Algorithm for the generalised BvN decomposition

Damien Lesens

April 2024

1 Introduction

The goal of this document is to present an algorithm for the generalized Birkhoff Von Neumann decomposition. We are given a graph which is a convex combination of perfect matchings, and we want to compute a valid decomposition. This was shown to be in P by Vazirani in [5] but his algorithm is very costly and unpractical. We present here a new algorithm, both faster and easily implementable.

In the following, we selected the elements from [5] that are needed to understand our algorithm, while adding some details in some proofs and regarding the implementation.

2 Definitions

We are given an undirected graph $G = (V, E)$, with $|V| = n$ vertices, $|E| = m$ edges, and n is even. For a vertex v, we use $\delta(v)$ to denote the set of edges incident on v, which is extended to a set of vertices S as $\delta(S) = \{e : e \in E, e = (s, v) \text{ with } s \in S, \text{ and } v \notin S\}$.

A matching in G is a set of edges no two of which share a vertex. A matching is called perfect if it matches all vertices. Edmonds [1] shows that the convex hull of all perfect matchings in G, which is a polytope, is defined by the following constraints:

$$\sum_{e\in\delta(v)} x_e = 1 \quad \text{for all } v \in V \tag{1a}$$

$$\sum_{e\in\delta(S)} x_e \geq 1 \quad \text{for all odd set } S \tag{1b}$$

$$x_e \geq 0 \quad \text{for all } e \in E \tag{1c}$$

A point x of this polytope is a *fractional perfect matching* in G.

By an *odd set in* G we mean a set $S \subset V$, with $|S|$ odd and $|S| \geq 3$ (excluding singletons).

An *α-fractional perfect matching* for $1 \geq \alpha \geq 0$ is a point of the polytope obtained by replacing 1's with α's in the right hand side of the polytope specified in (1). That is,

$$\sum_{e\in\delta(v)} x_e = \alpha \quad \text{for all } v \in V \tag{2a}$$

$$\sum_{e \in \delta(S)} x_e \geq \alpha \quad \text{for all odd set } S \tag{2b}$$

$$x_e \geq 0 \quad \text{for all } e \in E \tag{2c}$$

If x is a *α-fractional perfect matching*, then $\frac{1}{\alpha}x$ is a *fractional perfect matching*.

For $S \subset V$, the *cut of* S is the set of edges having exactly one end point in S, and hence equivalent to $\delta(S)$. Given a matching M and a set of vertices S, we say that M *crosses* the cut S if $M \cap \delta(S) \neq \emptyset$. In other words, a matching M crosses a cut S, if M matches at least one vertex of S to a vertex in $V \setminus S$. If M is a perfect matching, then M must cross every odd set S as captured by the inequality (1b).

We will note $G_x = (V, E_x)$ the graph G with edge weights x where E_x contains only edges e for which $x_e > 0$. We note $x(\delta(S))$ the value of the cut of S with respect to x, i.e., $x(\delta(S)) = \sum_{e \in \delta(S)} x_e$.

We note the following lemma about the odd set constraints (2b) for later use.

Lemma 2.1. *The polytope constraint $x(\delta(S)) \geq \alpha$ for all odd set S is equivalent to $x(E(S)) \leq \alpha\frac{|S|-1}{2}$ for all odd set S, where $E(S) = \{(u, v) \in E | u \in S, v \in S\}$.*

Proof. We have

$$\sum_{v \in S} \sum_{e \in \delta(v)} x(e) = x(\delta(S)) + 2x(E(S))$$

because edges exiting S are counted once and edges inside S twice. Because of constraints (2a) we have furthermore that

$$\alpha|S| = x(\delta(S)) + 2x(E(S)) \tag{3}$$

$$\Leftrightarrow x(E(S)) = \frac{\alpha|S| - x(\delta(S))}{2}$$

which can be used to show the equivalence of the constraints. □

A minimum odd cut in G_x is an odd set S with $|S| \geq 3$, where $x(\delta(S))$ is minimum among all odd sets. Odd sets S for which constraint (2b) is tight are called tight odd α-cuts. **These sets are central for the algorithm**.

Lemma 2.2 (Vazirani [5], Lemma 3)**.** *Let $b \cdot M$ be a term in some decomposition of x, where M is a perfect matching in G_x. Then, M must cross every tight odd α-cut exactly once.*

Proof. This proof is from Vazirani [5, Lemma 3]. M must cross every tight odd α-cut at least once because they are odd sets. Let $x' = x - b \cdot M$. Since $b \cdot M$ is a term in a valid decomposition of x, x' must be a $(\alpha - b)$-fractional perfect matching. If M crosses a tight odd α-cut S a number $r > 1$ times, then x' would not be an $(\alpha - b)$-fractional matching, which is a contradiction: it holds that $\sum_{e \in \delta(S)} x'_e = \alpha - rb$, which violates the constraint in (2b). □

Remark. *Looking at the polytope (1), a first question one might ask is whether it is in P to test if a point x belongs to the polytope. Indeed, constraints (1a) are easy to test, but there are exponentially many constraints (1b). However, Padberg and Rao show [2] that it is in P to find a minimum odd cut in a weighted graph. Using this algorithm, one* *can test if x is in Edmond's polytope by checking whether the minimum odd cut has value greater or less than 1. The minimum odd cut algorithm of Padberg and Rao will be used extensively in the next sections.*

3 Overview of the algorithm

Similarly to the bipartite Birkhoff algorithm, we are going to start with a 1-*fractional perfect matching* x, subtract a scalar $\beta > 0$ times a perfect matching M such that $x - \beta \cdot M$ is a $(1-\beta)$-*fractional perfect matching*, and repeat until we have reduced all edge weights to zero.

Like in the bipartite case, any perfect matching and coefficient will preserve the conditions (2a). However, in the general case we have odd set constraints which prevent us from choosing any perfect matching.

Indeed, a perfect matching in the decomposition needs to cross each tight odd α-cut exactly once. Let $\mathcal{F}$ be the family of all tight odd α-cuts. We can find a matching which crosses every set in $\mathcal{F}$ the following way. For each $e \in E_x$, define cuts(e)= $\{C \in \mathcal{F} | e \in \delta(C)\}$, and define the weight of edge e to be the cardinality of this set, i.e., $w(e) = |\text{cuts}(e)|$. We will choose a minimum weight perfect matching with respect to this weight w.

Lemma 3.1. *Let M be a minimum weight perfect matching in graph G_x under weight function w. Then the weight of this matching is $|\mathcal{F}|$ and it crosses **each** tight odd α-cut exactly once.*

Proof. We are going to show that the weight of a matching under weight function w is at least $|\mathcal{F}|$ and that this lower bound is reached by matchings which cross each tight odd α-cut exactly once. First, a matching crosses a tight odd α-cut at least once so the weight of any matching is at least $|\mathcal{F}|$. This lower bound is reached for matchings crossing each tight odd α-cut exactly once. Such a matching exist: consider any M in a valid decomposition of G_x. In other words, there is the equivalence: a PM M is a term in some decomposition of x iff M is MWPM in G_x and has a weight $|\mathcal{F}|$ □

Even if we know how to find a matching crossing every tight odd α-cut exactly once, the issue now is that there could be exponentially many such tight odd α-cut so building the weight function w can not be done in polynomial time.

The next section will show how we can address this issue by only having to consider a special subset of tight odd α-cuts of polynomial size.

4 Laminar families and uncrossing

Definition 4.1. *We say that two sets S and T are **intersecting** if $S \backslash T \neq \emptyset$, $T \backslash S \neq \emptyset$ and $S \cap T \neq \emptyset$.*

Remark. *This is **not** the classical definition with only $S \cap T \neq \emptyset$. This might be confusing but this is the term used in the literature.*

A *laminar* family $\mathcal{L}$ is a family of subsets of a ground set such that for all $S, T \in \mathcal{L}$, either $S \subset T$, or $T \subset S$, or $S \cap T = \emptyset$. In other words, a family of sets is laminar if no two sets in it are intersecting. A laminar family is *maximal* if one cannot add any more subsets of the ground set to it.

The algorithm will build and maintain a maximal laminar family of tight odd α-cuts $\mathcal{L}_\alpha$ which we will use to choose matchings that cross each tight odd α-cut exactly once. We use laminar families because we have the following lemma about intersecting tight odd α-cuts. This lemma is central to the algorithm.

Lemma 4.1. *Let x be an α-fractional perfect matching and M be a perfect matching in a given graph G. Let S and T be two intersecting tight odd α-cuts, where $|\delta(S) \cap M| > 1$. Then there exists a tight odd α-cut S' such that $|\delta(S') \cap M| > 1$ and S' and T are not intersecting. Moreover, S' is one of the four sets $S \cap T, S \cup T, S \backslash T,$ and $T \backslash S$.*

We will use this lemma to prove properties about maximal laminar families of tight odd α-cuts and to build them. The proof of this lemma was not present in Vazirani's paper but we explicit here because it is central to our new algorithm. This type of proof is well known in the literature. An example can be found in [4]. Goemans' slides [6] on uncrossing also indicate the 2 cases we explicit in the proof.

Proof. Recall that we note $\mathcal{F} = \{S | S \text{ odd with size} \geq 3 \text{ and } x(\delta(S)) = \alpha\}$ the family of all tight odd α-cuts.

We will use the polytope constraint $x(E(S)) = \frac{\alpha|S| - x(\delta(S))}{2}$ of Lemma 2.1.

We will also use the following results for the cut function $d(S) = x(\delta(S))$ for $S \subset V$ (from Goemans [6]):

$$d(A) + d(B) = d(A \cup B) + d(B \cap A) + 2x(A \backslash B : B \backslash A) \tag{4}$$

this can be easily seen by counting the contribution of each edge on both side. We also obtain

$$d(A) + d(B) = d(A \backslash B) + d(B \backslash A) + 2x(A \cap B : \overline{A \cup B}) \tag{5}$$

from the previous equality by noting $d(S) = d(\overline{S})$

For any set $F \subseteq E$, let $\chi(F) \in \mathbb{R}^{|E|}$ denote the characteristic vector of F. For a given perfect matching M let $|F|_M = |\delta(F) \cap M|$

Lemma 4.2. *If $S, T \in \mathcal{F}$ are intersecting then exactly one of the following is true:*

- *$S \cap T$ and $S \cup T$ are in $\mathcal{F}$ if they are not singletons and $|S|_M + |T|_M = |S \cup T|_M + |S \cap T|_M$*
- *$S \backslash T$ and $T \backslash S$ are in $\mathcal{F}$ if they are not singletons and $|S|_M + |T|_M = |S \backslash T|_M + |T \backslash S|_M$*

Proof. We have two cases depending on the parity of $S \cap T$:

Case 1: If $S \cap T$ is odd, then $S \cup T$ is also odd and we have

$$|S| + |T| = |S \cup T| + |S \cap T|$$
$$\geq \frac{2}{\alpha}(x(E(S \cup T)) + x(E(S \cap T)) + 2$$
$$\geq \frac{2}{\alpha}(x(E(S)) + x(E(T)) + 2$$

$$= |S| + |T|$$

the first inequality is because $S \cap T$ and $S \cup T$ are odd. The second comes from rewriting the equation (5) using the equation (3) and $d(S) = x(\delta(S))$.

$$\alpha|S| - 2x(E(S)) + \alpha|T| - 2x(E(T)) = \alpha|S \cap T| - 2x(E(S \cap T)) + \alpha|S \cup T| - 2x(E(S \cup T)) + 2x(S\backslash T : T\backslash S)$$

$$\Leftrightarrow x(E(S \cap T)) + x(E(S \cup T)) = x(E(S)) + x(E(T)) + x(S\backslash T : T\backslash S)$$

We obtain the same result in the end so we have equality throughout. Hence $S \cap T$ and $S \cup T$ are tight odd α-cuts and there are no edges between $S\backslash T$ and $T\backslash S$. Thus $\chi(E(S)) + \chi(E(T)) = \chi(E(S \cup T)) + \chi(E(S \cap T))$ and $|S|_M + |T|_M = |S \cup T|_M + |S \cap T|_M$.

Case 2: If If $S \cap T$ is even, then $S\backslash T$ and $T\backslash S$ are odd and we have:

$$|S| + |T| = |S\backslash T| + |T\backslash S| + 2|S \cap T|$$
$$\geq \frac{2}{\alpha}(x(E(S\backslash T)) + x(E(T\backslash S) + \alpha|S \cap T|) + 2$$
$$\geq \frac{2}{\alpha}(x(E(S)) + x(E(T)) + 2$$
$$= |S| + |T|$$

the second inequality is obtained using equation 5

$$\alpha|S| - 2x(E(S)) + \alpha|T| - 2x(E(T)) = \alpha|S\backslash T| - 2x(E(S\backslash T)) + \alpha|S\backslash T| - 2x(E(T\backslash S)) + 2x(S \cap T : \overline{S \cup T})$$

$$\Leftrightarrow \alpha|S \cap T| + x(E(S\backslash T)) + x(E(T\backslash S)) = x(E(S)) + x(E(T)) + x(S \cap T : \overline{S \cup T})$$

We have equality though out thus $S\backslash T$ and $T\backslash S$ are tight odd α-cuts and there are no edges between $S \cap T$ and $\overline{S \cup T}$. We get similarly $\chi(E(S)) + \chi(E(T)) = \chi(E(S\backslash T)) + \chi(E(T\backslash S))$ and $|S|_M + |T|_M = |S\backslash T|_M + |T\backslash S|_M$ □

With this intermediate result we get lemma 4.1: $S \cap T$, $S \cup T$, $S\backslash T$ and $T\backslash S$ do not intersect T and because of the equalities $|S|_M + |T|_M = |S\backslash T|_M + |T\backslash S|_M$ and $|S|_M + |T|_M = |S \cup T|_M + |S \cap T|_M$ if $|S|_M > 1$ then $|S'|_M > 1$

□

In order to find a perfect matching that crosses every tight odd α-cut in polynomial time, we will pick one that crosses every $C \in \mathcal{L}$ exactly once, with $\mathcal{L}$ a maximal laminar family of tight odd α-cut.

Like before for each $e \in E_x$, define cuts(e)= $\{C \in \mathcal{L} | e \in \delta(C)\}$, i.e., the set of cuts in $\mathcal{L}$ which e crosses. Then define the weight of edge e to be the cardinality of this set, i.e. $w(e) = |\text{cuts(e)}|$. By lemma 3.1 we know that a perfect matching of minimum weight will cross every $C \in \mathcal{L}$ exactly once. This actually implies that it will cross **every** tight odd α-cut exactly once.

Lemma 4.3. *Let M be a perfect matching and $\mathcal{L}$ a maximal laminar family of tight odd α-cut. If M crosses every $C \in \mathcal{L}$ exactly once, then M crosses every tight odd α-cut exactly once.*

Proof. This proof was not fully detailed in Vazirani's paper. It is adapted from Theorem 4 in [4].

Suppose M crosses some tight odd α-cut S more than once. If there are several such sets S, choose one that intersects as few sets of $\mathcal{L}$ as possible. Suppose S intersects some $T \in \mathcal{L}$ (otherwise S can be added to $\mathcal{L}$, contradicting maximality). From Lemma 2.2 there exists a tight odd α-cut S' such that $|\delta(S) \cap M| > 1$ and that doesn't intersect T, contradicting the choice of S.

□

An key fact about a maximal laminar family of tight odd α-cuts is that its size is bounded by $\frac{n}{2} - 1$. This is shown by Vazirani and will be used later on.

5 Algorithm

Suppose that we are advanced in the algorithm: we have an α-fractional perfect matching x and a corresponding maximal laminar family of tight odd α-cuts $\mathcal{L}$. We will explain later how to build and update the family throughout the algorithm.

We know how to choose a matching so that it crosses ever tight odd α-cut exactly once. We now have to find a coefficient b such that $x - b \cdot M$ is a $(\alpha - b)$-fractional perfect matching.

In the bipartite case, the coefficient was always $\beta = \min_{e \in M}(x_e)$.

Here, we cannot choose $b = \min_{e \in M}(x_e)$ in general: there could be an odd set S of cut value $\alpha + t$ which is crossed by M $|S|_M > 1$ times and such that $\alpha + t - b \cdot |S|_M < \alpha - b$. Setting $b = \beta$ will not respect the constraint (2b) for S on the next iteration. We thus realize that in order for the constraint on S to be respected, we have to take b which satisfies

$$\alpha + t - b \cdot |S|_M = \alpha - b$$
$$\Leftrightarrow b = \frac{t}{|S|_M - 1} = \frac{x(\delta(S)) - \alpha}{|S|_M - 1}\ .$$

In other words, if taking coefficient $b = \min_{e \in M}(x_e)$ yields a minimal odd cut that is of value strictly less than $\alpha - b$, we need to take as coefficient

$$\gamma = \min_{S \in \mathcal{S}} \frac{x(\delta(S)) - \alpha}{|S|_M - 1} \tag{6}$$

where $\mathcal{S} = \{\text{odd sets S with } |S|_M > 1\}$

This value γ can be found by binary search in the range $[0, \min_{e \in M}(x_e)]$, as it can be showed that γ has the following equivalent definition:

$$\gamma = \arg\max_b \{(y - b \cdot M) \text{ is a } (\alpha - b)\text{-fractional perfect matching}\}$$

Vazirani shows that the binary search will always take polynomial time, because the denominators of the values in x do not grow too much. We will not discuss this aspect here.

An important observation is that by taking the coefficient to be γ, we create at least one new tight odd α-cut and this odd cut can **always** be added to $\mathcal{L}$. This can be proven using multiple times Lemma 4.1 and the proof actually gives an algorithm.

We have an tight odd α-cut S with $|S|_M > 1$. If S intersects $T \in \mathcal{L}$, by Lemma 4.1 we can replace it with S' such that $|S'|_M > 1$ and it doesn't intersect T. Repeating with this new set for all $T \in \mathcal{L}$ that intersect it, we get a tight odd α-cut that can be added to $\mathcal{L}$.

This algorithm only allows to add one new tight odd α-cut to $\mathcal{L}$ but there can be many tight odd α-cut created at once. That is the reason why, in his version of the algorithm that we will explicit next, Vazirani is not using this procedure. We will explain in the section after how did we make use of it.

6 Vazirani's version

The pseudo-code for Vazirani's algorithm is Algorithm 1. At each iteration of the while loop, $\mathcal{L}$ is a maximal laminar family of tight odd α-cuts. At line 4 the procedure build$\mathcal{L}$ builds a maximal laminar family of tight odd α-cuts and at line 19 the procedure update$\mathcal{L}$ updates the family after we created at least a new tight odd α-cut by taking the coefficient of the component to be γ.

The procedures build$\mathcal{L}$ and update$\mathcal{L}$ are essentially the same algorithm and cost in the worst case $O(nm^2)$ calls to the Padberg-Rao algorithm.

They use a subroutine which finds a minimum odd-cut which is not a singleton. This procedure works by iterating on all the pair of edges of the graph, substracts to a pair $\epsilon = O(1/n^n)$ and calls Padberg-Rao. This costs $O(m^2)$ calls to Padberg-Rao and the use of an epsilon this small necessitate a multiprecision framework to handle edge weights.

This procedure is used to find a first tight odd α-cut and we add it to $\mathcal{L}$. Then the algorithms repeatedly finds new tight odd α-cuts which are not already in $\mathcal{L}$ and that don't intersect tight odd α-cuts already in it. This involves shrinking sets of vertices in the graph, we redirect to the original paper for more details.

The procedure update$\mathcal{L}$ works in the same way: it adds non-intersecting tight odd α-cuts to $\mathcal{L}$ until it can't anymore.

These procedures are the most costly steps in terms of computation time, and our main contribution, as we will see in the next section, is to avoid using them.

7 Issues in Vazirani's version

A first issue with Vazirani's version is as explain above the cost of the procedure build$\mathcal{L}$ and update$\mathcal{L}$ ans their impractability, due to the use of very small epsilons. The use of epsilons can actually be avoided, and we will present next a way to also avoid iterating on pairs of edges.

Second issue is if there is a cut S that is crossed $k = |\delta(S) \cap M| > 1$ times and has cut value $\tau = \alpha + (k-1)\beta$. Then it will not be detected and added to L, because we will pass through the if at line 10. This impacts the computation time as we have to run update$\mathcal{L}$ at each iteration. With this correction the cost of his algorithm becomes $O(n^2m^3)$ max-flow min-cut.

Algorithm 1 Vazirani's algorithm for generalized BvN decomposition

```
Input: $x$                                            ▷ fractional perfect matching
$\alpha := 1$;
$y := x$;
$\mathcal{L}$ :=build$\mathcal{L}$;
while $y \neq 0$ do                                   ▷ at most $m$ iterations
    $M :=$ minimum weight perfect matching in $G_y$ wrt. weights $w(\mathcal{L})$
    $\beta := \min_{e \in M} y_e$
    $y' := y - \beta M$
    $S :=$ a minimum odd cut in $(G_{y'}, y')$
    if $y'(\delta_{y'}(S)) \geq \alpha - \beta$ then
        Output $\beta \cdot M$
        $\alpha := \alpha - \beta$
        $y := y'$
    else
        via binary search in $[0, \beta]$, find $\gamma$.
        Output $\gamma \cdot M$
        $y' := y - \gamma \cdot M$
        $\alpha := \alpha - \gamma$
        update$\mathcal{L}$
    end if
end while
```

8 Our version

The idea is to build the laminar family lazily, adding elements by uncrossing only when we need to. Instead of using the routine iterating over all pairs of edges, we will use unsuitable perfect matchings to find tight odd α-cuts.

We start with $\mathcal{L} = \emptyset$, and find a min weight matching according to $w(\mathcal{L})$ (like defined by Vazirani). At the first iteration it will return any perfect matching. If this matching crosses every tight odd α-cutexactly once, we are good and can proceed like Vazirani. Else, we will find a tight odd α-cut by binary search: its γ is 0 so we it will be the minimum odd set for b small enough. For this part we only use original Padberg-Rao and the search has polynomial time, like explained by Vazirani. We can then uncross this cut and add a new member to $\mathcal{L}$ because it is crossed by the matching more than once and it is a tight odd α-cut. We then update $w(\mathcal{L})$, find the new minimum weight matching and repeat.

A pseudo-code of our algorithm can be found in algorithm 2.

In the following we will note F the cost of a max-flow min-cut computation. In theory $F = O(m^{1+o(1)})$ but is only implemented in $O(n^2\sqrt{m})$. For our algorithm, the fix cost per component remains $O(nF)$ like Vazirani. However, the additional cost to handle the laminar family is now $O((n^3 \log(n) + n^2 \log(d)m)F)$ instead of $O(n^2m^3F)$ before.

Moreover, in the best case scenario for us the cost is only $O(knF)$ (where k is the number of components) whereas for Vazirani it is at $O(knF + m^3F)$.

Algorithm 2 New algorithm for generalized BvN decomposition

```
1: **Input**: $x$                                   ▷ fractional perfect matching
2: $\alpha := 1$;
3: $y := x$;
4: $\mathcal{L} := \emptyset$;
5: **while** $y \neq 0$ **do**                       ▷ at most $m$ iterations
6:     $M :=$ minimum weight perfect matching in $G_y$ wrt. weights $w(\mathcal{L})$
7:     $\beta := \min_{e \in M} y_e$
8:     $y' := y - \beta M$
9:     $S :=$ a minimum odd cut in $(G_{y'}, y')$
10:    **if** $y'(\delta_{y'}(S)) \geq \alpha - \beta$ **then**
11:        **Output** $\beta \cdot M$
12:        $\alpha := \alpha - \beta$
13:        $y := y'$
14:    **else**
15:        via binary search in $[0, \beta]$, find $\gamma$.
16:        **if** $\gamma > 0$ **then**
17:            **Output** $\gamma \cdot M$
18:            $y' := y - \gamma \cdot M$
19:            $\alpha := \alpha - \gamma$
20:        **end if**
21:        $S' :=$ uncross($S$,$\mathcal{L}$)
22:        $L = L \cup \{S'\}$
23:        update $w(\mathcal{L})$
24:    **end if**
25: **end while**
```

Table 1: Performance of heuristics on a set of constructed matrices

n	k	"Any" matching	Bottleneck
50	10	55.4	23
100	30	167.8	65.6
200	40	223.4	80.4

9 Implementation and experiments

All the code is in Python, using Networkx algorithm for minimum weight perfect matching and max-flow min-cut. I actually didn't use Padberg-Rao but Rizzi's algorithm [3] which is faster on average.

Furthermore, I implemented a version in which at each step amongst matching crossing all tight odd α-cuts, we pick one with a minimal edge that is maximal. This will give the largest possible coefficient if we don't use the binary search. If we use the binary search, it will give the range $[0, b]$ the largest possible and hopefully the coefficient the largest possible.

I compared this bottleneck version with the original one on some artificial instances, built the following way: given n even and k the number of components, pick uniformly at random k perfect matchings in $[\![1, n]\!]$ as well as k coefficients in $[\![1, 10]\!]$ and add them up to form a weighted graph. Such graph is an α-fractional perfect matching, with α equal to the sum of coefficients. I had to restrict myself to such artificial instances as my code is only at a proof of concept phase and can only handle integer edge values. Future versions will be able to handle floating points values like in the bipartite case.

Results are presented in Table 1. Similarly to the bipartite case, picking a bottleneck matching greatly reduces the number of components produced, by a factor more than 2.

References

[1] Jack Edmonds. "Maximum matching and a polyhedron with 0, 1-vertices". In: *Journal of Research of the National Bureau of Standards B* 69 (1965), pp. 125–130.

[2] Manfred W. Padberg and M. R. Rao. "Odd Minimum Cut-Sets and b-Matchings". In: *Mathematics of Operations Research* 7.1 (1982), pp. 67–80. ISSN: 0364765X, 15265471. URL: `http://www.jstor.org/stable/3689360` (visited on 05/24/2024).

[3] Romeo Rizzi. "A simple minimum T-cut algorithm". In: *Discrete Applied Mathematics* 129.2 (2003), pp. 539–544. ISSN: 0166-218X. DOI: `https://doi.org/10.1016/S0166-218X(03)00182-3`. URL: `https://www.sciencedirect.com/science/article/pii/S0166218X03001823`.

[4] Michel X. Goemans. "Minimum bounded degree spanning trees". In: *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE. 2006, pp. 273–282.

[5] Vijay V. Vazirani. *An Extension of the Birkhoff-von Neumann Theorem to Non-Bipartite Graphs.* 2020. arXiv: `2010.05984 [cs.DS]`.

[6] Michel X. Goemans. *Uncrossing.* URL: `http://www.science.unitn.it/cirm/Goemans.pdf`.