \cup \mathscr{F} : uv \in \delta(S)} \Pi(S) = c(uv) \right\}.$$

Colloquially, E_Π is the set of “tight” edges with respect to Π. We say that Π is an $\mathscr{F}$*-critical dual* if every $S \in \mathscr{F}$ is $(G_\Pi, \mathscr{F})$-factor-critical and $\Pi(T) > 0$ for every non-maximal $T \in \mathscr{F}$. If Π is an $\mathscr{F}$-critical dual except that some sets $S \in \mathscr{F}$ for which $\Pi(S) = 0$ may not be $(G_\Pi, \mathscr{F})$-factor-critical, we say that Π is an $\mathscr{F}$-positively-critical dual.

Finally, we define a metric on solutions to $D_{\mathscr{F}}(G,c)$

$$\Delta(\Gamma, \Pi) = \sum_{S \in V \cup \mathscr{F}} \frac{1}{|S|} |\Gamma(S) - \Pi(S)|.$$

It can be easily verified that this has the properties of a metric.

For a given fixed Γ, we say that Π is Γ*-extremal* if it minimizes $\Delta(\Gamma, \Pi)$. Given Γ and a primal optimal solution x, we may find a Γ-extremal dual optimal solution by solving the following linear program [2, Section 5]:

$$\begin{aligned} \min \sum_{S \in V \cup \mathscr{F}} & \frac{1}{|S|} r(S) && && (D^*_{\mathscr{F}}(G,c)) \\ \text{s.t. } r(S) + \Pi(S) &\geq \Gamma(S) && \forall\, S \in V \cup \mathscr{F}_x \\ -r(S) + \Pi(S) &\leq \Gamma(S) && \forall\, S \in V \cup \mathscr{F}_x \\ \sum_{uv \in \delta(S)} \Pi(S) &= c(uv) && \forall\, uv \in \operatorname{supp}(x) \\ \sum_{uv \in \delta(S)} \Pi(S) &\leq c(uv) && \forall\, uv \notin \operatorname{supp}(x) \\ \Pi(S) &\geq 0 && \forall\, S \in \mathscr{F}_x \\ \Pi(S) &= 0 && \forall\, S \in \mathscr{F} \setminus \mathscr{F}_x \\ r(S) &= 0 && \forall\, S \in \mathscr{F} \setminus \mathscr{F}_x, \end{aligned}$$

where $\mathscr{F}_x = \{S \in \mathscr{F} : x(\delta(S)) = 1\}$. The solution will give us values for r and Π; we ignore r and take Π to be our Γ-extremal solution.

3 The Chandrasekaran-Végh-Vempala algorithm

Algorithm 1 for finding a minimum-cost perfect matching on G is due to Chandrasekaran *et al.* [2]. It assumes, as we will from now on, that the edge costs are integers.

Algorithm 1: C-P-Matching Algorithm

```
    Input: A graph $G = (V, E)$ with edge costs $c \in \mathbb{Z}^E$.
    Output: A binary vector $x$ representing a minimum-cost perfect matching on $G$.
 1  Let $c$ be the cost function on the edges after perturbation (i.e., after ordering the edges arbitrarily and
      increasing the cost of each edge $i$ by $2^{-i}$).
 2  $\mathscr{F} \leftarrow \emptyset$, $\Gamma \leftarrow 0$
 3  repeat
 4      Find an optimal solution $x$ to $P_{\mathscr{F}}(G, c)$.
 5      if $x$ is integral then
 6          return $x$
 7      Find a $\Gamma$-extremal dual optimal solution $\Pi$ to $D_{\mathscr{F}}(G, c)$ (possibly by solving $D^*_{\mathscr{F}}(G, c)$).
 8      $\mathscr{H}' \leftarrow \{S \in \mathscr{F} : \Pi(S) > 0\}$
 9      Let $\mathscr{C}$ denote the set of odd cycles in supp($x$). For each $C \in \mathscr{C}$, define $\hat{C}$ as the union of $V(C)$ and
          the maximal sets of $\mathscr{H}'$ intersecting it.
10      $\mathscr{H}'' \leftarrow \{\hat{C} : C \in \mathscr{C}\}$
11      $\mathscr{F} \leftarrow \mathscr{H}' \cup \mathscr{H}''$, $\Gamma \leftarrow \Pi$
12  end
```

The authors of the algorithm showed that $\mathscr{F}$ is always a laminar family and that the algorithm terminates after $O(n \log n)$ iterations, assuming that $P_{\mathscr{F}}(G, c)$ has a unique optimal solution in every iteration of the algorithm. This is ensured through the use of perturbations in the first step. The authors further demonstrate that a Γ-extremal dual solution, with an $\mathscr{F}$-critical Γ, is an $\mathscr{F}$-positively-critical dual optimal to $D_{\mathscr{F}}(G, c)$, so the result of step 7 is $\mathscr{F}$-positively-critical. When combined with the uniquness assumption, this leads to x being half-integral in each iteration.

The choice of using powers of $\frac{1}{2}$ for the perturbations is to keep the increases in input size polynomial. However, to guarantee uniqueness, powers of a sufficiently small $\epsilon > 0$ can be used instead.

▶ **Lemma 1.** *There exists a $\delta > 0$ such that the perturbations used in Algorithm 1 may be replaced with powers of ϵ for any $\delta > \epsilon > 0$.*

Proof. Consider the proof given for the efficacy of the 2^{-i} perturbation in [2, Section 7]. This uses only one property of the perturbation: that, if $\sum_{i=1}^m a(i)2^{-i} = \sum_{k=1}^n b(k)2^{-k}$, with $a(i), b(k) > 0$, then $m = n$ and $a(i) = b(i)$ for all i. We prove this for a class of arbitrary $\epsilon > 0$, after which the desired result follows.

Assume $\sum_{i=1}^m a(i)\epsilon^i = \sum_{k=1}^n b(k)\epsilon^k$. Assume further, without loss of generality, that $m \leq n$. Then $\sum_{i=1}^m (a(i) - b(i))\epsilon^i - \sum_{k=m+1}^n b(k)\epsilon^k = 0$.

Take $m < n$. For ϵ sufficiently small, either $a(i) = b(i)$ for all $i \in \{1, \ldots, m\}$ or $\sum_{i=1}^m |(a(i) - b(i))|\epsilon^i > \sum_{k=m+1}^n b(k)\epsilon^k$. In the first case, $\sum_{k=m+1}^n b(k)\epsilon^k = 0$, a contradiction since ϵ and all $b(k)$ are positive; in the second, $\sum_{i=1}^m (a(i) - b(i))\epsilon^i - \sum_{k=m+1}^n b(k)\epsilon^k \neq 0$. Therefore $m = n$.

Assume there exists a minimal l such that $a(l) - b(l) \neq 0$. Then

$$0 = \sum_{i=1}^m (a(i) - b(i))\epsilon^i = (a(l) - b(l))\epsilon^l + \epsilon^{l+1} \sum_{i=l+1}^n (a(i) - b(i))\epsilon^{i-l-1}.$$

For sufficiently small ϵ, $|(a(l) - b(l))\epsilon^l| > |\epsilon^{l+1} \sum_{i=l+1}^n (a(i) - b(i))\epsilon^{i-l-1}|$, so $(a(l) - b(l))\epsilon^l + \epsilon^{l+1} \sum_{i=l+1}^n (a(i) - b(i))\epsilon^{i-l-1} \neq 0$.

This shows that, for any given a and b, there exists a δ such that if $\sum a(i)\delta^i = \sum b(k)\delta^k$, then $a = b$ for all $\delta > \epsilon > 0$. In fact, we need to only consider the cases where a and b are basic feasible solutions to $P_{\mathscr{F}}(G, c)$, because if there exists an optimal solution that is not a basic feasible solution then there exist two distinct basic feasible solutions that are optimal. Therefore, if optimal basic feasible solutions are unique, so are optimal solutions in general.

Fix $\mathscr{F}$. Then, because $P_{\mathscr{F}}(G, c)$ is bounded and finite-dimensional, it has a finite number of basic feasible solutions $s_1, \ldots, s_k$. Every pair (s_p, s_q) gives us a δ by setting $a = s_p$, $b = s_q$ and running through the logic above. Take the smallest of these δs to complete the proof. ◀

For reasons that will become clear later (see Section 6), it will be more convenient to use powers of a sufficiently small $\epsilon > 0$ as perturbations instead of powers of $\frac{1}{2}$. In any case, increasing the bit-length required to represent the edge costs can lead to practical computation challenges since most LP solvers employ fixed-length floating-point formats. (Notable exceptions exist, such as QSopt-Exact [1] and the SoPlex rational solver [9], but they are significantly slower than non-exact solvers.) We feel strongly that the key to a successful implementation of Algorithm 1 using a typical LP solver is to not work with any explicit numerical perturbation. However, it remains to be seen whether or not our method is indeed preferable in practice to Algorithm 1 implemented using an arbitrary-precision LP solver.

An obvious way of modifying the algorithm is simply to not perturb the edge costs and run the rest of the procedure as stated, but this violates the uniqueness assumption, and as easily demonstrated in [2, Section 1], can lead to non-half-integrality and cycling.

Instead, we may emulate perturbations by ordering the edges (as in step 1 of Algorithm 1) and then finding a lexicographically-minimal optimal solution to $P_{\mathscr{F}}(G, c)$, where c is now an unperturbed cost function. This may be accomplished using Algorithm 2, which shows the process in a more general case.

Algorithm 2: Lexicographically-Minimal Primal Algorithm

```
   Input: A linear program $P$ of the form $\min c^\mathsf{T} x$ s.t. $Ax \geq b$, where $x \in \mathbb{R}^n$.
   Output: The lexicographically-minimal solution $x$ to $P$.
1  Solve $P$ and let its opimal value be $\gamma$.
2  $K \leftarrow \emptyset$, $x \leftarrow 0$
3  for $i \leftarrow 1$ to $n$ do
4      Set $x$ to an optimal solution to
           $\min x_i$
           s.t. $c^\mathsf{T} x = \gamma$
                $x_j = z \quad \forall (j, z) \in K$
                $Ax \geq b$.
5      $K \leftarrow K \cup \{(i, x_i)\}$
6  end
7  return $x$
```

By [16, p. 138], the lexicographically-minimal optimal solution to $P_{\mathscr{F}}(G, c)$ is the same as the optimal solution to the perturbed $P_{\mathscr{F}}(G, c)$. Unfortunately, this on its own ensures neither half-integrality nor convergence. Before giving a slightly more complex modification of the algorithm that uses a multi-stage approach to mimic solving with perturbation without actually working with perturbations, we first give some examples of graphs that demonstrate the problems just mentioned.

4 Non-half-integral solution

The following example, which we call the "Dancing Robot," shows that, if the edge costs are not perturbed at all, having an $\mathscr{F}$-critical dual is not sufficient to guarantee that all lexicographically-minimal optimal primal solutions are half-integral. Chandrasekaran *et al.* [2] provide an example early in their paper of a graph on which their algorithm as written does not maintain half-integrality, but this does not entirely suffice for our purposes, as the lexicographically-minimal primal solution on this graph, for any edge ordering, is integral.

The graph shown in Figures 1 to 4, with all edges having cost 1, eventually gives non-half-integral values when run through the original algorithm without any perturbation while enforcing a lexicographically-minimal optimal primal. In these figures, $\mathscr{F}_i$ denotes the $\mathscr{F}$ at the beginning of the loop in the i^{th} iteration while $\mathscr{F}_{i+1}$ is the laminar family after updating $\mathscr{F}_i$ at step 9 in the i^{th} iteration according to Algorithm 1. During each iteration, an optimal dual solution is given by Π, the vector having value $\frac{1}{2}$ on the entries indexed by the vertices and 0 on entries indexed by the sets in $\mathscr{F}$. Note that all edges in the graph are tight with respect to Π. We can

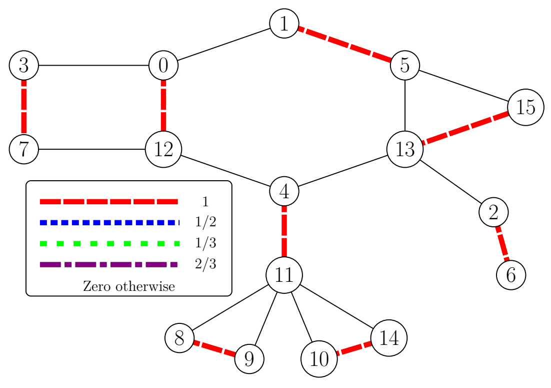
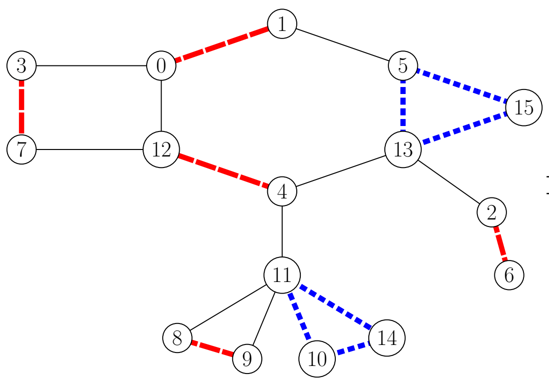


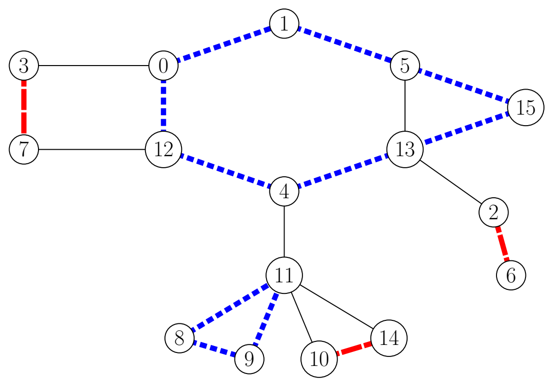
Figure 1 Lexicographically-minimal Perfect Matching



$\Pi_1(v) = \frac{1}{2}\ \forall v \in \mathcal{V},\ \mathscr{F}_1 = \emptyset$

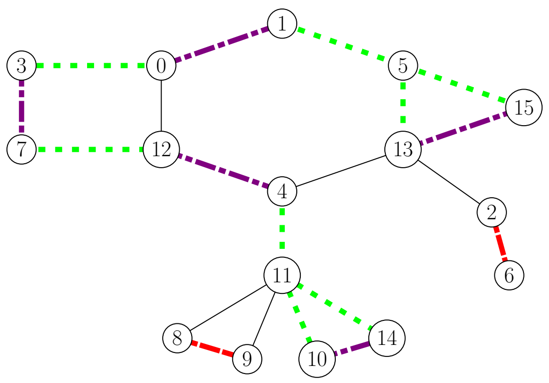
Figure 2 First Iteration

Edge ordering:
(1, 5)
(2, 13)
(10, 14)
(0, 3)
(4, 12)
(5, 13)
(7, 12)
(5, 15)
(3, 7)
(8, 9)
(0, 1)
(11, 14)
(0, 12)
(4, 13)
(2, 6)
(10, 11)
(9, 11)
(4, 11)
(8, 11)
(13, 15)



$\Pi_2(v) = \frac{1}{2}\ \forall v \in \mathcal{V},$
$\mathscr{F}_2 = \{\{5, 15, 13\}, \{10, 11, 14\}\},$
$\Pi_2(S) = 0\ \forall S \in \mathscr{F}_2$

Figure 3 Second Iteration



$\Pi_3(v) = \frac{1}{2}\ \forall v \in \mathcal{V},$
$\mathscr{F}_3 = \{\{0, 1, 5, 15, 13, 4, 12\}, \{8, 11, 9\}\},$
$\Pi_3(S) = 0\ \forall S \in \mathscr{F}_3$

Figure 4 Third Iteration

see that, although the primal solutions in the first and second iterations are half-integral (shown in Figures 2 and 3), the solution in the third iteration is no longer half-integral. The $\frac{1}{3}$- and $\frac{2}{3}$-edges are shown in Figure 4.

Meanwhile, the dual solution Π is a positively-critical optimal dual for the current $\mathscr{F}$ in every iteration, as well as a critical dual for the next $\mathscr{F}$. For instance, the Π from the second iteration, feasible to the dual problems from both the second and third iterations, is trivially an $\mathscr{F}$-positively-critical optimal dual for the second iteration, since none of the sets $S \in \mathscr{F}$ have positive dual value. For the third iteration, since there exists an $(S \setminus \{u\}, \mathscr{F})$-perfect-matching for any node $u \in S \in \mathscr{F}$, and since $\mathscr{F}$ only has maximal sets, that same Π is an $\mathscr{F}$-critical dual.

Even worse, the algorithm will eventually enter into an infinite loop on this example as illustrated in Appendix A. The example in the next section also loops but does not lose half-integrality.

On their own, then, a lexicographically-minimal primal and an $\mathscr{F}$-critical dual can guarantee neither half-integrality nor termination. It is worth mentioning that we could expand the graph to get one with more non-half-integral edges by making the 2-6 edge (the "arm" of the Dancing Robot) overlap with another Dancing Robot's 6-2 edge. This combination would have twice as many non-half-integral edges, spread across twice as many non-half-integral paths, as the original Dancing Robot. By combining multiple copies of the Dancing Robot in such a manner, we can get as many non-half-integral paths as we want.

We can even alter the Dancing Robot in order to give us arbitrarily small but nonzero values in a lexicographically-minimal optimal solution. Say we want a primal solution x such that, for some uv, $x(uv) = \frac{1}{2n+1}$ for some $n \in \mathbb{Z}_{\geq 0}$. We add $2(n-1)$ new edges between 0 and the 0-1 edge, alternately without and with adjoining 4-cycles. The next example shown in Figure 5 illustrates the case $n = 2$.

Simply by following the algorithm through, we see that we will eventually end up with the cut

$$\{4, 12, 0, 16, 17, 1, 5, 15, 13\}$$

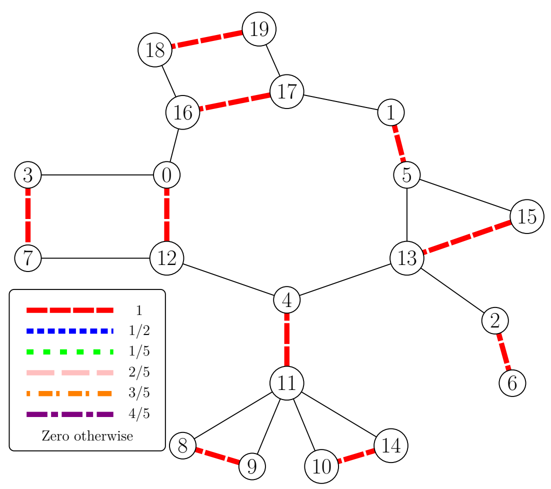


Figure 5 Lexicographically-minimal Perfect Matching

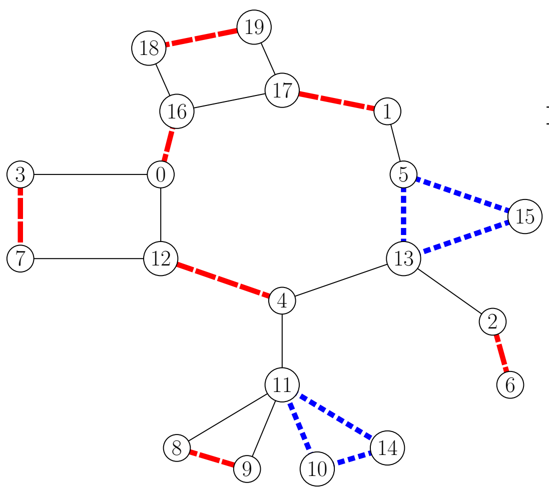


Figure 6 First Iteration

Edge ordering:
(1, 5)
(2, 13)
(10, 14)
(0, 3)
(17, 19)
(4, 12)
(5, 13)
(7, 12)
(16, 18)
(5, 15)
(3, 7)
(18, 19)
(8, 9)
(0, 16)
(1, 17)
(11, 14)
(0, 12)
(16, 17)
(4, 13)
(2, 6)
(10, 11)
(9, 11)
(4, 11)
(8, 11)
(13, 15)

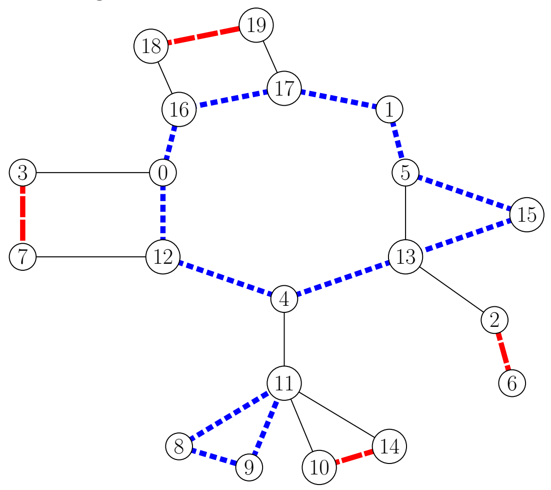


Figure 7 Second Iteration

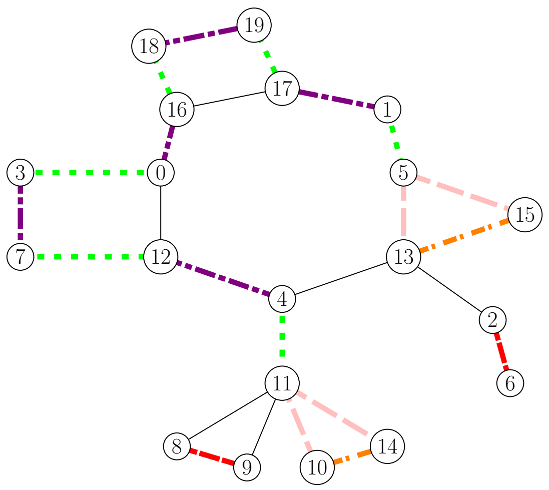


Figure 8 Third Iteration

(see Figure 7) with $2n+1$ edges across it. Furthermore, by the conditions of the matching ($x(\delta(u)) = 1$) and the fact that these edges form a path in the matching, each edge across this cut must have the same value, which we will call ζ. Since $x(\delta(S)) \geq 1$, the minimum cost is when $(2n+1)\zeta = 1$ or $\zeta = \frac{1}{2n+1}$.

5 Cycling example

Even when seeking a lexicographically-minimal optimal solution to $P_{\mathscr{F}}(G, c)$ with half-integrality maintained throughout, cycling can still occur in the absence of perturbation. The graph in Figures 9 and 10 (which is easily seen to have a perfect matching), with each edge having cost 1, exhibits such behavior. At all times, an optimal $\mathscr{F}$-positively critical dual is given by a vector with the vertices having value $\frac{1}{2}$ and the odd sets in $\mathscr{F}$ having value 0. Since Algorithm 1 only retains cuts which have nonzero values in the dual (step 8), no cuts are preserved between iterations. Thus, the blossom inequalities which were violated in the previous iteration are once again allowed to be violated in the next iteration, leading to cycling.

In this example, in every iteration, there is nothing forcing the edge $(2, 7)$ to have any matching value other than 0. Therefore, to satisfy lexicographical minimality, that edge is indeed left out of the matching, forcing the two bow-tie arrangements on each end of the edge to form their own half-integral matching as best as they can. This is of course impossible since each contains five vertices, but the only thing the algorithm sees as going wrong is the formation of a half-integral triangle. There are two ways to form this triangle, each satisfying the constraint derived from the other, so the algorithm flip-flops between them forever.

This precludes us from ensuring termination of an approach using a lexicographically-minimal primal with an unperturbed dual. More significantly, it also means that we could not even implement a heuristic version

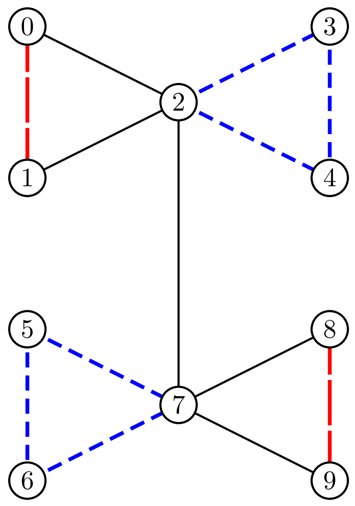


Figure 9 Odd iterations

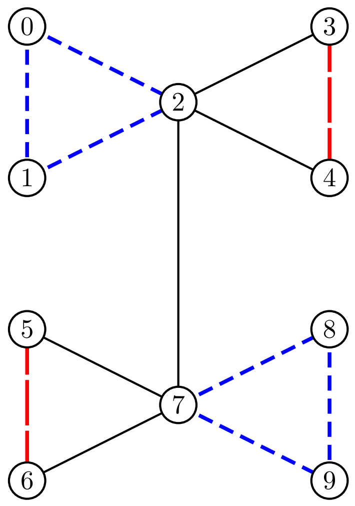


Figure 10 Even iterations

Edge ordering:
(2, 7)
(7, 9)
(7, 8)
(8, 9)
(6, 7)
(5, 7)
(5, 6)
(1, 2)
(0, 2)
(0, 1)
(3, 4)
(2, 4)
(2, 3)

which, in the event of cycling, would restart the algorithm with randomized edge orderings. Were half-integrality to occur in tandem with cycling, as it does in Section 4, we could simply verify half-integrality at each iteration, and, in rare cases of non-half-integrality, begin the entire algorithm again with a different edge ordering, giving us a good average-case runtime.[1] However, this graph shows that when a possibility of cycling exists, it is likely undetectable by any means other than direct comparison between iterations. This forces us to adopt an algorithm which simulates perturbations in the dual.

6 Handling perturbed costs through lexicographic linear goal programming

Chandrasekaran *et al.* [2] chose a specific perturbation of the costs, namely, adding 2^{-i} on each edge i. In general, perturbation in linear programming (usually for the purpose of eliminating degeneracy, as in [3]) is of the form ϵ^i where ϵ is sufficiently small. In theoretical analysis, ϵ is simply left unspecified.

Using a technique that is widely known in the goal programming community, we show in this section how we could obtain optimal solutions to both the primal and dual problems with perturbed costs, working with the fact that ϵ is sufficiently small yet not given exactly. Our method thus avoids working directly with cost values that exceed the representation capacity of fixed-length floating-point formats typically used by LP solvers. It is an adaptation of a method due to Ignizio [12] for lexicographic linear goal programming which computes a multidimensional dual solution, a critical piece in the modification of Algorithm 1. In this section, we briefly discuss the ideas in general terms and we specialize it to Algorithm 1 in the next section.

Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and $c_0, \ldots, c_k \in \mathbb{R}^n$ for some nonnegative integer k. Let $N \subseteq \{1, \ldots, n\}$. Let $F = \{1, \ldots, n\} \setminus N$. Define c_ϵ as $\sum_{p=0}^{k} c_p \epsilon^p$ where $\epsilon \geq 0$.

Consider the linear programming problem:

$$\begin{aligned} \min\ & c_\epsilon^{\mathsf{T}} x && && (P(\epsilon)) \\ \text{s.t. } & Ax \geq b \\ & x_j \geq 0 && \forall\, j \in N. \end{aligned}$$

Its dual is

$$\begin{aligned} \max\ & b^{\mathsf{T}} y && && (D(\epsilon)) \\ \text{s.t. } & A_{:,j}^{\mathsf{T}} y \leq c_\epsilon(j) && \forall\, j \in N \\ & A_{:,j}^{\mathsf{T}} y = c_\epsilon(j) && \forall\, j \in F \\ & y \geq 0. \end{aligned}$$

[1] We discovered the Dancing Robot after searching through over two thousand randomly-generated graphs, all of which were rapidly and correctly solved by the use of a lexicographically-minimal primal and unperturbed dual. Furthermore, if a random edge ordering is applied to the Dancing Robot for example, it is very likely that a perfect matching will be found without issue.

Readers familiar with linear goal programming (see [11] for an introduction) would recognize the above primal-dual pair as a representation of the following lexicograhic linear goal program

$$\begin{aligned} &\text{lexmin } [c_0^\mathsf{T} x, \ldots, c_k^\mathsf{T} x] \\ &\text{s.t. } Ax \geq b \\ &\qquad x_j \geq 0 \qquad\qquad \forall\, j \in N. \end{aligned}$$

and its multidimensional dual ([12]):

$$\begin{aligned} &\text{lexmax } b^\mathsf{T} Y \\ \text{s.t. } & A_{:,j}^\mathsf{T} Y_{:,p} \leq c_p(j) && \forall\, p \in \{0,1,\ldots,k\},\ j \in N \\ & A_{:,j}^\mathsf{T} Y_{:,p} = c_p(j) && \forall\, p \in \{0,1,\ldots,k\},\ j \in F \\ & Y_{i,:} \overset{\text{lex}}{\geq} 0 && \forall\, i \in \{1,\ldots,m\} \\ & Y \in \mathbb{R}^{m \times (k+1)} \end{aligned}$$

The correspondence between y and Y is given by $y = Y\zeta$ where $\zeta = \begin{pmatrix} 1 & \epsilon & \cdots & \epsilon^k \end{pmatrix}^\mathsf{T}$, with the understanding that ϵ is unspecified yet sufficiently small.

(Here, the notation $Y_{i,:} \overset{\text{lex}}{\geq} 0$ denotes that if s is the least index such that $Y_{i,s} \neq 0$, then in fact $Y_{i,s} > 0$. For example, it is true that $\begin{pmatrix} 0 & 1 & -5 \end{pmatrix} \overset{\text{lex}}{\geq} 0$ but it is not true that $\begin{pmatrix} 0 & -5 & 1 \end{pmatrix} \overset{\text{lex}}{\geq} 0$. Note that when $k = 1$, $Y_{i,:} \overset{\text{lex}}{\geq} 0$ is simply a nonnegativity constraint.)

To keep this paper self-contained and maintain the flow of exposition, we will not go into the topic of linear goal programming in any detail. Instead, we adapt the method in [12] to the current pertubation setting and give proofs to all the necessary results, even though none of the results can be regarded as new. Readers interested in learning more about lexicographic linear goal programming are referred to [11] and the references therein.

We claim that Algorithm 3 solves the primal-dual pair $P(\epsilon)$ and $D(\epsilon)$.

Algorithm 3: Algorithm for perturbed LP primal-dual pair

```
Input: $P(\epsilon)$ with $\epsilon > 0$ sufficiently small.
Output: An optimal $x'$ to $P(\epsilon)$ and an optimal $y'$ to $D(\epsilon)$.
1  $E \leftarrow \emptyset$, $J \leftarrow \emptyset$
2  for $p \leftarrow 0$ to $k$ do
3      $\overline{J} \leftarrow N \setminus J$
4      Set $x_p$ to be an optimal solution to
           $\min \sum_{j \in \overline{J}} c_p(j) x(j)$
           s.t. $\sum_{j \in \overline{J}} A_{i,j} x(j) \geq b(i) \quad \forall\, i \notin E$
                $\sum_{j \in \overline{J}} A_{i,j} x(j) = b(i) \quad \forall\, i \in E$
                $x(j) \geq 0 \quad \forall\, j \in \overline{J}$
        and $y_p$ to an optimal solution to its dual.
5      $E \leftarrow E \cup \{i : y_p(i) > 0\}$
6      $J \leftarrow J \cup \{j : A_{:,j}^\mathsf{T} y_p < c_p(j)\}$
7  end
8  Form $x' \in \mathbb{R}^n$ such that $x'(j) = x_k(j)$ for all $j \notin J$ and $x'(j) = 0$ for all $j \in J$.
9  $y' \leftarrow \sum_{p=0}^{k} \epsilon^p y_p$
10 return $x', y'$
```

The correctness of Algorithm 3 follows from Lemma 2 below. Before we give the proof, we illustrate the algorithm with an example. For each p, let M_p denote the LP problem in step 4 of the algorithm. Consider $P(\epsilon)$ with

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad c_0 = \begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix}, \qquad c_1 = \begin{pmatrix} 4 \\ 2 \\ 0 \end{pmatrix}, \qquad c_2 = \begin{pmatrix} -2 \\ -1 \\ 1 \end{pmatrix}, \qquad N = \{1, 2\}.$$

The dual problem is

$$\begin{array}{rcccccl} \max & y(1) & + & y(2) & & & \\ \text{s.t.} & y(1) & & & \leq & 1 + 4\epsilon - 2\epsilon^2 \\ & & & y(2) & \leq & 1 + 2\epsilon - \epsilon^2 \\ & y(1) & + & 2y(2) & = & 3 + \epsilon^2 \\ & y(1) & , & y(2) & \geq & 0. \end{array}$$

Note that $x_0 = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}^\mathsf{T}$ and $y_0 = \begin{pmatrix} 1 & 1 \end{pmatrix}^\mathsf{T}$ are optimal solutions to $P(0)$ and $D(0)$, respectively, which are in turn equivalent to M_0 and its dual.

Since $y_0(1), y_0(2) > 0$ and all the constraints in $D(0)$ are satisfied with equality at y_0, the problem M_1 is

$$\begin{array}{rcccccccl} \min & 4x(1) & + & 2x(2) & & & & & \\ \text{s.t.} & x(1) & & & + & x(3) & = & 1 \\ & & & x(2) & + & 2x(3) & = & 1 \\ & x(1) & , & x(2) & & & \geq & 0. \end{array}$$

The dual of M_1 is

$$\begin{array}{rcccccl} \max & y(1) & + & y(2) & & \\ \text{s.t.} & y(1) & & & \leq & 4 \\ & & & y(2) & \leq & 2 \\ & y(1) & + & 2y(2) & = & 0. \end{array}$$

An optimal solution to M_1 is $x_1 = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}^\mathsf{T}$. An optimal dual solution is $y_1 = \begin{pmatrix} 4 & -2 \end{pmatrix}^\mathsf{T}$. The second constraint in the dual is not active at y_1. Hence, M_2 is

$$\begin{array}{rcccccl} \min & -2x(1) & + & x(3) & & \\ \text{s.t.} & x(1) & + & x(3) & = & 1 \\ & & & 2x(3) & = & 1 \\ & x(1) & & & \geq & 0. \end{array}$$

The dual of M_2 is

$$\begin{array}{rcccccl} \max & y(1) & + & y(2) & & \\ \text{s.t.} & y(1) & & & \leq & -2 \\ & y(1) & + & 2y(2) & = & 1. \end{array}$$

An optimal solution to M_2 is $\begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}^\mathsf{T}$. An optimal dual solution is $y_2 = \begin{pmatrix} -2 & \frac{3}{2} \end{pmatrix}^\mathsf{T}$.

Setting

$$y' = y_0 + \epsilon y_1 + \epsilon^2 y_2 = \begin{pmatrix} 1 + 4\epsilon - 2\epsilon^2 \\ 1 - 2\epsilon + \frac{3}{2}\epsilon^2 \end{pmatrix},$$

we have that y' is a feasible solution to $D(\epsilon)$ and satisfies complementary slackness with $x' = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}^\mathsf{T}$ for the primal-dual pair $P(\epsilon)$ and $D(\epsilon)$ for a sufficiently small $\epsilon > 0$.

▶ **Lemma 2.** *Let M_p denote the LP problem solved in step 4 of Algorithm 3.*

1. *For every $p \in \{1, \ldots, k\}$, x_p is an optimal solution to $M_0, \ldots, M_{p-1}$.*
2. *x' and y' are feasible to $P(\epsilon)$ and $D(\epsilon)$, respectively, and satisfy complementary slackness.*

Proof. For each $j = 1, \dots, p$, M_j is obtained from M_{j-1} by adding constraints to enforce complementary slackness with y_{j-1}. Removing $x(j)$ can be viewed as adding the constraint $x(j) = 0$. It follows that x_p is feasible to M_j and satisfies complementary slackness with y_j for all $j \in \{0, \dots, p-1\}$.

To prove the second part, we start by noting that x' is feasible to $P(\epsilon)$. Let E_p, J_p, and $\overline{J_p}$ be the sets E, J, and $\overline{J}$ referred to in M_p. The dual of M_p is

$$\begin{aligned} &\max\ b^{\mathsf{T}} y \\ \text{s.t. } & A_{:,j}^{\mathsf{T}} y \le c_p(j) && \forall\ j \in \overline{J_p} \\ & A_{:,j}^{\mathsf{T}} y = c_p(j) && \forall\ j \in F \\ & y(i) \ge 0 && \forall\ i \notin E_p. \end{aligned}$$

Clearly, $A_{:,j}^{\mathsf{T}} y' = c_\epsilon(j)$ for all $j \in F$. Next, we show that $A_{:,j}^{\mathsf{T}} y' \le c_\epsilon(j)$ for all $j \in N$.

Suppose that $j \in \overline{J_k}$. Since $\overline{J_p} \subseteq \overline{J_{p-1}}$ for $p = 1, \dots, k$, we have $A_{:,j}^{\mathsf{T}} y_p \le c_p(j)$. Thus,

$$A_{:,j}^{\mathsf{T}} y' = \sum_{p=0}^{k} \epsilon^p A_{:,j}^{\mathsf{T}} y_p \le \sum_{p=0}^{k} \epsilon^p c_p(j) = c_\epsilon(j).$$

Now, suppose that $j \in J_k$. Then, there exists $r < n$ such that $A_{:,j}^{\mathsf{T}} y_r < c_r(j)$. Let $s_i = c_i(j) - A_{:,j}^{\mathsf{T}} y_i$ for $i = 1, \dots, m$. Thus,

$$\begin{aligned} c_\epsilon(j) - A_{:,j}^{\mathsf{T}} y' &= \sum_{p=0}^{k} \epsilon^p s_p \\ &= \sum_{p=0}^{r} \epsilon^p s_p + \sum_{p=r+1}^{k} \epsilon^p s_p \\ &\ge \epsilon^r \left(s_r + \sum_{p=r+1}^{k} \epsilon^{p-r} s_p \right) \\ &= \epsilon^r \left(s_r + \epsilon \sum_{q=0}^{k-r-1} \epsilon^q s_{q+r+1} \right) \\ &> 0 \end{aligned}$$

for $\epsilon > 0$ sufficiently small.

We now show that $y' \ge 0$. Consider $y'(j)$ for some $j \in \{1, \dots, m\}$. If $j \notin E_k$, then $y_p(j) \ge 0$ for $p = 0, \dots, k$, implying that $y'(j) \ge 0$. Otherwise, $j \in E_r$ for some $r \in \{1, \dots, k\}$. Choose r as small as possible. We must have $y_{r-1}(j) > 0$. Then,

$$\begin{aligned} y'(j) &= \sum_{p=0}^{k} \epsilon^p y_p(j) \\ &\ge \sum_{p=r}^{k} \epsilon^p y_p(j) \\ &= \epsilon^r \left(y_r(j) + \epsilon \sum_{p=0}^{k-r-1} \epsilon^p y_{p+r+1}(j) \right) \\ &> 0 \end{aligned}$$

for $\epsilon > 0$ sufficiently small.

Finally, to see that x' and y' satisfy complementary slackness, note that, by part 1, if $x'(j) > 0$, then $A_{:,j}^{\mathsf{T}} y_p = c_p(j)$ for all $p \in \{0, \dots, k\}$. Thus $A_{:,j}^{\mathsf{T}} y' = c_\epsilon(j)$. Furthermore, if $A_{i,:} x' < b(i)$ for some i, $y_p(i) = 0$ for all $p \in \{0, \dots, k\}$. This implies that $y'(i) = 0$. ◀

We now make two observations that will be useful in the next section. First, we can see (as observed by Ignizio [12] in his setting) from the proof of Lemma 2 that the dual of M_p can be obtained directly from the dual of M_{p-1} and an associated optimal solution y_{p-1} by removing constraints (including the nonnegativity bound constraints) that are not active at y_{p-1}. It follows that one can work exclusively with the duals of $M_0, \ldots, M_k$ if one is only interested in obtaining an optimal solution to $D(\epsilon)$. Moreover, in practice, y' is represented by the list $y_0, \ldots, y_k$ without any reference to a particular value of ϵ. Then, to determine if $y'(i) \neq 0$ for some i, we can simply check if there exists a p such that $y_p(i) \neq 0$, since, for sufficiently small ϵ, $y'(i) = 0$ if and only if $y_p(i) = 0$ for all p.

7 Modified Chandrasekaran-Végh-Vempala algorithm

We now modify Algorithm 1 to circumvent the need to utilize an explicit perturbation of the edge costs. First, we arbitrarily order the edges and increase the cost of each edge i by ϵ^i for some sufficiently small $\epsilon > 0$ that will remain unspecified. By Lemma 1, we may assume that with such a perturbation, Algorithm 1 will still return a minimum-cost perfect matching. In Algorithm 1, step 4 and step 7 involve solving $P_{\mathscr{F}}(G,c)$ and $D^*_{\mathscr{F}}(G,c)$ respectively with perturbed data. We emulate perturbations in the first of these by finding a lexicographically-minimal optimal solution. The other is handled by applying the method developed in the previous section to $D^*_{\mathscr{F}}(G,c)$. Unfortunately, the formulation $D^*_{\mathscr{F}}(G,c)$ as stated is not in the form discussed in the previous section. Therefore, we write $D^*_{\mathscr{F}}(G,c)$ as the following equivalent LP:

$$\begin{aligned}
\max \sum_{S \in V \cup \mathscr{F}_x} & -\frac{1}{|S|} r(S) \\
\text{s.t. } -r(S) - \Pi(S) &\leq -\Gamma(S) && \forall\, S \in V \cup \mathscr{F}_x \\
-r(S) + \Pi(S) &\leq \Gamma(S) && \forall\, S \in V \cup \mathscr{F}_x \\
\sum_{S \in V \cup \mathscr{F}_x : uv \in \delta(S)} \Pi(S) &= c(uv) && \forall\, uv \in \operatorname{supp}(x) \\
\sum_{S \in V \cup \mathscr{F}_x : uv \in \delta(S)} \Pi(S) &\leq c(uv) && \forall\, uv \in E \setminus \operatorname{supp}(x) \\
\Pi(S) &\geq 0 && \forall\, S \in \mathscr{F}_x, \\
r(S) &\geq 0 && \forall\, S \in V \cup \mathscr{F}_x
\end{aligned}$$

where $\mathscr{F}_x = \{S \in \mathscr{F} : x(\delta(S)) = 1\}$. Note that we have introduced the redundant bound constraints $r(S) \geq 0 \ \forall\, S \in V \cup \mathscr{F}_x$, which are implied by the first two sets of constraints, to make all variables in the formulation nonnegative.

With explicit perturbation of the edge costs, Γ and c will be polynomials in ϵ. Intuitively, we define Γ_i and c_i to be the coefficients of ϵ^i in Γ and c; we will define these rigorously in a moment.

The reason for writing $D^*_{\mathscr{F}}(G,c)$ as above serves no purpose other than to make it plain that it can be viewed as the dual problem $D(\epsilon)$ of some $P(\epsilon)$ with cost values given by polynomials in ϵ. However, in an actual algorithm as seen below, we can work directly with $D^*_{\mathscr{F}}(G,c)$ as originally written. With these changes and the following definitions, we obtain Algorithm 4.

Given an ordering $\sigma : E \mapsto \{1, \ldots, |E|\}$ on the edges of G, define the following cost function:

$$c_i(uv) = \begin{cases} c(uv) & i = 0 \\ 1 & i > 0,\ \sigma(uv) = i \\ 0 & i > 0,\ \sigma(uv) \neq i \end{cases}$$

With this, we define the following linear program:

$$\begin{aligned}
\min \sum_{S \in V \cup \mathscr{F}_x} & \frac{1}{|S|} r(S) && && (D^i_{\mathscr{F}}(G,c,\sigma,\Gamma,L,M,N,Q)) \\
\text{s.t. } r(S) + \Pi(S) &\geq \Gamma_i(S) && \forall\, S \in (V \cup \mathscr{F}_x) \setminus L \\
-r(S) + \Pi(S) &\leq \Gamma_i(S) && \forall\, S \in (V \cup \mathscr{F}_x) \setminus M \\
\sum_{S \in V \cup \mathscr{F}_x : uv \in \delta(S)} \Pi(S) &= c_i(uv) && \forall\, uv \in \operatorname{supp}(x) \\
\sum_{S \in V \cup \mathscr{F}_x : uv \in \delta(S)} \Pi(S) &\leq c_i(uv) && \forall\, uv \notin \operatorname{supp}(x) \cup N \\
\Pi(S) &\geq 0 && \forall\, S \in \mathscr{F}_x \setminus Q.
\end{aligned}$$

Intuitively, c_i and Γ_i correspond to the coefficients of ϵ^i in c and Γ if we were to perturb the edge costs on the graph by ϵ^i and run Algorithm 1.

Algorithm 4: Unperturbed C-P-Matching Algorithm

```
    Input: A graph $G = (V, E)$ with edge costs $c \in \mathbb{Z}^E$ and an ordering $\sigma : E \mapsto \{1, \ldots, |E|\}$.
    Output: A binary vector $x$ representing a minimum-cost perfect matching on $G$.
 1  $\mathscr{F} \leftarrow \emptyset$; $\Gamma_0, \ldots, \Gamma_{|E|} \leftarrow 0$
 2  repeat
 3      Let $x$ be the lexicographically-minimal optimal solution to $P_{\mathscr{F}}(G, c)$ with respect to $\sigma$.
 4      if $x$ is integral then
 5          return $x$
 6      $L \leftarrow \emptyset; M \leftarrow \emptyset; N \leftarrow \emptyset; Q \leftarrow \emptyset; D_0, \ldots, D_{|E|} \leftarrow 0$
 7      $\mathscr{F}_x \leftarrow \{S \in \mathscr{F} : x(\delta(S)) = 1\}$
 8      for $i \leftarrow 0$ to $|E|$ do
 9          Obtain an optimal solution $r, \Pi$ to $D^i_{\mathscr{F}}(G, c, \sigma, \Gamma, L, M, N, Q)$.

10          $L \leftarrow L \cup \{S \in V \cup \mathscr{F}_x : r(S) + \Pi(S) \neq \Gamma_i(S)\}$
11          $M \leftarrow M \cup \{S \in V \cup \mathscr{F}_x : -r(S) + \Pi(S) \neq \Gamma_i(S)\}$
12          $N \leftarrow N \cup \{uv \in E : \sum_{uv \in \delta(S)} \Pi(S) \neq c_i(uv)\}$
13          $Q \leftarrow Q \cup \{S \in \mathscr{F}_x : \Pi(S) \neq 0\}$

14          $D_i \leftarrow \Pi$
15      end
16      $\mathscr{H}' \leftarrow \{S \in \mathscr{F} : \exists\ i$ s.t. $D_i(S) > 0\}$
17      Let $\mathscr{C}$ be the set of odd cycles in supp($x$). For each $C \in \mathscr{C}$, let $V(C)$ be the union of $C$ with all sets
         in $\mathscr{H}'$ intersecting it.
18      $\mathscr{H}'' \leftarrow \{V(C) : C \in \mathscr{C}\}$
19      $\mathscr{F} \leftarrow \mathscr{H}' \cup \mathscr{H}''$
20      $\Gamma \leftarrow D$
21  end
```

Steps 10 through 13 exist to remove the slack constraints from the next iterations of $D^i_{\mathscr{F}}(G, c, \sigma, \Gamma, L, M, N, Q)$, as in Algorithm 3.

A reference implementation, written in Python 3, is available at [13].

▶ **Lemma 3.** *In every iteration of the Unperturbed C-P-Matching Algorithm (4), x is equal to its counterpart in the C-P-Matching Algorithm (1) with perturbations* $c(i) = \epsilon^i$.

Proof. As mentioned in Section 3, by [16], the lexicographically-minimal unperturbed primal solution is equal to the unique perturbed optimal primal solution for a given $\mathscr{F}$, so we need to only show that $\mathscr{F}$ is always equal to its counterpart.

Consider Γ as a single vector of polynomials in ϵ, with the coefficients of the ϵ^i terms given by Γ_i. Then, by Lemma 2, $y = \sum_i \epsilon^i D_i$ is an optimal solution to the linear program in step 8, and $y(S) > 0$ if and only if $\max(D_i) > 0$. But the linear program in question is exactly that which the C-P-Matching algorithm uses to obtain a Γ-extremal dual optimal solution. Therefore $\mathscr{H}'$, which is defined solely based on whether $y(S) > 0$ or not, is equal to its counterpart in the C-P-Matching algorithm. Since $\mathscr{H}''$ is defined exactly the same way as its counterpart, the two are equal, so $\mathscr{F}$ is equal to its counterpart as well. ◀

Since, by Lemma 1, neither the correctness nor the complexity of Algorithm 1 are affected by changing from the perturbation $c(i) = 2^{-i}$ to the perturbation $c(i) = \epsilon^i$, we can rephrase this to give

▶ **Theorem 4.** *The Unperturbed C-P-Matching Algorithm gives a minimum-cost perfect matching.*

The lemma also has the following

▶ **Corollary 5.** *The Unperturbed C-P-Matching algorithm requires solving $O(mn \log n)$ linear programming problems in the worst case.*

Proof. According to [2, Theorem 1], the C-P-Matching Algorithm takes at most $O(n \log n)$ iterations. The Unperturbed C-P-Matching Algorithm has the same number of iterations, but each iteration utilizes $2(m+1)$ linear programming problems, which is $O(m)$. Therefore, the Unperturbed C-P-Matching Algorithm requires solving $O(m) \times O(n \log n) = O(mn \log n)$ linear programming problems in total. ◀

8 Final remarks

We feel that there are combinatorial optimization problems for which there exist solution methods that assume the existence of a unique optimal solution in their analyses. In such cases, the method of solving a perturbed problem via lexicographic linear goal programming described in Section 6 would be applicable. Unfortunately, in our literature search and conversations with colleagues, we have not yet encountered an actual example. We hope that knowledgeable readers can inform us of such examples.

We do not yet know if our algorithm, when properly implemented and optimized, can in practice be made competitive with combinatorial methods such as Edmonds' blossom algorithm or with the original method implemented with an arbitrary-precision LP solver. In terms of theoretical complexity, our algorithm, which solves $O(mn \log n)$ linear programs, each of which requires the use of a theoretically polynomial-time solver, is significantly slower in the worst case than the best known asymptotic running time for an implementation of Edmonds' blossom algorithm which is $O(n(m + \log n))$ according to [14].

We encountered a number of interesting phenomena regarding the subroutine for finding the lexicographically-minimal primal optimal solution. Although, as written, it requires solving a fixed number of linear programs ($|E| + 1$), we noticed in empirical testing that it often gave this solution far more quickly than that, with the last few linear programs all giving the same answer. We did not investigate this any further, but hypothesize that shortcuts exist to decrease the runtime by a factor of $\frac{1}{4}$ or more. An immediate future research direction is to implement our algorithm using a highly efficient lexicographic linear goal programming solver that also takes advantage of the similarities of the linear programming problems that need to be solved and evaluate its empirical performance in comparison with existing algorithms for minimum-cost perfect matching.

Acknowledgments

The authors would like to thank James Addis, as well as the organizers and supporting organizations of the Fields Undergraduate Summer Research Program — Brittany Camp, Bryan Eelhart, Huaxiong Huang, Michael McCulloch, and the Fields Institute and Fields Centre for Quantitative Analysis and Modelling (Fields CQAM) — without whom this research would not have been possible.

References

1 David L. Applegate, William Cook, Sanjeeb Dash, and Daniel G. Espinoza. Exact solutions to linear programming problems. *Operations Research Letters*, 35(6):693–699, 2007.

2 Karthekeyan Chandrasekaran, László A. Végh, and Santosh S. Vempala. The cutting plane method is polynomial for perfect matchings. *Mathematics of Operations Research*, 41(1):23–48, February 2016.

3 A. Charnes. Optimality and degeneracy in linear programming. *Econometrica*, 20(2):160–170, April 1952.

4 William Cook and Rohe André. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing*, 11(2):138–148, 1999.

5 William Cook, William Cunningham, William Pulleyblank, and Alexander Schrijver. *Combinatorial optimization.* Wiley-Interscience series in discrete mathematics and optimization. Wiley, New York, 1998.

6 William Cook, Thorsten Koch, Daniel E. Steffy, and Kati Wolter. An exact rational mixed-integer programming solver. In Oktay Günlük and Gerhard J. Woeginger, editors, *Integer programming and combinatoral optimization*, volume 6655, pages 104–116. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

7 Jack Edmonds. Maximum matching and a polyhedron with $0, 1$ vertices. *J. of Res. the Nat. Bureau of Standards*, 69 B:125–130, 1965.

8 Jack Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.

9 Ambros M. Gleixner, Daniel E. Steffy, and Kati Wolter. Iterative refinement for linear programming. *INFORMS Journal on Computing*, 28(3):449–464, 2016.

10 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization.* Springer, 1988.

11 James Ignizio. *Introduction to linear goal programming*, volume 56 of *Sage University Paper Series on Quantitative Applications in the Social Sciences.* Sage Publications, Beverly Hills, CA, 1985.

12 James P. Ignizio. An algorithm for solving the linear goal programming problem by solving its dual. *Journal of the Operational Research Society*, 36(6):507–515, 1985.

13 Paul Michael Kielstra. Code for revisiting a cutting plane method for perfect matchings, August 2019.

14 Vladimir Kolmogorov. Blossom V: A new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, 1(1):43–67, July 2009.

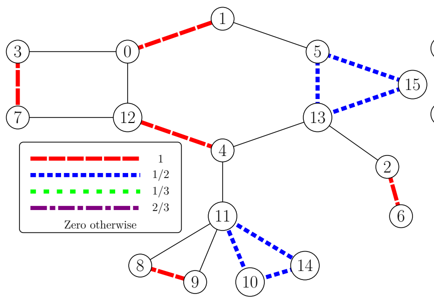
15 Manfred W. Padberg and M. R. Rao. Odd minimum cut-sets and b-matchings. *Math. Oper. Res.*, 7(1):67–80, February 1982.

16 Alexander Schrijver. *Theory of Linear and Integer Programming.* Wiley-Interscience series in discrete mathematics and optimization. Wiley, Chichester, reprinted edition, 2000. OCLC: 247967491.

A Appendix

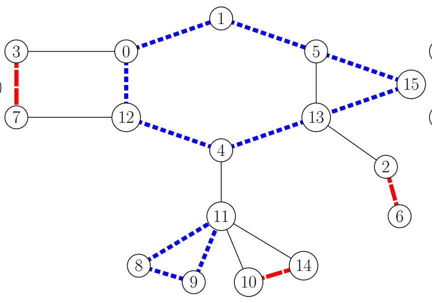
The following sequence of figures illustrate how the Dancing Robot using the same ordering as in Section 4 cycles every six iterations and gives rise to non-integral primal solutions every three iterations. The arrows in the captions indicate the order of the cycling sequence.

Edge ordering:
(1, 5), (2, 13), (10, 14), (0, 3),(4, 12), (5, 13), (7, 12), (5, 15), (3, 7), (8, 9),
(0, 1), (11, 14), (0, 12), (4, 13), (2, 6), (10, 11), (9, 11), (4, 11), (8, 11), (13, 15)



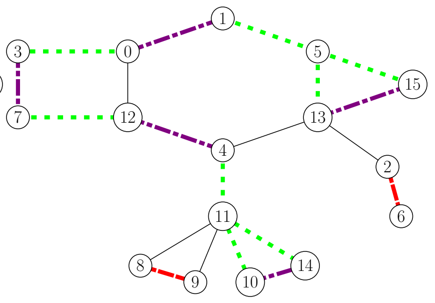
$\Pi_1(v) = \frac{1}{2}\ \forall v \in \mathcal{V}$,
$\mathscr{F}_1 = \emptyset$

Figure A.1 First Iteration ⇒



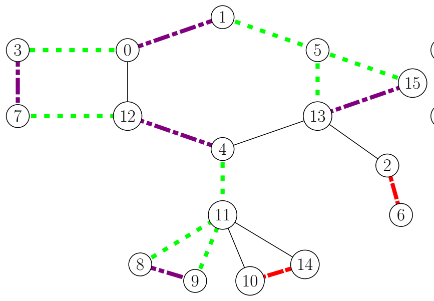
$\Pi_2(v) = \frac{1}{2}\ \forall v \in \mathcal{V}, \mathscr{F}_2 =$
$\{\{5, 15, 13\}, \{10, 11, 14\}\}$,
$\Pi_2(S) = 0\ \forall S \in \mathscr{F}_2$

Figure A.2 Second Iteration ⇒



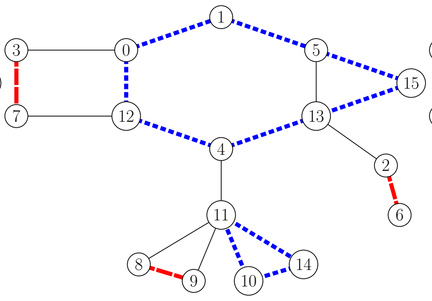
$\Pi_3(v) = \frac{1}{2}\ \forall v \in \mathcal{V}, \mathscr{F}_3 =$
$\{\{0, 1, 5, 15, 13, 4, 12\}, \{8, 11, 9\}\}$,
$\Pi_3(S) = 0\ \forall S \in \mathscr{F}_3$

Figure A.3 Third Iteration ⇓



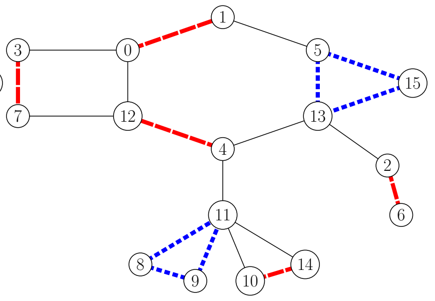
$\Pi_6(v) = \frac{1}{2}\ \forall v \in \mathcal{V}, \mathscr{F}_6 =$
$\{\{10, 11, 14\}, \{0, 1, 5, 15, 13, 4, 12\}\}$,
$\Pi_6(S) = 0\ \forall S \in \mathscr{F}_6$

Figure A.6 Sixth Iteration ⇑



$\Pi_5(v) = \frac{1}{2}\ \forall v \in \mathcal{V}, \mathscr{F}_5 =$
$\{\{5, 13, 15\}, \{8, 9, 11\}\}$,
$\Pi_5(S) = 0\ \forall S \in \mathscr{F}_5$

Figure A.5 Fifth Iteration ⇐



$\Pi_4(v) = \frac{1}{2}\ \forall v \in \mathcal{V}, \mathscr{F}_4 =$
$\{\{13, 15, 5, 1, 0, 3, 7, 12, 4, 11, 14, 10\}\}$,
$\Pi_4(S) = 0\ \forall S \in \mathscr{F}_4$

Figure A.4 Fourth Iteration ⇐