NYILATKOZAT

Név: Szabó Eszter
ELTE Természettudományi Kar, szak: Alkalmazott matematikus MSc
NEPTUN azonosító: V26R76
Diplomamunka címe: Parametric search approach for graph packing problems

A **diplomamunka** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2021.05.31.

Szabó Eszter

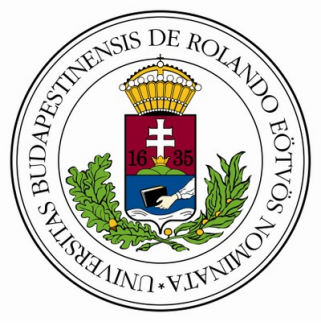
a hallgató aláírása

Eötvös Loránd University
Faculty of Science

Parametric search approach for graph packing problems

Eszter Szabó
MSc in Applied Mathematics

Supervisor:
Alpár Jüttner
Operations Research Department



2021, Budapest

Acknowledgement

I would like to thank Alpár Jüttner, my supervisor for his guidance and helpful discussions. I am grateful for the valuable time he spent to help me.
I would also like to thank my family for their support.

Contents

1 Introduction

This thesis presents some algorithm for combinatorial optimization, based on parametric search and Newton's approach. We consider two topics in graph theory. In the first part, we describe balanced network and submodular flow problem and related algorithms. The second part deals with packing problems in graphs.

The first section introduces some fundamental definitions and theorem, that we will use throughout the paper.

In the third section, we show balanced network flow problem and we describe Scutellà's algorithm. Balanced optimization problems model is meant to find equitable distribution of resources. Several balanced optimization problems have been analysed in the literature. Like the balanced spanning tree problem has been studied by Camerini [4] just like the balanced assignment problem by Martello [14]. Ahuja proposed a parametric simplex method for the general balanced linear programming problem [1].
We deal with the balanced network flow problem, the problem of finding a feasible flow on a given network which minimizes the difference between the maximum and the minimum weighted flow on single arcs. First we describe parametric formulation of this problem and its dual, which is called maximum mean cut problem. This can be solved in polinomail time by S.Thomas McCormick and Thomas R. Ervolina [15]. Then we present Scutella's algorithm for balanced network flow problem using Newton's approach. In the end of this section, we analyse the iteration number of Newton's method, that is proposed by Radzik [18].

In section 4, we consider the balanced submodular network problem, that is more general balanced problem. By using Scutella technique, we show a strongly polynomial algorithm. It should be pointed out that, being this problem a linear problem with $(0, -1, +1)$-matrix, it can be solved in strongly polynomial time by Éva Tardos [19]. In the beginning of this section, we formulate the problem, then we examine parametric problem and its dual. We show that the dual parametric problem is equivalent to minimum mean cut problem than we present an algorithm for solve this problem. Then we use Newton's approach to find the optimal balanced submodular flow and we estimate the number of iterations of the Newton method. In the end of this section, we consider the same problem with intersecting submodular function.

The fifth section deals with maximum packing of T-joins problem. T-joins appear in the solution of the Chinese postman problem by Edmonds and Johnson [5]. They gave a combinatorial polynomial algorithm to solve maximum packing of T-cuts problem and its dual [5]. The algorithm of Edmonds and Johnson can be modified to produce this integer dual optimal solution [2]. There are many other packing

problems in the literature, like maximum packing arborescences, that is solved by Gabow and Manu [8]. In the beginning of this section, we formulate the maximum packing of T-joins problem, using theory of Blocking Polyhedra [7]. Then we describe Padberg and Rao's algorithm for finding a minimum T-cut [17], and thereafter we give a description and analysis Barahona's algorithm for T-join packing problem [3]. This algorithm is based on the idea, that we can compute the weight of a given T-join in the maximal packing. For this Newthon's approach is used again.

In the sixth section we present a polynomial algorithm for maximum packing of perfect matching problem in bipartite graphs along Barahona's algorithm. First we deal with formulation and we show an algorithm for compute the optimal value of the problem. Then we examine dual optimal solutions, we give some statement to them, after that we present an algorithm for maximum packing problem in bipartite graph, which is very similar to Barahona's algorithm.

The last section deals with some problem related to previous two sections. In the beginning, we describe 6 problems corresponding to packing problem, and thereafter we give their LP formulations. Then we discuss their relationships. We show that minimum cost perfect matchings problem is reducible to minimum cost 1-packing of T-joins problem.

2 Preliminaries

This section introduces some notations, definitions and fundamental theorem, that will be used in the sequence.

Throughout the paper, we are concerned with directed or undirected graphs containing no graph loops or multiple edges. We use the following definition of network flow problem:

Definition 2.1 (Network flow problem)**.** *The input of network flow problem is a directed graph $G = (V, E)$ and a demand function $b = V \to \mathbb{R}$. The goal is to construct a flow $f : E \to \mathbb{R}^+$, i.e. each edge receives a flow, such that the difference between the sum of incoming and outgoing flow for each node v is equal to $b(v)$. If this equation holds for all $v \in V$, then we say that f satisfies all demands.*
Each edge can have a capacity. In this case the amount of flow on an edge can't exceed the capacity of the edge. (If there is no capacity constraint, the amounts of flow can be arbitrary large.)

Newthon method's is a well-known root-finding algorithm which produces successively better approximations to the roots of a real-valued function. Its basic idea is to start with an initial guess, then to approximate the function by its tangent line, and thereafter to compute the x-intercept of this tangent line. This x-axis intercept will be a better approximation than the previous one. The method can be iterated, until we find our end point. We will use this approach not only to find the root, but also to determine minimum point of a convex function or first break-point of a piece-wise linear function.

Let us see some definitions:

Definition 2.2. *Let S be a finite set and $f : 2^S \to \mathbb{R}$ a set function defined in the subsets of S. The function f is a submodular function, if the following inequality holds for every $X, Y \subseteq S$:*

$$f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$$

We say that $X, Y \subseteq S$ are intersecting, if $X \cap Y \neq \emptyset$

Definition 2.3. *Let S be a finite set and $f : 2^S \to \mathbb{R}$ a set function defined in the subsets of S. The function f is an intersecting submodular function, if the following inequality holds for all intersecting subsets $X, Y \subseteq V$:*

$$f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$$

Definition 2.4. *A set family $\mathcal{F}$ is a laminar set family, if each pair of sets are either disjoint or related by containment. Formally, for all $X, Y \in \mathcal{F}$ one of the following holds: $X \cap Y = \emptyset$, $X \subseteq Y$ or $Y \subseteq X$.*

We will use the next fundamental theorem by Hoffman [11].

Theorem 2.1 (Hoffman)**.** *Let $G = (V, E)$ be a digraph and b be a demand function $b = V \to \mathbb{R}$, with lower bound $l : E \to \mathbb{R}$ and upper bounds $u : E \to \mathbb{R}$ on the arcs, such that $l \leq u$. There exists a feasible flow f satisfying $l(e) \leq f(e) \leq u(e)$ if and only if $\rho_0(X) - \delta_u(X) \leq b(X)$ holds true for all $X \subseteq V$.*

Given an linear optimization problem, the optimal solution is related to the optimal solution of the dual linear program. More precisely, the concept of complementary slackness relating the two problems states the following:

Statement 2.1 (Complementary Slackness)**.** *If, in an optimal solution, the value of the dual variable associated with a constraint is nonzero, then that constraint must be satisfied with equality. Further, if a constraint is satisfied with strict inequality, then its corresponding dual variable must be zero.*

Definition 2.5 (Gomory-Hu tree)**.** *Let $G = (V, E)$ be an undirected graph with a capacity c_G along the edges. Let λ_{st} denote the minimum capacity of an s-t cut in G for each pairs $s, t \in V$. Then we say that a tree $T = (V, A)$ with a capacity c_T is the Gomory-HU tree of G, if for each $s, t \in V$ the followings hold:*

- *$\lambda_{st} = \min_{e \in P_{st}} c(e)$, where P_{st} is the s-t path in T*
- *$c_G(S_e, T_e) = c_T(S_e, T_e)$, where S_e, T_e are the two connected components of $T - e$*

Theorem 2.2 (Gomory-Hu tree)**.** *For an arbitrary undirected graph G, there exists a Gomory-Hu tree of G and it can be constructed in $|V| - 1$ max-flow computations.*

Parametric search is frequently used for solving optimization problems, whose technique invented by Megiddo [16]. This is a method to transforming a decision algorithm (that decides that the optimal solution less, equal or greater than a given number) into an optimization algorithm (that computes the optimal value). Its basic idea is to simulate the decision algorithm as if it is running with the optimal solution value λ^*. The optimal value is unknown, but each of comparisons or tests needs to be simulated during the simulation. To make each comparison, the parametric search applies a second algorithm, that is the decision algorithm with an real known number. If the simulated algorithm makes a comparison two parametric number, that are $\alpha_1 - \beta_1\lambda^*$ and $\alpha_2 - \beta_2\lambda^*$, then all that we have to do is to run decision algorithm with $\frac{\alpha_1 - \alpha_2}{\beta_1 - \beta_2}$. There are three possible case:

- 1. case, when $\beta_1 > \beta_2$: $\frac{\alpha_1-\alpha_2}{\beta_1-\beta_2} \leq \lambda^*$ if and only if $\alpha_1 - \beta_1\lambda^* \leq \alpha_2 - \beta_2\lambda^*$
- 2. case, when $\beta_1 = \beta_2$: $\alpha_1 \leq \alpha_2$ if and only if $\alpha_1 - \beta_1\lambda^* \leq \alpha_2 - \beta_2\lambda^*$
- 3. case, when $\beta_1 < \beta_2$: $\frac{\alpha_1-\alpha_2}{\beta_1-\beta_2} \geq \lambda^*$ if and only if $\alpha_1 - \beta_1\lambda^* \leq \alpha_2 - \beta_2\lambda^*$

In the end of the parametric search, we get an interval, that defines the value of the optimal solution.

3 Balanced network problem

In this section, we deal with balanced network flow problem. Then we describe parametric flow problem and its dual, after that we present Scutellà's algorithm.

Let a graph $G = (V, E)$ be a directed graph. Let $b = V \to \mathbb{R}$ be a flow demand function associated with node and $c(e)$ be a non-negative weight for unit of flow associated with arc $e \in E$. We would like to find a feasible flow, such that satisfies the demand condition and the difference between the maximum and the minimum weighted flow along single arcs ($\max_{e\in E} c(e)x(e) - \min_{e\in E} c(e)x(e)$) be minimized. We call this problem balanced network flow problem. A strongly polynomial algorithm is showed for solve this problem by Maria Grazia Scutellà and Bettina Klinz [12].
As a motivation for the problem, assume to have a network: we want to find a flow in such a way to satisfy all the demands and to distribute the network as "equitable" as possible. We want to avoid arcs too much void and arcs too much charged.
We use $u(e)$ to denote the reciprocal of $c(e)$, i.e. $u(e) = \frac{1}{c(e)}$.
By introducing $z = \max_E\{x(e)\}$ and $y = \min_E\{x(e)\}$ notations, the problem can be formulated as follows:

$$\min(z - y) \tag{1}$$

$$\sum_{e\in\rho(v)} x(e) - \sum_{e\in\delta(v)} x(e) = b(X) \qquad \forall v \in V \tag{2}$$

$$x(e) \le u(e)z \qquad \forall e \in E \tag{3}$$

$$x(e) \ge u(e)y \qquad \forall e \in E \tag{4}$$

$$x(e) \ge 0 \qquad \forall e \in E \tag{5}$$

Now we give some extra notations:

$$\lambda = (Z - y)$$

$$k(v) = \sum_{e\in\rho(v)} u(e) - \sum_{e\in\delta(v)} u(e) \qquad \forall v \in V$$

$$g(e) = x(e) - u(e)y \qquad \forall e \in E$$

Fixing value of y, we get a parametric flow problem in a parametric graph G_y. G_y is obtained from G, where parametric flow demands $b(v) - k(v)y$ are now imposed on each node. In this problem λ denotes the maximum flow value along single arcs.

Parametric flow problem is obtained by the following LP formulation:

$$\min \lambda \tag{6}$$

$$\sum_{e \in \rho(v)} g(e) - \sum_{e \in \delta(v)} g(e) = b(v) - k(v)y \qquad \forall v \in V \tag{7}$$

$$g(e) \geq 0 \qquad \forall e \in E \tag{8}$$

$$g(e) \leq u(e)\lambda \qquad \forall e \in E \tag{9}$$

3.1 Uniform Balanced Network flow problem

Hereinafter, we restrict our attention to the uniform case, where $c \equiv 1$. This implies that $u \equiv 1$ and we get the next simplified LP-formulation:

$$\min \lambda \tag{10}$$

$$\sum_{e \in \rho(v)} g(e) - \sum_{e \in \delta(v)} g(e) = b(v) - k(v)y \qquad \forall v \in V \tag{11}$$

$$g(e) \geq 0 \qquad \forall e \in E \tag{12}$$

$$g(e) \leq \lambda \qquad \forall e \in E \tag{13}$$

Where $k(v) = |\rho(v)| - |\delta(v)|$.

We note that being the uniform balanced network flow problem a linear problem with $(0, 1, -1)$-constraint matrix, it can be solved in strongly polynomial time by Tardos algorithm [19].

For compute its optimal solution, let us exploit the dual version of the parametric flow problem:

$$\max \sum_{v \in V} y(v)(b(v) - k(v)y) \tag{14}$$

$$y(v) - y(u) - y(uv) \leq 0 \qquad \forall uv \in E \tag{15}$$

$$\sum_{E} y(uv) = 1 \tag{16}$$

$$y(uv) \geq 0 \qquad \forall uv \in E \tag{17}$$

We can rewrite this problem as follows:

$$\max \frac{\sum_{v \in V} y(v)(b(v) - k(v)y)}{\sum_{uv \in E} y(uv)} \tag{18}$$

$$y(v) - y(u) - y(uv) \leq 0 \qquad \forall uv \in E \tag{19}$$

$$y(uv) \geq 0 \qquad \forall uv \in E \tag{20}$$

The optimal solutions of (14)-(17) are equivalent to the optimal solutions of (18)-(20). Let y^* be an optimal solution of (14)-(17), then it is a feasible solution of the rewritten dual problem, whose objective function value is equal to the dual optimum. (Because of $\sum_E y(uv) = 1$.)
Let y' be an optimal solution of (18)-(20), then $\sum_{uv \in E} y'(uv) > 0$.
If $\sum_{uv \in E} y'(uv) = 0$ hold, then there would be a node set X in G, such that $\rho(X) = 0$ and $\sum_{v \in X} y(v)(b(v) - k(v)y) \geq 0$. If $\sum_{v \in X} y(v)(b(v) - k(v)y) > 0$, then the parametric problem would be infeasible by Hoffman theorem.
Otherwise the objective value of y' is 0. Then there is a solution of (14)-(17), whose objective value is 0. This means that there is a trivial feasible flow in G, where the flow value is same on every arc. ($g \equiv 0$ is an optimal solution.) In this case $b(v) = yk(v)$ must hold for each $v \in V$.
Therefore, if $b(v) \neq yk(v)$ and parametric problem is feasible, then $\sum_{uv \in E} y'(uv) > 0$. In this case, $y = \frac{1}{\sum_E y'(uv)} y'$ is a feasible solution of (14)-(17) and its objective value is equal to the value of y'. Thus we have that the optimal solutions of (14)-(17) are equivalent to the optimal solutions of (18)-(20). Solving the rewritten dual problem, we get the optimal value of (10)-(13).

3.2 Maximum mean cut problem

Let us consider the rewritten dual problem, which is formulated by (18)-(20). If we write this LP problem in matrix form, we get a TU matrix. This implies that there is optimal solution, that has only $-1, 0$ or 1 entries.

If we can write $\sum_{e \in \rho(v)} g(e) - \sum_{e \in \delta(v)} g(e) \leq b(v) - k(v)y \quad \forall v \in V$ condition instead (11), then we get the same problem, because $\sum_V b(v) - k(v)y = 0$. Thus we can assume that $y(v) \geq 0$. This implies that there is optimal solution, that has only 0 or 1 entries. Then it defines a node set X and an arc set L. ($v \in X \Leftrightarrow y(v) = 1$ and $e \in L \Leftrightarrow y(e) = 1$.) Because of condition 19, we have to choose $y(uv) = 1$ if $y(v) = 1$ and $y(u) = 0$. So $y(e) = 1$ for all $e \in \rho(X)$ and we want that $|L|$ is as small as possible. Thus we can assume that $L = \rho(X)$. This means that the rewritten dual problem is equivalent to find a cut X in G, that maximize $\frac{\sum_{i \in X}(b(i) - k(i)y_i)}{\rho(X)}$. We call this problem maximum mean cut problem. This problem can be solved in $\mathcal{O}(mn^3)$ time by McCormick [15].

We remark that choosing $\lambda = \max\{\frac{\sum_{i \in X}(b(i) - k(i)y_i}{\rho(X)}\}$ there is a feasible flow, that satisfies all demands, because
$\sum_{i \in X}(b(i) - yk(i)) \leq \rho(X)\lambda \Leftrightarrow b(X) \leq \rho(X)(\lambda + y) - \delta(X)y$.

For solving the maximum mean cut problem, first we construct an auxiliary graph G_λ from G. Let P be the set of nodes, such that $b(v) \leq k(v)y$. Let N be the set of nodes, such that $b(v) > k(v)y$.

$P = \{v \in V \mid b(v) \leq k(v)y\}$
$N = \{v \in V \mid b(v) > k(v)y\}$
We define an arc capacity c in G_λ. Every edge $e \in E$ let $c(e)$ be λ. Then two extra nodes s, t are added to G_λ.
Edge $s \to v$ is added to G_λ with capacity $-(b(i) - k(i)y_i)$ for each $v \in P$.
Edge $v \to t$ is added to G_λ with capacity $(b(i) - k(i)y_i)$ for each $v \in N$.
Let X be an s-t cut in G_λ, let us see its capacity:

$$\sum_{v \in P-X}(-(b(v) - k(v)y)) + \sum_{i \in N \cap X}(b(v) - k(v)y) + \delta_G(X)\lambda =$$
$$= \sum_{v \in N}(b(v) - k(v)y) + \sum_{v \in P-X}(-(b(v) - k(v)y)) - \sum_{v \in N-X}(b(v) - k(v)y) + \delta_G(X)\lambda =$$
$$= \sum_{v \in N}(b(v) - k(v)y) - \sum_{v \in P \cap \overline{X}}(b(v) - k(v)y) - \sum_{v \in N \cap \overline{X}}(b(v) - k(v)y) + \rho_G(\overline{X})\lambda =$$
$$= \sum_{v \in N}(b(v) - k(v)y) - \sum_{v \in \overline{X}}(b(v) - k(v)y) + \rho_G(\overline{X})\lambda$$

Since $\sum_{i \in N}(b(i) - k(i)y)$ is constant for a fix y, X is a minimum cut in G' if it is maximize $\sum_{v \in \overline{X}}(b(v) - k(v)y) - \rho(\overline{X})\lambda$ for an λ.
First we guess the value of λ, let its initial value be 0. Then we compute minimum s-t cut in G_λ.
If its capacity is equal to $\sum_{v \in N}(b(v) - k(v)y)$, then the actual λ is satisfies $\sum_{i \in X}(b(i) - yk(i)) \leq \rho(X)\lambda$ for all $X \subset V$. So λ is a right upper bound for flow values along arcs in the parametric flow problem with parameter y.
If its capacity is less than $\sum_{v \in N}(b(v) - k(v)y)$, then $\sum_{v \in X}(b(v) - k(v)y) > \rho(X)\lambda$ holds for minimum cut X. Thus there is no flow in the parametric problem, such that $g(e) \leq \lambda$ for each $e \in E$. So λ must be increased for finding a feasible flow. In this case we compute the next guess of $\lambda' = \frac{\sum_{\overline{X}}(b(v) - k(v)y)}{\rho(\overline{X})}$). We mention that the right λ can't be less than λ', because $\sum_{v \in X}(b(v) - k(v)y) \leq \rho(X)\lambda$ must hold. Then we repeat this procedure for λ', until we find a minimum cut with capacity $\sum_{v \in N}(b(v) - k(v)y)$. This last found cut is the optimal solution of the maximum mean cut problem.

Algorithm 1 Max mean cut(G, y)

$\lambda = 0$
construct G_λ
compute minimum s'-t' cut $\to X$
while $c(\delta(X)) < \sum_N (b(v) - k(v)y)$ **do**
    $\lambda = \frac{\sum_{\overline{X}}(b(v)-k(v)y)}{\rho(\overline{X})}$)
    construct G_λ
    compute minimum s'-t' cut in $G_\lambda \to X$
RETURN λ, X

Let X_j be the minimum cut that the algorithm find at the j-th iteration. Then $\rho(X_{j+1}) < \rho(X_j)$ holds, otherwise:

$$
\begin{aligned}
&bX_{j+1} - kX_{j+1}y - \rho(X_{j+1})\lambda_j \leq bX_j - kX_jy - \rho(X_j)\lambda_j \\
&\frac{b(X_{j+1}) - k(X_{j+1})y}{\rho(X_{j+1})} - \lambda_j \leq \frac{b(X_j) - k(X_j)y - \rho(X_j)\lambda_j}{\rho(X_{j+1})} \\
&\frac{b(X_{j+1}) - k(X_{j+1})y}{\rho(X_{j+1})} \leq \frac{b(X_j) - k(X_j)y - \rho(X_j)\lambda_j}{\rho(X_{j+1})} + \lambda_j \leq \\
&\leq \frac{b(X_j) - k(X_j)y - \rho(X_j)\lambda_j}{\rho(X_j)} + \lambda_j = \lambda_{j+1}
\end{aligned}
$$

This is a contradiction, because if the algorithm doesn't stop at the $j+1$-th iteration, then X_{j+1} is a cut, such that $\sum_{X_{j+1}}(b(v) - k(v)y) > \rho(X_{j+1})\lambda_{j+1}$. Therefore size of $\rho(X_j)$ is decrease at every repetition. This implies the following lemma:

Lemma 3.1. *The algorithm 1 does at most $|E|$ iterations. Therefore this algorithm finds the maximum mean cut at most $|E|$ minimum s-t cut computations. Thus its total running time is $\mathcal{O}(mn^3)$.*

3.3 Newton's approach for uniform balanced network flow problem

We consider the classical Newton's method for find the optimal balanced flow in G. In the sequel we will use the following notations:
Let $h(y)$ be the optimal value of λ in the parametric flow problem corresponding to parameter y.
Let $X_{ST} \in \{0,1\}^{|V|+|E|}$ denote the family of incidence vector of cuts (S,T), where $\forall v \in V\ x_v = 1 \Leftrightarrow i \in T$ and $\forall uv \in E\ x_{uv} = 1 \Leftrightarrow u \in S, u \in T$ holds true for all $x \in X_{ST}$.
Let b, k denote $|V| + |E|$-length vectors, such that $\forall v \in V\ b_v = b(v), k_v = k(v)$ and $\forall uv \in E\ b_{uv} = k_{uv} = 0$.

Let $u \in \{0,1\}^{|V|+|E|}$ be the vector, such that $\forall v \in V\ u_v = 0$ and $\forall (uv) \in E\ u_{uv} = 1$. Using the notation $h(y)$, what we have thus to do is to minimize function h(y) for solve uniform balanced network flow problem. Function h is a piece-wise linear, convex function, that Newton's approach finds its minimum point y^*. By extending Radzik's analysis of Newton's approach, it is shown that, the method minimizes function h(y) in a strongly polynomial number of iterations [18]. It needs at most $\mathcal{O}(n \log^3(n))$ maximum mean cut computations.

Using above notations, function $h(y)$ can be rewritten as follows:

$$h(y) = \max\{\frac{bx}{ux} - \frac{kx}{ux}y \mid x \in X_{ST}\}$$

Let describe Newton's method for find minimum point y^* of h:
Firstly, let $Y = [y_1, y_2]$ be an interval, that contains optimal y^*. If $y_1 = y_2$, then we stop and the optimal solution is y_1. If $y_1 \neq y_2$, then we compute and intersect the tangents to h(y) at end points of Y and we denote its intersection by y_3. There are three possible cases:
1. the slope of tangent at y_3 is positive: we replace Y by $[y_1, y_3]$ and we repeat the process.
2. the slope of tangent at y_3 is negative: we repeat the process with the interval $Y = [y_3, y_2]$.
3. the slope of tangent at y_3 is zero: in this case y_3 is the minimum point of h. We stop, because we find the optimal solution.
We repeat this process until we find the minimum point of h.
At the beginning of the method, $y_1 = 0$ is a good choose to initialization. The value of y_2 can be chosen as follows: we take an arbitrary feasible flow x in G, such that satisfy all the demand. Then we compute $y_2 = \max_E\{x(e)\} - \min_E\{x(e)\}$.

The convergence of this algorithm isn't worse than Newton's method convergence when $h(y)$ is an increasing function, and we want to find its root.
Let $\overline{h}$ be the function obtained from h by translating the y-axis such that $\overline{h}(y^*) = 0$. Let us see the increasing lines composing $\overline{h}$, that are the lines having a positive slope:
$\overline{h}(y) = \max\{\frac{bx}{ux} - \frac{kx}{ux}y - y^* \mid x \in X_{ST}\}$
We note that the translation only effect to the constant component of b. Let examine the convergence of Newton's method to find root of $\overline{h}$. Let y_i be the parameter in the beginning of the i-th iteration.
During the algorithm, first we compute $h_i = h(y_i)$. If $h_i > 0$, then the next approximation parameter y_{i+1} is computed, that is $y_{i+1} = \frac{bx_i}{kx_i}$
Let s_i denote the slope in the i-th iteration, such that $s_i = -\frac{kx_i}{ux_i}$.
By using these notations, we have the following lemma:

Lemma 3.2. $y_{i+1} - y_i = -\frac{h_i}{s_i}$

Proof.

$$h_i = \frac{bx_i}{ux_i} - \frac{kx_i}{ux_i}y_i$$
$$y_{i+1} - y_i = \frac{bx_i}{kx_i} - (\frac{bx_i}{ux_i} - h_i)\frac{ux_i}{kx_i} = h_i\frac{ux_i}{kx_i} = -\frac{h_i}{s_i}$$

□

Lemma 3.3. $\frac{h_{i+1}}{h_i} + \frac{s_{i+1}}{s_i} \leq 1$

Proof. Let X_i be the incidence vector of the cut, that is found in the i-th iteration.

$$h_i = \frac{bx_i}{ux_i} - \frac{kx_i}{ux_i}y_i \geq \frac{bx_{i+1}}{ux_{i+1}} - \frac{kx_{i+1}}{ux_{i+1}}y_i = \frac{bx_{i+1}}{ux_{i+1}} - \frac{kx_{i+1}}{ux_{i+1}}y_{i+1} + (y_{i+1} - y_i)\frac{kx_{i+1}}{ux_{i+1}} =$$
$$= h_{i+1} + \frac{h_i}{s_i}s_{i+1}$$
$$1 \geq \frac{h_{i+1}}{h_i} + \frac{s_{i+1}}{s_i}$$

□

Because of h_i, s_i are always positive during the Newton's approach, we get that:

Lemma 3.4. $(\frac{h_{i+1}}{h_i})(\frac{s_{i+1}}{s_i}) \leq \frac{1}{4}$

We separately examine the iterations which decrease the slope a lot and those which not:

Lemma 3.5. *Newton's approach does at most* $\mathcal{O}(\log(n))$ *iterations, such that* $s_{i+1} \leq \frac{2}{3}s_i$.

Proof. Let us consider a sequence of consecutive iterations, that satisfy condition of lemma. Then there is at least half-length sub-sequence, where $s_{i+1} \leq \frac{4}{9}s_i$ holds for all consecutive iterations. We divide this sequence into L-length sub-sequences, where $L = \log((m+1)^2)$ $(m = |E|)$.
Then it is for i-th elements of every sub-sequence:

$$-\frac{kx_{L+i}}{ux_{L+i}} \leq -\frac{1}{(m+1)^2}\frac{kx_i}{ux_i} \quad \Leftrightarrow \quad -kx_{l+i}ux_i \leq \frac{1}{(m+1)^2}(-kx_i)ux_{L+i}$$

where $i \in \{0, 1, ..., L-1\}$ and $(-kx_i)$ is positive.

Because of ux_i is integer and $0 \leq ux_i \leq m$, we have that $ux_i \geq 1 > \frac{1}{m+1}ux_{L+i}$. This means that $-kx_{L+i} \leq \frac{1}{m+1}(-kx_i)$ must hold. Since $-k_i \in [-n+1, n-1]$, then kx_i is integer and $-kx_i \in [-n^2, n^2]$. Therefore $-kx_{L+i} \leq \frac{1}{m+1}(-kx_i)$ can be true for constant number of pairs (x_i, x_{i+L}).

Thus Newton's method does at most $\mathcal{O}(L) = \mathcal{O}(\log((m+1)^2)) = \mathcal{O}(n)$ iterations, such that $s_{i+1} \leq \frac{2}{3}s_i$. □

We will use the following lemma, whose proof is reported in Radizk's work [18]:

Lemma 3.6 (Goemans). *Let $b \in \mathbf{R}^p$ be a real vector and let $x_1, x_2, ..., x_q$ be p-length binary vectors,such that holds:*

$$0 < bx_{i+1} \leq \frac{1}{2}bx_i \quad \forall i \in \{1, 2, .., q-1\}$$

Then $q = \mathcal{O}(p \log(p))$

Lemma 3.7. *Newton's approach does at most $\mathcal{O}(n \log^2(n))$ consecutive iterations, such that $s_{i+1} > \frac{2}{3}s_i$.*

Proof. Let us indicate the sequence of iterations, such that above inequality holds: $s_1, s_2, .., s_r$. By lemma 3.3: $\frac{h_{i+1}}{h_i} \leq \frac{1}{3}$. Using lemma 3.2, we have that:

$$y_{i+2} - y_{i+1} = -\frac{h_{i+1}}{s_{i+1}} = -\frac{h_{i+1}}{s_{i+1}}\frac{s_i}{h_i}\frac{h_i}{s_i} = \frac{h_{i+1}}{h_i}\frac{s_i}{s_{i+1}}\left(-\frac{h_i}{s_i}\right) \leq \frac{1}{3}\frac{3}{2}\left(-\frac{h_i}{s_i}\right) = \frac{1}{2}(y_{i+1} - y_i)$$

The difference between the parameters of this kind of iteration reduces fast.

$$\begin{aligned} &y_{r+1} - y_{i+1} = (y_{r+1} - y_r) + (y_r - y_{r-1}) + ... + (y_{i+2} - y_{i+1}) \leq \\ &\leq \frac{1}{2}((y_r - y_{r-1}) + ... + (y_{i+1} - y_i)) = \frac{1}{2}(y_r - y_i) \leq \\ &\leq \frac{1}{2}(y_r - y_i) + \frac{1}{2}(y_{r+1} - y_r)\frac{1}{2}(y_{r+1} - y_i) \end{aligned}$$

$y_{i+1} = \frac{bx_i}{kx_i}$ implies that: $(-b + ky_{r+1})x_i = -bx_i + kx_iy_{r+1} = -y_{i+1}kx_i + kx_iy_{r+1} = (y_{r+1} - y_{i+1})kx_i$
Then for all this kind of iterations the following relation holds:

$$\begin{aligned} &\frac{(-b + ky_{r+1})x_{i+1}}{ux_{i+1}} = \frac{(y_{r+1} - y_{i+2})kx_{i+1}}{ux_{i+1}} \leq \frac{1}{2}\frac{(y_{r+1} - y_{i+1})kx_{i+1}}{ux_{i+1}} = \\ &= \frac{1}{2}(y_{r+1} - y_{i+1})\frac{kx_{i+1}}{ux_{i+1}} == \frac{1}{2}(y_{r+1} - y_{i+1})s_{i+1} \leq \\ &\leq \frac{1}{2}(y_{r+1} - y_{i+1})s_i = \frac{1}{2}(y_{r+1} - y_{i+1})\frac{kx_i}{ux_i} = \frac{1}{2}\frac{(-b + ky_{r+1})x_i}{ux_i} \end{aligned}$$

In similar way to the proof of previous lemma, we divide this sequence into L-length sub-sequences. Then we get that the following has to hold true:

$$\frac{(-b + ky_{r+1})x_{i+L}}{ux_{i+L}} \leq \frac{1}{(m+1)^2}\frac{(-b + ky_{r+1})x_i}{ux_i} \quad \forall i = 0, 1, 2..., L-1$$

Similarly, this implies that: $(-b+ky_{r+1})x_{i+L} \leq \frac{1}{m+1}(-b+ky_{r+1})x_i$ || Now we apply Goemans lemma for $b' = (-b+ky_{r+1}), x'_j = x_{i+jL} \quad j = 0, 1, 2, .., \frac{r}{L}$. This implies that $\frac{r}{L} = \backslash \log(\backslash)$. Because of $\mathcal{O}(L) = \mathcal{O}(n)$, the length of the sequence of consecutive iterations is $\mathcal{O}(n \log^2(n))$, such that $s_{i+1} > \frac{2}{3}s_i$. □

Above lemmas implies the number of iteration of the Newton's approach to find $\overline{h}$ root the following:

Theorem 3.1. *Newton's method finds the root of $\overline{h}(y)$ by performing $\mathcal{O}(n \log^3(n))$ iterations.*

As we mentioned, the number of iteration when root has to be found is greater than or equal to the number of iterations performed by the Newton' approach to find minimum point of h. In consequence, we get the following theorem:

Theorem 3.2. *Newton's method solves uniform balanced network flow problem by performing $\mathcal{O}(n \log^3(n))$ maximum mean cut computations, that is $\mathcal{O}(nm \log^3(n)) = \mathcal{O}(n^3 \log^3(n))$ max-flow computations.*

3.4 Balanced network flow problem with general weight

In this subsection, we return the general balanced network flow problem. We can solve this more general problem very similar way, than in the uniform case. The dual problem of (6)-(9) can be reformulated as follows:

$$\max \frac{\sum_{v \in V} y(v)(b(v) - k(v)y)}{\sum_{uv \in E} u(uv)y(uv)} \tag{21}$$

$$y(v) - y(u) - y(uv) \leq 0 \qquad \forall uv \in E \tag{22}$$

$$y(uv) \geq 0 \qquad \forall uv \in E \tag{23}$$

This problem is called maximum mean weighted cut problem. It is equivalent to compute the root of the next function:

$$H_y(\lambda) = \max\{bx - kxy - \lambda ux\}$$

We can compute value of $H_y(\lambda)$ for any positive λ by constructing an auxiliary graph G_λ. Let P be the set of nodes, such that $b(v) \leq k(v)y$. Let N be the set of nodes, such that $b(v) > k(v)y$. Each arc of G has an capacity $\lambda u(e)$. Let us enlarge G_λ by adding a source s and a destination t. Edge $s \to v$ is added with capacity $-(b(i) - k(i)y_i)$ for each $v \in P$ and edge $v \to t$ is added with capacity $(b(i) - k(i)y_i)$ for each $v \in P$. We can claim a statement of the minimum s-t cut of G_λ like in the uniform case. The minimum cut in G_λ is defined the value of $H_y(\lambda)$. Then all we have

to do is to find the root of H_y with a typical Newton's iteration, until reaching the required flow feasibility. The value of the next iteration can be computed starting from the maximum flow computed at the previous iteration of Newton's approach. Scutellà proved that it does at most $|E|$ iteration [12].

Then, we can define same function $h(y)$ like in the previous section and find its minimum point with Newton iteration. This is obtained an $\mathcal{O}(n^5m^3)$ time algorithm for solve balanced network flow problem. [12]

4 Balanced sub-modular flow problem

In this section we deal with balanced submodular network problem. We consider LP formulation of the parametric problem and its dual problem and we show an method to compute optimal value of the parametric submodular flow problem. Then we show a polynomial time algorithm for the balanced sub-modular flow problem based on Scutella's network flow algorithm.

Given a directed graph $G = (V, E)$, there is given a fully submodular function $b(X)$ for all $X \subseteq V$ node set. We call a flow in the graph submodular flow, if $\sum_{e\in\rho(X)} x_e - \sum_{e\in\delta(X)} x_e \leq b(X)$ holds for all $X \subset V$. (The difference between indegree and outdegree of X is less than $b(X)$.) We would like to find a submodular network in graph G, such that the difference between the minimal and maximal amount of the network is minimal.

This problem is formulated the following way:

$$\min(\max\{x_e\} - \min\{x_e\})$$
$$\sum_{e\in\rho(X)} x_e - \sum_{e\in\delta(X)} x_e \leq b(X) \qquad \forall X \subset V$$
$$x \geq 0$$

We will use next notations:

$$y = \min_E\{x_e\}$$
$$z = \max_E\{x_e\}$$
$$\delta = z - y$$

Then there is a fundamental y-flow on all edge, where y is the minimum amount of flow that can pass through an edge. This defines a new $b'(X) = b(X) - \rho(X)y + \delta(X)y$ demand function for every $X \subseteq V$ node set.
$b'(X)$ is also fully submodular:

$$\forall X, Y \subset V:$$
$$b'(X) + b'(Y) = b(X) - \rho(X)y + \delta(X)y + b(Y) - \rho(Y)y + \delta(Y)y =$$
$$= b(X) + b(Y) - \rho(X \cup Y)y - \rho(X \cap Y)y + \delta(X \cup Y)y + \delta(X \cap Y)y \geq$$
$$\geq b(X \cup Y) + b(X \cap Y)y - \rho(X \cup Y)y + \delta(X \cup Y)y - \rho(X \cap Y)y + \delta(X \cap Y)y =$$
$$= b'(X \cup Y) + b'(X \cap Y)$$

The problem can be reformulated as follows:

$$\max -\delta \tag{24}$$

$$\sum_{e\in\rho(X)} x_e - \sum_{e\in\delta(X)} x_e \leq b'(X) \qquad \forall X \subset V \tag{25}$$

$$x \leq \delta \tag{26}$$

$$x \geq 0 \tag{27}$$

4.1 Dual problem

Let us consider the dual problem of the rewritten balanced submodular network flow problem:

$$\min \sum_{Z\subset V} y_Z b'(Z)$$

$$\sum_{e\in\rho(Z)} y_Z - \sum_{e\in\delta(Z)} y_Z + y_e \geq 0 \qquad \forall e \in E$$

$$\sum_{e\in E} -y_e \geq -1$$

$$y_e, y_Z \geq 0 \qquad \forall e \in E, Z \subset V$$

$y \equiv 0$ is a feasible solution of this problem, for this the optimum isn't positive.

Dual optimal solutions are equivalent to optimal solutions of the following modified problem:

$$\min \frac{\sum_{Z\subset V} y_Z b'(Z)}{\sum_{e\in E} y_e}$$

$$y_e \geq \sum_{e\in\delta(Z)} y_Z - \sum_{e\in\rho(Z)} y_Z \qquad \forall e \in E$$

$$\sum_{e\in E} y_e > 0$$

$$y_e, y_Z \geq 0 \qquad \forall e \in E, Z \subset V$$

We denote the dual optimal solution by y^*. Then we make a feasible solution of the modified problem, whose value of the objective function is equal to the dual optimum.

As $\sum_{e\in E} y_e^* = 0$ holds for y^*, then increasing the value of y_e^* on an arbitrary edge e by 1 we get an same-valued feasible solution. This y^* is feasible solution of the modified problem.

If $0 < \sum_{e \in E} y_e^* \leq 1$ holds for y^*, then we can make a new feasible solutions by dividing vector y^* by $\sum_{e \in E} y_e^*$. The objective function value of this new solution isn't greater than original y^*. Because of y^* is an optimal solution, the new solution is also optimal. So we get a solutions of the modified problem, whose objective function value is equal to the optimum of the original problem. (Since $\sum_{e \in E} y_e = 1$.)

Let us see the other direction. We use y' to denote an optimal solution of the modified problem. Then we make a feasible solution of the original dual problem, whose value of the objective function is equal to the modified optimum.
Vector y' is multiplied by $\frac{1}{\sum_{e \in E} y_e}$, than we get the feasible solution of both problem and their objective values are equals.

By following the above, we assume that $\sum_{e \in E} y_e^* = 0$ or $\sum_{e \in E} y_e^* = 1$ holds for the dual optimal solution. If $0 < \sum_{e \in E} y_e^* < 1$, then dividing it by $\sum_{e \in E} y_e^*$ we get a not worse solution, such that $\sum_{e \in E} y_e^* = 1$.

If $\sum_{e \in E} y_e^* = 0$ and $y^* b < 0$, then the dual problem is unbounded. In this case, we can choose some node sets $S_1, S_2, ..., S_k \subset V$, that their total weight is negative ($\sum_{i=1}^{k} b'(S_i) < 0$) and $\forall e \in E$-re $y_e^* = 0$. Having multiplied y^* by $c > 0$ arbitrary, positive scalar, we get also an feasible solution, whose objective value is c times the value of original y^*. This way we can make an dual feasible solutions with arbitrary small objective value.
In this case the primal must be infeasible, that is to say there is no submodular network in G with b demand function. Let us see the union of above-mentioned node sets. There is no outgoing edge of the union set, otherwise $y_e^* > 0$ holds for that edge, that is impossible because of condition $\sum_{e \in E} y_e^* = 0$.

If $\sum_{e \in E} y_e^* = 0$ and $y^* b = 0$, then there is no $Z \subset V$, such that $b'(Z) < 0$, otherwise selecting it gives a solution with negative objective value. (This can't happen, because the optimal value is 0.) Therefore $b(Z)' \geq 0 \ \forall Z \subset V$, thus $x \equiv 0, \delta = 0$ is primal feasible solution of the balanced submodular flow problem.

Now let us consider the modified dual problem, whose optimal solutions are equivalent to original dual optimal solutions. This problem can be reformulated as follows: we choose some node sets. The set of nodes Z is selected with weight y_Z. Then we compute $y_e = (\sum_{e \in \delta(Z)} y_Z - \sum_{e \in \rho(Z)} y_Z)^+$ value for all $e \in E$, where $(x)^+$ denotes the positive part of x. If $\sum_{e \in E} y_e = 0$, then due to the above reasons the primal problem is infeasible or has a trivial solution.

Suppose that the primal problem has an non-trivial solution, there is $Z \subset V$, such that $b'(Z) < 0$ and for all $\forall Z' \subset V$ at least one of the following holds: $b'(Z') \geq 0$ or $\delta(Z') > 0$. We would like to find a family of sets and weight y_z, which minimizes $\min \frac{\sum_{Z \subset V} y_Z b'(Z)}{\sum_{e \in E} y_e}$.
Let us see the selected family of node sets:
If there are two intersecting sets X, Y, such that $y_X, y_Y > 0$ hold, then we can re-

place them by their intersection and union. It follows from b' fully submodularity:
$b'(X) + b'(Y) \geq b'(X \cup Y) + b'(X \cap Y)$ and $\delta(X) + \delta(Y) \geq \delta(X \cap Y) + \delta(X \cup Y)$
Thus weights $y_{X \cup Y}, y_{X \cap Y}$ are increased by $\min\{y_X, y_Y\}$ and weights y_X, y_Y are decreased by the same value. We get a feasible solution and its objective function value isn't increased.
We can assume that the family of selected sets $\mathcal{F}$ is laminar. (We say that Z is selected, if $y_Z > 0$.)

If there are two selected, disjoint sets X, Y, then we can replace them by their union:
$b'(X) + b'(Y) \geq b'(X \cap Y) + b'(\emptyset)$
$b'(\emptyset) > 0$, otherwise having selected it with arbitrary big weight, we get an arbitrary small dual solution. (Then the dual problem is unbounded.)
Thus we increase weights $y_{X \cup Y}, y_{X \cap Y}$ by $\min\{y_X, y_Y\}$ and decrease weights y_X, y_Y by the same amount. We get a feasible solution and its objective function value isn't increased.
Thus we can assume that the family of selected sets $\mathcal{F}$ doesn't contain two disjoint sets. Since we assumed that the family is laminar, for every two sets of $\mathcal{F}$ one contains the other. ($\forall X, Y \in \mathcal{F} : X \subset Y$ or $Y \subset X$.)

For all $e \in E$, y_e is equal to number of selected sets of which e is an outgoing edge. There is no edge, such that leaves an selected set and enters another one.
That is $\sum_{e \in E} y_e = \sum_{Z \in \mathcal{Z}} \delta(Z)$.
Let us $X, Y \in \mathcal{F}$ be two selected sets, $Y \subset X$ and we want to proved that one of the following holds:

$$\frac{b(X)y_X + b(Y)y_Y}{\delta(X)y_X + \delta(Y)y_Y} \geq \frac{b(X)y_X}{\delta(X)y_X}$$

or

$$\frac{b(X)y_X + b(Y)y_Y}{\delta(X)y_X + \delta(Y)y_Y} \geq \frac{b(Y)y_Y}{\delta(Y)y_Y}$$

We will prove it by contradiction. We suppose that the above proposition isn't true, thus next two statements hold:

$$\frac{b(X)y_X + b(Y)y_Y}{\delta(X)y_X + \delta(Y)y_Y} < \frac{b(X)y_X}{\delta(X)y_X}$$
$$\frac{b(X)y_X + b(Y)y_Y}{\delta(X)y_X + \delta(Y)y_Y} < \frac{b(Y)y_Y}{\delta(Y)y_Y}$$

Multiplying equations by $\delta(X)y_X$ and $\delta(Y)y_Y$, we get:

$$b(Y)y_Y\delta(X)y_X < b(X)y_X\delta(Y)y_Y$$
$$b(Y)y_Y\delta(X)y_X > b(X)y_X\delta(Y)y_Y$$

This is an contradiction and this shows that the original proposition must be true. This means that one of two above inequality holds.

Then let Z_1 be an arbitrary member of $\mathcal{F}$. Due to the above proposition, one of the following holds:

$$\frac{\sum_{Z\in\mathcal{F}} b(Z)y_Z}{\sum_{Z\in\mathcal{F}} \delta(Z)y_Z} \geq \frac{b(Z_1)y_{Z_1}}{\delta(Z_1)y_{Z_1}}$$

or

$$\frac{\sum_{Z\in\mathcal{F}} b(Z)y_Z}{\sum_{Z\in\mathcal{F}} \delta(Z)y_Z} \geq \frac{\sum_{Z\in\mathcal{F},Z\neq Z_1} b(Z)y_Z}{\sum_{Z\in\mathcal{F},Z\neq Z_1} \delta(Z)y_Z}$$

This way $\mathcal{F}$ set-family can be replaced by a family, whose size is smaller and objective value isn't greater. By repeating this replacing method, we get an 1-element family.

Therefore, the family of sets can be replaced by its one member, for example by its minimal mean set. Let us Z' be the set, which minimize $\min_{Z\in\mathcal{F}} \frac{b'(Z)}{\delta(Z)}$. By taking this set Z' with weight $y_{Z'}$ instead of family, we also get a feasible solution and its objective value doesn't increase.
In this case the value of weight y_Z doesn't matter, because it doesn't change the value of the objective function: $\frac{b'(Z)y_Z}{\delta(Z)y_Z} = \frac{b'(Z)}{\delta(Z)}$.

Now we can assume that the family of sets in the optimal solution contains exactly one set Z, such that $y_Z = 1$ and $y_e = 1$ for all $e \in \delta(Z)$. This means that we have to find a cut Z, which minimize $\frac{b'(Z)}{\delta(Z)}$. This problem is called minimum mean cut problem. In our case weight function b' is submodular. In the next subsection we show an method for solve this problem in polynomial time.

4.2 Minimum mean cut problem with submodular functions

Let us given a graph $G = (V, E)$ and a weight function $b(S)$ for all subset $S \subseteq V$. The minimum mean cut problem is to find a cut S, that minimizes $\frac{b(S)}{\delta(S)}$.

First we take $\lambda_1 = 0$,$b_1(Z) = b(Z)$ and we minimize the fully submodular function b_1. We denote by Z_1 the minimal set.

Giving the solution of the $i-1$-th iteration (Z_{i-1}), we will use the following notations in the i-th iteration: $\lambda_i = \frac{b(Z_{i-1})}{\delta(Z_{i-1})}$ and $b_i(Z) = b(Z) - \lambda_i\delta(Z)$.

Algorithm 2 Minimum mean cut with submodular function(G, b)

$\lambda_1 = 0$, $b_1(Z) = b(Z)$, $i = 1$
Minimize submodular function $b_1 \to Z_1$
if $b(Z_1) \geq 0$ **then**
    RETURN λ_1,Z_1
if $\delta(Z_i) = 0$ **then**
    RETURN "Primal is infeasible"
while $b(Z_i) < 0$ **do**
    $i = i + 1$
    $\lambda_i = \frac{b(Z_{i-1})}{\delta(Z_{i-1})}$
    $b_i(Z) = b(Z) - \lambda_i \delta(Z)$
    Minimize submodular function $b_i \to Z_i$
    if $\delta(Z_i) = 0$ **then**
        RETURN "Primal is infeasible"
RETURN λ_i,Z_i

Let us see that b_i is fully submodular:

$$b_i(X)+b_i(Y) = b(X)-\lambda_i\delta(X)+b(Y)-\lambda_i\delta(Y) \geq b(X\cap Y)+b'(X\cup Y)-\lambda_i(\delta(X)+\delta(Y))$$
$$\geq b(X \cap Y) + b(X \cup Y) - \lambda_i(\delta(X \cap Y) + \delta(X \cup Y)) =$$
$$b(X \cap Y) - \lambda_i\delta(X \cap Y) + b(X \cup Y) - \lambda_i\delta(X \cup Y) = b_i(X \cap Y) + b_i(X \cup Y)$$

Indeed, this holds true, since $\lambda_i \leq 0$ and $\delta(X \cap Y) + \delta(X \cup Y) \leq \delta(X) + \delta(Y)$.

We repeat this iterative computation, until the weight of the minimal cut become non-negative. (If the last iteration is the i-th,then $b_i(Z_i) \geq 0$ and $b_{i-1}(Z_{i-1}) < 0$.) Let λ be equal to λ_i, that we used in the last iteration. For all $Z \subset V$:
$\frac{b(Z)}{\delta(Z)} \geq \lambda$, since $b_i(Z) = b(Z) - \lambda\delta(Z) \geq b(Z_i) - \lambda\delta(Z_i) \geq 0$
Then λ is the optimum value of the problem and the final Z_i is the optimal set.

If on any iteration we find a set Z_i, such that $\delta(Z_i) = 0$, then the algorithm is stopped. If we find it in the first iteration ($i = 1$) and $b(Z_i) \geq 0$, then b is non-negative and there exist a trivial feasible solution for the balanced submodular flow problem. Otherwise, the primal problem can't be feasible, since $b'(Z_i) < 0$ and $\delta(Z_i) = 0$.

The algorithm does at most $|E|$ repetition, because $\delta(Z_i)$ is decreased at every iteration. ($\delta(Z_{i+1}) < \delta(Z_i)$)
Due to definitions of λ_i, Z_i and $\lambda_{i-1} > \lambda_i$:
$b(Z_i) - \lambda_{i-1}\delta(Z_i) \leq b(Z_{i+1}) - \lambda_{i-1}\delta(Z_{i+1})$
We can prove it by contradiction:
$\delta(Z_i) \leq \delta(Z_{i+1}) \to -(\lambda_i - \lambda_{i+1})\delta(Z_i) \leq -(\lambda_i - \lambda_{i+1})\delta(Z_{i+1})$

$b(Z_i) - \lambda_{i-1}\delta(Z_i) - (\lambda_i - \lambda_{i+1})\delta(Z_i) \leq b(Z_{i+1}) - \lambda_{i-1}\delta(Z_{i+1}) - (\lambda_i - \lambda_{i+1})\delta(Z_{i+1})$
$b(Z_i) - \lambda_i\delta(Z_i) \leq b(Z_{i+1}) - \lambda_i\delta(Z_{i+1})$

This can't be happened, because of the definition of Z_{i+1}:
$b(Z_i) - \lambda_i\delta(Z_i) \geq b(Z_{i+1}) - \lambda_i\delta(Z_{i+1})$

At each iteration the value of λ is always decreased. If there are at least two repetition, then $b'(Z_i) < 0$ holds true for all found Z_i (except in the last iteration).

If there is a set Z in the graph, such that $b'(Z) < 0$ and $\delta(Z) = 0$, then it is found during the algorithm. Let us see sets with the property that $\delta(Z') > 0$ holds, whose function values are increased at every iteration. After some repetition, sets, such that $\delta(Z) = 0$, can stay negative.

4.3 Newton's approach for balanced submodular network flow problem

We remind of the reformulated balanced submodular network flow problem:

$$\begin{aligned} &\max -\delta \\ \sum_{e\in\rho(X)} x_e - \sum_{e\in\delta(X)} x_e &\leq b'(X) \qquad\qquad \forall X \subset V \\ x &\leq \delta \\ x &\geq 0 \end{aligned}$$

If we fix parameter y in the above formulation, then we obtained a parametric submodular flow problem. Let us $h(y)$ denote its optimal value corresponding to y. Then the optimal value of the original problem is equal to the minimum of $h(y)$. So we have to minimize function $h(y)$.

As we proved, its dual problem is equivalent to the minimum mean cut problem with a submodular function. In the previous section, we saw a polynomial algorithm for solving the dual problem. This means that we can compute $h(y)$ in polynomial time.

$h(y) = \min\left(\frac{b_y(S)}{\delta(S)} \mid S \subset V\right)$, where $b_y(S) = b(S) - \rho(S)y + \delta(S)y$.
Then $h(y)$ is a convex, piece-wise linear function. We use Newton's approach to find the minimum point y^* of function h.
First we started from an internal $[y_1, y_2]$, such that contains y^*. We compute the tangents at endpoints of the internal and we take their intersection of their tangents. This point y' splits the internal. If the slope of tangent at y' is negative, we repeat the process with the internal $[y', y_2]$ in the same way. If the slope of tangent at y' is positive, we continue with the internal $[y_1, y']$. The convergence of this algorithm isn't worse as the convergence the following Newton's method: $\overline{h}(y)$ is a decreasing

function and we want to find its root. Let $\overline{h}$ be the function obtained from h by translating the y-axis such that $\overline{h}(y^*) = 0$:
$\overline{h}(y) = h(y) - h(y^*) = min\big(\frac{b_y(S)}{\delta(S)} \mid S \subset V\big) - h(y^*) =$
$= \big(\frac{b(S)-h(y^*)\delta(S)-\rho(S)y+\delta(S)y}{\delta(S)} \mid S \subset V\big)$
This means that it is enough to examine the convergence the Newton's method for $\overline{h}(y)$.

First we choose an initial guess y_0, which is certainly less than the root of $\overline{h}$. $y_0 = 0$ is a good choose for this. Then we compute $\overline{h}(y_0)$ and the corresponding cut. If $\overline{h}(y_0) > 0$, then we compute the next value of parameter y. Let us see a general iteration:
Let y_i be the actual guess at the beginning of the i-th iteration. Then we run the minimum mean cut method with submodular function and we get $h_i = \overline{h}(y_i)$ with corresponding cut Z_i. If $\overline{h}(y_i) = 0$, we stop, because we find the root of the function. If $\overline{h}(y_i) > 0$, we compute the next parameter approximation y_{i+1}. Let m_i denote the slope of the line at the i-th iteration.

$$y_{i+1} = y_i - \frac{\overline{h}(y_i)}{\overline{h}'(y_i)} = -\frac{b(Z_i)}{\delta(Z_i) - \rho(Z_i)}$$

$$m_i = \frac{\delta(Z_i) - \rho(Z_i)}{\delta(Z_i)}$$

First let us see the Radzik's analysis of Newton's method, that proves that the number of iteration is linear in b. We will use the following lemma to prove the fast convergence of the Newton's approach:

Lemma 4.1. $\frac{h_{i+1}}{h_i} + \frac{m_{i+1}}{m_i} \leq 1$ *holds for all iteration of the method.*

Proof. Because of Z_{i+1} minimize $\frac{b_{y_{i+1}}(Z)}{\delta(Z)}$:
$h_{i+1} = \frac{b_{y_{i+1}}(Z_{i+1})}{\delta(Z_{i+1})} \leq \frac{b_{y_{i+1}}(Z_i)}{\delta(Z_i)}$
Due to the fact that $y_{i+1} - y_i = -\frac{h_i}{m_i} > 0$ and $m_i < m_{i+1} < 0$:

$$\begin{aligned} h_{i+1} &\leq \frac{b_{y_{i+1}}(Z_i)}{\delta(Z_i)} = \frac{b(Z_i)}{\delta(Z_i)} + \frac{\delta(Z_i) - \rho(Z_i)}{\delta(Z_i)} y_{i+1} = \\ &= \frac{b(Z_i)}{\delta(Z_i)} + \frac{\delta(Z_i) - \rho(Z_i)}{\delta(Z_i)} y_i + \frac{\delta(Z_i) - \rho(Z_i)}{\delta(Z_i)} (y_{i+1} - y_i) \leq \\ &\leq h_i + m_{i+1} - \frac{h_i}{m_i} = h_i - h_i \frac{m_{i+1}}{m_i} \end{aligned}$$

□

The following inequality holds true:

Lemma 4.2. $(\frac{h_{i+1}}{h_i})(\frac{m_{i+1}}{m_i}) \leq \frac{1}{4}$

The next lemma can be proved in very similar way than lemma 3.5

Lemma 4.3. *The Newton's approach does at most $\mathcal{O}(\log(n))$ iterations, such that $|m_{i+1}| \leq \frac{2}{3}|m_i|$ holds.*

Lemma 4.4. *The Newton's approach does at most $\mathcal{O}\Big(\log_2(n)+\log_2\big(\max\{b(Z)\}\big)\Big)$ consequence iterations, such that $|m_{i+1}| \geq \frac{2}{3}|m_i|$*

Proof. Let us see the sequence of iterations, such that above inequality holds: $m_1, m_2, .., m_r$ (after reindexing). By lemma 4.2 ($\frac{h_{i+1}}{h_i} + \frac{m_{i+1}}{m_i} \leq 1$) we have that $\frac{h_{i+1}}{h_i} \leq \frac{1}{3}$.
Due to the fact that $y_{i+1} - y_i = -\frac{\overline{h}(y_i)}{\overline{h}'(y_i)} = -\frac{h_i}{m_i}$:

$$y_{i+2} - y_{i+1} = -\frac{h_{i+1}}{m_{i+1}} = \frac{h_{i+1}}{|m_{i+1}|}\frac{|m_i|}{h_i}\frac{h_i}{|m_i|} = \frac{h_{i+1}}{h_i}\frac{|m_i|}{|m_{i+1}|}\frac{h_i}{|m_i|} \leq \frac{1}{3}\frac{3}{2}\frac{h_i}{|m_i|} = \frac{1}{2}(y_{i+1} - y_i)$$

Because of $y_{i+1} = -\frac{b(Z_i)}{\delta(Z_i)-\rho(Z_i)}$:

$$y_{i+1} - y_i = -\frac{b(Z_{i+1})}{\delta(Z_{i+1}) - \rho(Z_{i+1})} + \frac{b(Z_i)}{\delta(Z_i) - \rho(Z_i)} = \\ = \frac{b(Z_{i+1})}{|\delta(Z_{i+1}) - \rho(Z_{i+1})|} - \frac{b(Z_i)}{|\delta(Z_i) - \rho(Z_i)|}$$

One of the following holds true:

$$\frac{b(Z_{i+1})}{|\delta(Z_{i+1}) - \rho(Z_{i+1})|} - \frac{b(Z_i)}{|\delta(Z_i) - \rho(Z_i)|} \geq \frac{b(Z_{i+1}) - b(Z_i)}{|\delta(Z_{i+1}) - \rho(Z_{i+1})|}$$
$$\frac{b(Z_{i+1})}{|\delta(Z_{i+1}) - \rho(Z_{i+1})|} - \frac{b(Z_i)}{|\delta(Z_i) - \rho(Z_i)|} \geq \frac{b(Z_{i+1}) - b(Z_i)}{|\delta(Z_i) - \rho(Z_i)|}$$

In both of the above case we have that:

$$y_{i+1} - y_i = \frac{b(Z_{i+1})}{|\delta(Z_{i+1}) - \rho(Z_{i+1})|} - \frac{b(Z_i)}{|\delta(Z_i) - \rho(Z_i)|} \geq \frac{b(Z_{i+1}) - b(Z_i)}{|E|}$$

As we mentioned, the difference between the parameters of this kind of iteration reduces fast:

$$y_{i+2} - y_{i+1} \leq \frac{1}{2}(y_{i+1} - y_i) \rightarrow (y_{i+k} - y_{i+k+1)} \leq \frac{1}{2^k}(y_{i+1} - y_i)$$

$y_{i+1} - y_i \geq \frac{b(Z_{i+1})-b(Z_i)}{|E|}$ implies that:

$$\frac{\min\{b(Z_{i+1}) - b(Z_i)\}}{|E|} \leq (y_{i+k} - y_{i+k+1)} \leq \frac{1}{2^k}(y_{i+1} - y_i) \leq \frac{1}{2^k}\max(b(Z))$$

We can estimate the number of iterations, such that $|m_{i+1}| \geq \frac{2}{3}|m_i|$:

$$\frac{\min\{b(Z_{i+1}) - b(Z_i)\}}{|E|} \leq \frac{1}{2^k} \max(b(Z)) \rightarrow$$

$$k \leq \log_2 \left(\frac{|E| \max(b(Z))}{\min\{b(Z_{i+1}) - b(Z_i)\}} \right) \leq \log_2(|E| \max\{b(Z)\})$$

Theorem 4.1. *Newton's method finds the root of $\overline{h}(y)$ by performing $\mathcal{O}(\log_2(n))\mathcal{O}\Big(\log_2(n) + \log_2\big(\max\{b(Z)\}\big)\Big) = \max\{\mathcal{O}(\log_2^2(n)), \mathcal{O}(\log_2(n)\max(b))\}$ iterations.*

Now we show that the Newton approach gives a strongly polynomial algorithm to balanced submodular network flow problem.

Theorem 4.2. *Newton's method finds the root of $\overline{h}(y)$ by performing $\mathcal{O}(|E|^2)$ iterations.*

Proof. The slope of the actual line is increased at every iteration and it is computed by the following equation:

$$m_i = \frac{\delta(Z_i) - \rho(Z_i)}{\delta(Z_i)}$$

This takes at most $2|E|^2$ different values, because $-m \leq \delta(Z_i) - \rho(Z_i) \leq m$, $0 \leq \rho(Z_i) \leq m$ and both of them are integer.

Since the slope is changed at each iteration, the number of iterations isn't greater than $2|E|^2$. □

Theorem 4.3. *Balanced submodular flow problem can be solved with computing $\mathcal{O}(|E|^3)$ submodular function minimization problem.*

We note that this analysis works for uniform balanced network flow problem. Thus we can prove that the algorithm in the previous section takes at most $\mathcal{O}(|E|^3)$ maximum flow computations.

The next lemmas imply that we can easily find the integer optimal submodular flow, having got a fractional optimum. We remind that the parametric submodular flow problem is formulated by (24)-(24). We use $h_I(y)$ to denote the optimal integer solution of (24)-(24) for any integer $y \in \mathbb{Z}_{\geq 0}$.

Lemma 4.5. *For an arbitrary integer $y \in \mathbb{Z}_{\geq 0}$, the optimal integer solution of the parametric problem corresponding to y is equal to $\lceil h(y) \rceil$, i.e. $h_I(y) = \lceil h(y) \rceil$*

Proof. $h(y)$ is denoted the smallest number, such that there exist a feasible submodular flow in G and $y \leq x(e) \leq h(y)$ holds for all $e \in E$. Due to this definition, it is impossible to exist an integer flow, such that $y \leq x(e) \leq \lceil h(y) \rceil - 1$ holds true.
However, there is a fractional flow with $\lceil h(y) \rceil$ upper bounds as well as y and $\lceil h(y) \rceil$

are integer. Then there is an integer flow in G, such that $y \leq x(e) \leq \lceil h(y) \rceil$. Then above two facts imply that $h_I(y) = \lceil h(y) \rceil$. □

By the definition of h and h_I, it is easy to see the following lemma:

Lemma 4.6. *The optimal integer solution of the balanced submodular flow problem is the minimum of $h_I(y)$, where $y \in \mathbb{Z}_{\geq 0}$.*

Let y^* define the optimal solution of the LP problem. Now we get that, the optimal integer solution can be compute simply as follows:

Theorem 4.4. *The integer optimal solution of balanced submodular flow problem is equal to* $\min\{h_I(\lceil y^* \rceil), h_I(\lfloor y^* \rfloor)\} = \min\{\lceil h(\lceil y^* \rceil) \rceil, \lceil h(\lfloor y^* \rfloor) \rceil\}$*, where y^* is the minimum point of h.*

□

4.4 Balanced submodular network flow problem with intersecting submodular function

Let us consider balanced sub-modular flow problem with intersecting sub-modular function. We can define the problem in very similar way: Given a directed graph $G = (V, E)$, there is given an intersecting sub-modular function $b(X)$ for all $X \subseteq V$ node set. A flow in the graph is a sub-modular flow, if $\sum_{e \in \rho(X)} x_e - \sum_{e \in \delta(X)} x_e \leq b(X)$ holds for all $X \subset V$. The problem and its dual problem is formulated as the same way, when b was fully sub-modular. Then the solutions of the dual problem are a family of node sets, such that minimize $\min \frac{\sum_{Z \subset V} y_Z b'(Z)}{\sum_{e \in E} y_e}$. (Every set $Z \subset V$ has a weight y_Z in the family.) We denote the family $\mathcal{Z}$, such that $Z \in \mathcal{Z} \Leftrightarrow y_Z > 0$.

We remind of the modified dual problem: Dual optimal solutions are equivalent to optimal solutions of the following modified problem.

$$\begin{aligned}
&\min \frac{\sum_{Z \subset V} y_Z b'(Z)}{\sum_{e \in E} y_e} \\
&y_e \geq \sum_{e \in \delta(Z)} y_Z - \sum_{e \in \rho(Z)} y_Z && \forall e \in E \\
&\sum_{e \in E} y_e > 0 \\
&y_e, y_Z \geq 0 && \forall e \in E, Z \subset V
\end{aligned}$$

Now we examine the optimal solutions of this problem. If there are two intersecting sets X, Y in the family, then we can replace them by their intersection and

union. Weights $y_{X\cup Y}, y_{X\cap Y}$ are increased by $\min\{y_X, y_Y\}$ and weight y_X, y_Y are decreased by the same value. We get a new feasible solution, whose objective function value isn't increased.

Thus we can assume that $\mathcal{Z}$ is laminar. We transform this laminar set family $\mathcal{Z}$ as follows: Every step of the transformation starts with a family (Z) and ends a new family (Z'), that is objective value isn't greater than value of (Z). Besides there are some labelled sets in the family. We say that an unlabelled set $Z \in \mathcal{Z}$ is largest, if there is no other unlabelled set $Z' \in \mathcal{F}$, such that $Z \neq Z', Z \subset Z'$.

One step of transformation:

If there are more than one largest sets in $\mathcal{Z}$:

Let us see all largest unlabelled sets in $\mathcal{Z}$, that are denoted by $Z_1, Z_2, .., Z_i$. Let Z'_1 be the union of largest sets and let be $b'(Z'_1) = \sum_{j=1,2..,i} b(Z_j)$. We define $b'(Z)$ for each original set of the family $Z \in \mathcal{Z}$, which is equal the weight of Z in the beginning of this step. Weights of $Z_1, Z_2, ...Z_i$ are decrease by $\min_{j=1,2..i} y_{Z_j}$ and set Z'_1 is added to family $\mathcal{Z}$ with wight $\min_{j=1,2..i} y_{Z_j}$. Then amount of $\sum_{Z\in\mathcal{Z}} b'(Z)$ is equal to the total weight of the original family. (Original family is the family in the beginning of the actual step of the transformation.) Amount of $\sum_{e\in E} y_e$ isn't increased. (Some edges have the same weight and weights of other edges is decreased.) Therefore, the dual objective value of family $\mathcal{Z}'$ isn't increased during this step of the transformation, this means that: $\frac{\sum_{Z\in\mathcal{Z}} b(Z)y_Z}{\sum_{e\in E} y_e} \geq \frac{\sum_{Z\in\mathcal{Z}'} b'(Z)y'_Z}{\sum_{e\in E} y'_e}$
After this Z'_1 is labelled.

If there is exactly one largest unlabelled set in $\mathcal{Z}$, then it become labelled.

We doesn't consider labelled sets during steps, but at the end of the transformation we will use them.

We repeat this step, while there is at least one unlabelled set in $\mathcal{Z}$. The beginning family of a step is the obtained family at the end of the previous step.

Number of pairs of disjoint sets is decreased at each step, because we deleted at least one set of $Z_1, Z_2, .., Z_i$ and we added their union, that isn't disjoint with any set in $\mathcal{Z}$. Since every labelled set contains all unlabelled sets, the union can't be disjoint with any unlabelled set. Because of it is union of unlabelled sets, it also can't be disjoint any labelled set.

Transformation method does at most ($|\mathcal{Z}|$+#(number of disjoint pairs)) steps. Let $\mathcal{Z}^*$ be the family of sets obtained by the transformation. We have that:

$$\frac{\sum_{Z\in\mathcal{Z}} b(Z)y_Z}{\sum_{e\in E} y_e} \geq \frac{\sum_{Z\in\mathcal{Z}^*} b'(Z)y^*_Z}{\sum_{e\in E} y^*_e} = \frac{\sum_{Z\in\mathcal{Z}^*} b'(Z)y^*_Z}{\sum_{Z\in\mathcal{Z}^*} \delta(Z)y^*_Z}$$

Thus $\mathcal{Z}^*$ is a family in which each pair of sets are related by containment. Set function b' is fully submodular for sets in $\mathcal{Z}^*$, since for all $Y \subset X$ the submodularity constraint holds true: $b''(X) + b''(Y) \geq b''(X \cap Y) + b''(X \cup Y) = b''(X) + b''(Y)$

Similarly to the case where b was fully sub-modular, we can choose a set Z^* from $\mathcal{Z}^*$, such that:

$$\frac{\sum_{Z\in\mathcal{Z}^*} b'(Z)y_Z^*}{\sum_{Z\in\mathcal{Z}^*} \delta(Z)y_Z^*} \geq \frac{b'(Z^*)}{\delta(Z^*)}$$

Now $y_{Z^*}^*$ can be equal to 1, as its value doesn't matter in the objective fuction. Then Z^* is an union of some disjoint sets of the original family $\mathcal{Z}$. They are denoted by $Z_1^*, Z_2^*, ..., Z_i^*$, that is $Z^* = Z_1^* \cup Z_2^* \cup .. \cup Z_i^*$. Due to the definiton of b':

$$\frac{\sum_{Z\in\mathcal{Z}} b(Z)y_Z}{\sum_{e\in E} y_e} \geq \frac{b'(Z^*)}{\delta(Z^*)} = \frac{\sum_{j=1,2..,i} b(Z_j^*)}{\delta(\bigcup_{j=1,2..,i} Z_j^*)}$$

In summary, we can assume that the optimal solution of the modified dual problem consists pairwise disjoint sets, where weights of every selected set are the same (practicably 1).

If $\delta(\bigcup_{j=1,2..,i} Z_j^*) = 0$ holds true, then the dual problem is unbounded. Since the dual optimal solution isn't positive, thus $\sum_{j=1,2..,i} b(Z_j^*) < 0$. Adding these sets with an arbitrary large weight, we got an arbitrary small solution. This means that the primal problem can't be feasible.

If $b(Z) \geq 0$ hold for all $Z \subset V$, then the primal problem has a trivial solution, because of $x_e \equiv 0$ is primal feasible.

Therefore, provided the primal problem is feasible and the trivial solution isn't feasible, then we can solve the dual problem finding pairwise disjoint sets $Z_1, Z_2, ..., Z_i$, that minimizes $\frac{\sum_{j=1,2..,i} b(Z_j)}{\delta(\bigcup_{j=1,2..,i} Z_j)}$

5 Packing of T-joins

In this section we will describe a polynomial combinatorial algorithm for finding an optimal T-join packing, that is presented by Barahona [3]. First we will discuss its LP formulation using theory of Blocking Polyhedra, that is proved by Fulkerson [7]. Then we will give a description of Padberg-Rao's algorithm for finding a minimum T-cut [17]. In the end of this section, we will give a formal description and analysis of Barahona's algorithm for maximum T-joins packing problem [3].

5.1 Problem formulation

Let be $G = (V, E)$ an undirected graph, and $T \subseteq V$, $|T|$ even cardinality subset of nodes.

Definition 5.1. *Given $S \subset V$, we say that $\delta(S)$ is a T-cut, if $|S \cap T|$ is odd.*

Definition 5.2. *Given $J \subset E$, we say that $H = (V, J)$ spanning sub-graph is a T-join, if $d_H(V)$ is odd if and only if $v \in T$. (In the other words, $d_H(V)$ is odd for all $v \in T$ and $d_H(V)$ is even for all $v \in V \setminus T$.)*

Given a matrix A whose rows are the incidence vectors of T-cuts and a non-negative arc cost w, the linear program below is described:

$$\begin{aligned} \min(wx) \\ Ax \geq 1 \\ x \geq 0 \end{aligned}$$

Edmonds and Johnson proved that its optimal integer solution is an incidence vector of T-join [5]. They showed a polynomial combinatorial algorithm to solve this minimum cost T-join problem and its dual, that looks like as follows:

$$\begin{aligned} \max(y1) \\ yA \leq w \\ y \geq 0 \end{aligned}$$

This dual problem gives the maximum packing of T-cuts.

We will use the theory of Blocking Polyhedra [6], that is given by Fulkerson. He investigate a duality relationship between non-negative convex polyhedra (blocking and anti-blocking pairs of polyhedra). Let us $\mathcal{A} = \{x \in \mathbb{R}^n_+ \mid Ax \geq 1\}$ be a convex

polyhedron, where A is a 0-1 matrix with rows $a_1, a_2, .., a_n$. We denote by $b_1, b_2...b_m$ the extrem points of $\mathcal{A}$. Let B be a matrix, whose rows are $b_1, b_2...b_m$.

We say that $\mathcal{B}$ is the blocker (or blocking matrix) of $\mathcal{A}$, if $\mathcal{B} = \{y \in \mathbb{R}^n_+ \mid yx \geq \forall x \in 1 \forall x \in \mathcal{A}\}$.

Theorem 5.1. *Let $\mathcal{A}, \mathcal{B}, a_1, ..a_n, b_1, .., b_m$ be defined above. Then $\mathcal{B} = \{x \in \mathbb{R}^n_+ \mid Bx \geq 1\}$ and $\mathcal{A}$ is the blocker of $\mathcal{B}$.*

By using this theorem, we can describe the dual solutions of T-join packing problem. The minimum cost T-join problem can be formulated with matrix A, whose rows are the incidence vector of T-cuts. We mentioned that $\{x \in \mathbb{R}^n_+ \mid Ax \geq 1\}$ problem's optimal solutions is matched the incidence vectors of T-join. Let us given B matrix, whose rows are incidence vectors of T-joins in G. Applied the theory of Blocking Polyhedra, B is the blocking matrix of A as well as the extrem points of the following polyherdon are the incidence vectors of T-cuts.

$$\begin{aligned} \min(cx) \\ Bx &\geq 1 \\ x &\geq 0 \end{aligned}$$

Therefore the optimal solutions of this problem are the incidence vectors of T-cuts.

The dual problem gives the maximal packing of T-joins:

$$\begin{aligned} \max(y1) \\ yB &\leq c \\ y &\geq 0 \end{aligned}$$

Barahona showed an algorithm for finding the maximum fractional packing of T-joins, that runs in $\mathcal{O}(n^6)$ time. [3]
We will use $\overline{c}(S)$ to denote the capacity of the cut $(S, V \setminus S)$: $\overline{c}(S) = \sum_{e \in \delta(S)} c(E)$.

5.2 Minimum T-cut problem

The minimum T-cut can be found with Padberg-Rao's algorithm in polynomial time. It is based on the following lemma:

Lemma 5.1. *Let S be a minimum cut, which separates at least two T nodes. If $|S \cap T|$ is odd then S is a minimum T-cut.*

If $|S \cap T|$ is even then there is a $S' \subset S$ or $S' \subset V \setminus S$, that defines a minimum T-cut.

Proof. The statement is trivial, in the case where $|S \cap T|$ is odd.
So we can assume that $|S \cap T|$ is even. Let be A a minimum T-cut in the graph.

1. case: $|A \cap S \cap T|$ is odd:

1.a) case: $A \cup S$ separating at least two nodes in T:
Because of $\overline{c}$ submodularity:
$\overline{c}(A \cap S) + \overline{c}(A \cup S) \leq \overline{c}(A) + \overline{c}(S)$
But $A \cup S$ separates two T-nodes, for this $\overline{c}(S) \leq \overline{c}(A \cup S)$. As well as $A \cap S$ is a T-cut, thus $\overline{c}(A) \leq \overline{c}(A \cap S)$.
Therefore $\overline{c}(A \cap S) = \overline{c}(A) \rightarrow A \cap S$ is a minimum T-cut.

1.b) case: $A \cup S$ not separating two T nodes, i. e. $T \subseteq A \cup S$:
$A' = V \setminus A$ is also a minimum T-cut. $A' \cup S$ doesn't include T, since $A \setminus S$ contains T node for sure. Thus $A' \cup S$ separates two nodes in T. Now, we can see that $A' \cap S$ is a minimum T-cut really similarly way as in case a).

2. case: $|A \cap S \cap T|$ is even:
Then $S' = V \setminus S$ defines the same cut and $|A \cap S' \cap T|$ is odd. Apply the first case technique to see that S' is a minimum T-cut. □

Based this lemma, a fast algorithm can be given for find the minimum T-cut.

Algorithm 3 Min. T-cut search(G)

```
if |T| = 0 or |V| = 0 then
    exercise not reasonable, RETURN FAIL
Let be S minimum cut separating two nodes in T
if S ∩ T is odd then
    Split contracted nodes, if there is any
    RETURN S
if S ∩ T is even then
    G_1 = G/_S (contract S in G)
    G_2 = G/_{V\S} (contract V \ S in G)
    Min. T-cut search(G_1)
    Min. T-cut search(G_2)
```

If we use the graph's Gomory-Hu tree [9], the algorithm will become more simplier. By this, we can choose the minimum edge wich defines a T-cut in the Gomory-Hu tree.

5.3 Maximum packing of T-joins algorithm

Follow from the weak duality theorem, capacity of an arbitrary T-cut isn't less than the value of any T-joins packing. For this bound to be tight, in the optimal fractional packing of T-joins every positive weighted T-join has to intersect in exactly one edge every minimum T-cut.

Let $\lambda(G)$ be the optimal value in G graph. It can be computed with the previous algorithm.
For arbitrary $U \subseteq E$ and $\alpha > 0$, let $G - \alpha U$ be a graph, that is obtained by the following way: capacities of edges in U is reduced to $c(e) - \alpha$. If capacity of an edge becomes 0, then we remove that edge from the graph.
Let us be $\mu(U) = \min_{e \in U} c(E)$.
For all U T-join, we define α_U value, such that
$\alpha_U = \max(\alpha \mid \lambda(G - \alpha U) = \lambda(G) - \alpha, 0 \leq \alpha \leq \mu(U))$

With these notations, the problem can be solved recursively: Given a T-join U with a positive weight in an optimal packing, we can compute α_U. Then $G' = G - \alpha_U U$ graph's optimal T-join packing is completed with the α_U weighted U T-join. This way we get the optimal packing of G graph.

Lemma 5.2. *If U is a T-join and $\alpha_U = 0$, then there is a minimum T-cut S, such that $|\delta(S) \cap U| > 1$*

Proof. In $G - \alpha_U U$ graph, the cost of a T-cut decreases by $k\alpha_U$, where $k = |\delta(S) \cap U|$. Therefore, if $|\delta(S) \cap U| = 1$ holds for some T-cut, then $\exists \alpha_U > 0$ small value, such that $\lambda(G - \alpha_U U) = \lambda(G) - \alpha_U$ □

Similarly, if there are a T-join U and a T-cut S, where $|\delta(S) \cap U| > 1$, then $\alpha_U = 0$ must hold.

Lemma 5.3. *Let A, B be minimal T-cuts, which are intersecting (that is to say $A \cap B \neq \emptyset, A \setminus B \neq \emptyset, B \setminus A \neq \emptyset, V \setminus (A \cup B) \neq \emptyset$) and $|A \cap B \cap T|$ is odd. Let U be a T-join.*
If $\delta(A \cap B), \delta(A \cup B)$ intersect U in exactly one edge $\Rightarrow$ $\delta(A), \delta(B)$ also intersect U in one edge.

Proof. Because of $\overline{c}$ submodularity: $\overline{c}(A \cap B) + \overline{c}(A \cup B) \leq \overline{c}(A) + \overline{c}(B)$. Since A, B are minimal T-cuts, $A \cap B, A \cup B$ must be minimal T-cuts and between $A \setminus B, B \setminus A$ there is no edge.
For a T-join U and a T-cut S, $|U \cap \delta(S)|$ has to be odd. Now it is easy to see, that if a T-join intersect $A \cap B, A \cup B$, then it also intersects A, Bin one edge. □

Very similar proof can be given for the following lemma:

Lemma 5.4. *Let A, B be minimal T-cuts, which are intersecting (that is to say $A \cap B \neq \emptyset, A \setminus B \neq \emptyset, B \setminus A \neq \emptyset, V \setminus (A \cup B) \neq \emptyset$) and $|A \cap B \cap T|$ is even. Let U be a T-join.*
If $\delta(A \setminus B), \delta(B \setminus A)$ intersect U in one edge $\Rightarrow$ $\delta(A), \delta(B)$ also intersect U in one edge.

With the previous two lemma, we get the following statement. Let $\mathcal{S}$ be a non-expandable laminar set family of T-cuts. If a T-join U intersects in one edge every $S \in \mathcal{S}$, than U intersects every minimal T-cut in one edge. In this case, U can be taken to the packing with positive weight by lemma 5.2.

In the algorithm for finding an optimal packing, we simultaneously build T-join packing and a Φ laminar set family of T-cuts. At every iteration, we find a T-join with positive weight in the optimal packing and we can reduce the graph or we find a new minimum T-cut and we can extend Φ.

Let U be a T-join, which intersects in one edge every T-cut in Φ. Then there are two possible case:

1. The first case when $\alpha_U = \mu(U)$: As at least one edge capacity decrease to 0 and this edge is deleted from G, number of edges in $G - \alpha_U U$ is less, then number of edges in G.

2. The second case when $\alpha_U < \mu(U)$: as T-join U can't be packed greater weight than α_U, in $G - \alpha_U U$ there is a minimal T-cut S, such that $|U \cap \delta(S)| > 1$. This means that $S \notin \Phi$. (In $G - \alpha_U U$ α_U is equal to 0.) So S is added to ϕ and Φ can be uncrossed. (We want Φ always to be a laminar set family.)

Said uncrossing procedure works as follows:

Algorithm 4 Uncrossing procedure(Φ, S, U)

while There is $A \in \Phi$, such that U cross A **do**
  if $|A \cap S \cap T|$ is odd **then**
    if $\delta(A \cup S) \cap U > 1$ **then**
      $S \coloneqq A \cup S$
    if $\delta(A \cup S) \cap U = 1$ **then**
      $S \coloneqq A \cap S$
  if $|A \cap S \cap T|$ is even **then**
    if $\delta(A \setminus S) \cap U > 1$ **then**
      $S \coloneqq A \setminus S$
    if $\delta(A \setminus S) \cap U = 1$ **then**
      $S \coloneqq A \setminus A$
S is added to Φ

Therefore the algorithm does polynomial iterations, because at every iteration either number of edges in G decreases or size of Φ increases. So the algorithm does

at most $m + 2n - 1$ iterations. (As a laminar set family contains at most $2n - 1$ sets over n elements.)

Algorithm 5 Maximum packing of T-joins (G)

1: $\Phi = \emptyset$
2: Search U T-join, such that $|U \cap \delta(S)| = 1 \; \forall S \in \Phi$
3: Compute α_U
4: **if** $\alpha_U < \mu(U)$ **then**
5: Find $S \notin \Phi$ set, such that $G - \alpha_U U$-ban $|U \cap \delta(S)| > 1$
6: Add S to Φ
7: Run Uncrossing procedure(Φ, S, U)
8: $G = G - \alpha_U U$
9: **if** $\lambda(G) > 0$ **then**
10: Go to 2. step

Now we describe the second step in Maximum packing of T-joins algorithm. We want to find a T-cut U, such that $|U \cap \delta(S)| = 1 \; \forall S \in \Phi$. A new arc cost is defined as follows: $c'(e) =$#(Number of sets $S \in \Phi$, such that $e \in \delta(S)$). We search a minimal cost T-joins. Its cost is at least $|\Phi|$, since every T-join intersects any T-cut at least 1 edge. Therefore minimal T-joins, whose cost is equal $|\Phi|$, has to intersect any set in Φ in exactly one edge.

In third step of the algorithm, we have to compute α_U. For this we define $f(\alpha) = \lambda(G - \alpha U)$. The function f is the minimum of linear functions, thus f is concave and piece-wise linear. We would like to find its first break-point f. If we choose a bigger α than this break-point, we find a new T-cut, which is defined that linear function. Because of the greater gradient, this T-cut intersects U in minimum 2 edge. As $|U \cap \delta(S)| > 1$ holds, we can added this new minimum T-cut to Φ. (This T-cut's cost in the reduced graph is equal to minimum T-cut.) This procedure looks like as follows:

Algorithm 6 Compute α_U

1: $\alpha_U = \mu(U)$
2: Search S minimal T-cut in $G - \alpha_U U$
3: **if** $\lambda(G - \alpha U) = \lambda(G) - \alpha_U$ **then**
4: Return α_U, S
5: **if** $\lambda(G - \alpha U) < \lambda(G) - \alpha_U$ **then**
6: Let us α' be the solution of $\lambda(G) - \alpha = \overline{c}(S) - k\alpha$, such that $k = |U \cap \delta_G(S)|$
7: $\alpha_U = \alpha'$ and go to 2. step

The value of k is decreased at every iteration and k is an integer. Thus the procedure takes at most $|U| \leq n - 1$ iterations. Consider its running time, there are

two possible case:
If $\alpha_U = \mu(U)$, then it makes at most $\mathcal{O}(n)$ minimum st-cut computations, so it runs at $\mathcal{O}(n^4)$ times.
If $\alpha_U < \mu(U)$, then it makes at most $\mathcal{O}(n^2)$ minimum st-cut computations, so it runs at $\mathcal{O}(n^5)$ times.

In the fifth step in the algorithm 5, a new minimum T-cut S is found. The procedure 6 in the third step gives this minimum T-cut.

As we said, the 5 algorithm does at most $m + 2n - 1$ iteration. The running time of step 2 is $\mathcal{O}(n^3)$, because we compute all-pairs shortest path problem and a minimum cost perfect matching. (Both problem can be solved in $\mathcal{O}(n^3)$ time.) There are two types of iteration:
Its does at most m iterations, when $\alpha_U = \mu(U)$: In this case, the total running time is $\mathcal{O}(m * (n^4 + n^3)) = \wr(mn^4)$.
And there are at most $2n - 1$ iterations, when $\alpha_U < \mu(U)$: in this case, the total running time is $\mathcal{O}(n * (n^5 + n^3)) = \mathcal{O}(n^6)$.
Thus the algorithm runs in $\mathcal{O}(mn^4 + n^6) = \mathcal{O}(n^6)$ time.

6 Packing of perfect matchings in bipartite graph

In this section, we consider maximum packing of perfect matching problem in bipartite graphs. In the beginning we show an algorithm to compute optimal solution based on an separation method. Then we characterize dual optimal solutions and thereafter we present a polynomial time algorithm for solve maximum packing problem.

Let us given a bipartite graph $G = (S, T, E)$ and every edge e has a non-negative capacity $u(e)$. We would like to find the maximum fractional packing of perfect matchings. Let A be the incidence matrix of G, the linear program below is described:

$$\begin{aligned} \max(0x) \\ Ax &= \alpha \\ x &\leq u \end{aligned}$$

If $\alpha = 1$, then the above polyhedron is the perfect matching polytope. An arbitrary vector of this polytope is a convex combination of perfect matchings incidence vectors. If a solution of the above problem is divided by α, then we get a vector, which is in the perfect matchings polytope. For this, it can be written as convex combination of perfects matchings. This is a fractional 1-packing in G. Then coefficients is multiplied by α. The given packing of perfect matchings is a feasible packing in G. Therefore, the solutions of above problem are α-valued packing of perfect matchings in G.

It can be converted to a network flow problem. Let $G' = (S \cup T, E)$ be a directed graph, that differs from the graph G since edges are directed from S to T. We define a demand function $b : V \to \mathbb{R}$ as follows: $b(s) = -\alpha$ for all $s \in S$ and $b(t) = \alpha$ for all $t \in T$. If there exist a circulation in G', such that $\rho(v) - \delta(v) = b(v)$ for each $v \in S \cup T$, then there is an α-valued packing of perfect matching in G. If X_e is equal to the amount of the flow on e, then x is a feasible solution of above problem. ($\sum_{e=(st_i)} x_e = \rho(s) + \delta(s) = \delta(s) = \alpha$, because $\rho(s) = \emptyset$ and similarly $\sum_{e=(s_i t)} x_e = \alpha$.)

6.1 Computing optimal value

For solving perfect matchings packing problem, it is enough to find a feasible circulation of the above network problem. Hoffman gave a characterization for this problem [11]: there exists a feasible flow if and only if $\rho_0(X) - \delta_u(X) \leq b(X)$ holds true for all $X \subseteq V$. This means that if the α-packing problem isn't feasible, there

exist a cut $X \subseteq V$, such that $\rho_0(X) - \delta_u(X) > b(X)$.
Re-written: $\exists X \subseteq V : -\delta_u(X) > b(X) \Leftrightarrow 0 > b(X) + \delta_u(X)$

Now we want to find $X \subseteq V$, that minimize the following function:

$$b(X) + \delta_u(X) = \delta_u(X) - \alpha|X \cap S| + \alpha|X \cap T| = \delta_u(X) - \alpha(|X \cap S| - |X \cap T|)$$

In the beginning we construct an auxiliary graph G_α as follows: we add two extra nodes s', t' to G'. We define a weight function w, that is $w(e) = u(e)$ for all $e \in E$.
Edge $s' \to v$ is added with weight $w(sv) = \alpha$ for all $v \in S \cup T$.
Each $s \in S$ has a new edge $s \to t'$ with weight $w(st') = 0$ and each $t \in T$ has a new edge $t \to t'$ with weight $w(tt') = 2\alpha$.
Let us see the weight of an arbitrary s'-t' cut X in G_α:

$$\begin{aligned}
w(X) = \sum_{e \in \delta(X)} &= \sum_{e \in \delta(X), e = s'v} w(e) + \sum_{e \in \delta(X), e = vt'} w(e) + \sum_{e \in \delta(X), e = st} w(e) = \\
&= \sum_{s'v \in \delta(X)} \alpha + \sum_{st' \in \delta(X)} 0 + \sum_{tt' \in \delta(X)} 2\alpha + \sum_{st \in \delta(X)} u(e) = \\
&= (|S| + |T| - |X|)\alpha + |X \cap T|2\alpha + \sum_{e \in \delta_{G'}(X)} u(e) = \\
&= (|S| - |X \cap S|)\alpha + (|T| - |X \cap T|)\alpha + |X \cap T|2\alpha + \sum_{e \in \delta_{G'}(X)} u(e) = \\
&= |S|\alpha - |X \cap S|\alpha + |T|\alpha + |X \cap T|\alpha + \sum_{e \in \delta_{G'}(X)} u(e) = \\
&= (|S| + |T|)\alpha + \sum_{e \in \delta_{G'}(X)} u(e) - \alpha(|X \cap S| - |X \cap T|)
\end{aligned}$$

Thus weight of cut $X \cup S'$ is $(|S|+|T|)\alpha$ greater than $\delta_u(X) - \alpha(|X \cap S| - |X \cap T|)$, that we want to minimize. If α is fixed, than $(|S| + |T|)\alpha$ is constant.
This means that the minimum cut in G_α is the same as the cut, that minimize $\delta_u(X) - \alpha(|X \cap S| - |X \cap T|)$. Since the minimum weight s'-t' cut can be found, we get the minimum of the above formula.

Now we show a method, that computes the optimal value of packing of the perfect matchings. First we choose α_0, which is certainly greater than the optimum value. (For example $\alpha_0 = \sum_{e \in \delta(s)} u(e)$ is a good choose, where s is an arbitrary node in S.) Then we construct G_{α_0} auxiliary graph and we compute a minimum s'-t' cut X_0. If this minimum is equal to 0, then there is an α_0-valued packing in G. If it is negative, we compute the next guess, that is got closer to the optimum.

$$\alpha_{i+1} = \frac{\delta_u(X_i)}{(|X_i \cap S| - |X_i \cap T|)}$$

Since $0 > \delta_u(X_i) - \alpha(|X_i \cap S| - |X_i \cap T|)$ holds true for any $\alpha > \alpha_{i+1}$, then optimal value is less then or equal to α_{i+1} by Hoffman theorem.
We repeat this iterative computation, until we find the optimal value of perfect matching packing.

Algorithm 7 Compute optimal value(G, u)

1: $\alpha = \sum_{e \in \delta(s)} u(e)$, where s is chosen randomly from S
2: construct G_α
3: compute minimum weight s'-t' cut $\rightarrow X$
4: **while** $0 > \delta_u(X) - \alpha(|X \cap S| - |X \cap T|)$ **do**
5: $\quad \alpha = \frac{\delta_u(X_i)}{(|X_i \cap S| - |X_i \cap T|)}$
6: $\quad$ construct G_α
7: $\quad$ compute minimum weight s'-t' cut $\rightarrow X$
8: RETURN α, X

The algorithm 7 does at most $|S|$ repetition, because $(|X_i \cap S| - |X_i \cap T|)$ is decreased at every iteration. ($0 \leq |X_i \cap S| - |X_i \cap T| \leq |S|$ and it is integer.)
Let X_i denote the minimal cut found at the i-th iteration.
If $|X_{i+1} \cap S| - |X_{i+1} \cap T| \geq |X_i \cap S| - |X_i \cap T|$:

$$\alpha_{i+1} < \alpha_i \rightarrow (\alpha_i - \alpha_{i+1})(|X_i \cap S| - |X_i \cap T|) \leq (\alpha_i - \alpha_{i+1})(|X_{i+1} \cap S| - |X_{i+1} \cap T|)$$
$$\delta_u(X_i) - \alpha_i(|X_i \cap S| - |X_i \cap T|) \leq \delta_u(X_{i+1}) - \alpha_i(|X_{i+1} \cap S| - |X_{i+1} \cap T|)$$
$$\delta_u(X_i) - \alpha_i(|X_i \cap S| - |X_i \cap T|) + (\alpha_i - \alpha_{i+1})(|X_i \cap S| - |X_i \cap T|) \leq$$
$$\leq \delta_u(X_{i+1}) - \alpha_i(|X_{i+1} \cap S| - |X_{i+1} \cap T|) + (\alpha_i - \alpha_{i+1})(|X_{i+1} \cap S| - |X_{i+1} \cap T|)$$
$$\delta_u(X_i) - \alpha_{i+1}(|X_i \cap S| - |X_i \cap T|) \leq \delta_u(X_{i+1}) - \alpha_{i+1}(|X_{i+1} \cap S| - |X_{i+1} \cap T|)$$
$$0 \leq \delta_u(X_{i+1}) - \alpha_{i+1}(|X_{i+1} \cap S| - |X_{i+1} \cap T|)$$

This is a contradiction, provided the algorithm doesn't stop at the $i+1$-th iteration. In this case X_{i+1} is the minimum cut in $G_{\alpha_{i+1}}$, so it have to be negative. Thus, $|X_{i+1} \cap S| - |X_{i+1} \cap T| < |X_i \cap S| - |X_i \cap T|$ holds true at any intermediate iteration. Every repetition runs is $\mathcal{O}(n^3)$ time, because its running time is dominated by minimum s'-t' cut computations. Therefore the algorithm finds the optimal value of packing of perfect matching in $\mathcal{O}(mn^3) = \mathcal{O}(n^5)$ time.

We can solve the packing problem as follows: First we compute the optimal α. Then we formulate the linear program with this α. Solving this LP problem, the solution can be written as a convex combination of perfect matchings. As we described at the beginning of this section, given packing is a feasible solution of the problem.

Let us note that running one iteration of algorithm 7 with an arbitrary λ, we can decide either there is a λ-packing in G or not. Using this method, a separation algorithm is obtained for an arbitrary λ to decide that the optimal value of the packing problem is less than λ or not. However, we can't say that the optimal value is equal to λ or not. With this separation method, Megiddo's parametric search computes the optimal value of maximum packing problem [16].

6.2 An algorithm for solving packing problem

We show another algorithm for solving maximum packing of perfect matching in a bipartite graph based on that we can compute the optimum value in polynomial time.

It is easy to see the next lemma by Hall-theorem [10]:

Lemma 6.1. *Let X, Y be node sets, such that $X \subseteq S, Y \subseteq T$ and $|Y| < |X|$. Let $L \subset E$ be given as follows: $L = \{e \in E \mid e \in E(X, T \setminus Y)\}$, that L contains every edge between X and $T \setminus Y$. Then every perfect matching intersects L in at least $|X| - |Y|$ edge.*

Let A be a matrix, whose columns are incidence vectors of perfect matchings. Maximum packing problem can be formulated as follows:

$$\begin{aligned} \max(1x) \\ Ax &\leq u \\ x &\geq 0 \end{aligned}$$

Let us see its dual problem:

$$\begin{aligned} \min(yu) \\ yA &\geq 1 \\ y &\geq 0 \end{aligned}$$

We showed that the optimal value can be calculated and a cut Z is gotten, such that $\frac{\delta_u(Z)}{(|Z \cap S| - |Z \cap T|)}$ is equal to the optimum. We prove that this cut Z defines an optimal solution of the dual problem.

Lemma 6.2. *Let Z be the cut, that is found by algorithm 7. Let y be a 0-1 vector, such that $y_e = 1 \Leftrightarrow e \in \delta(Z)$ in G'. Then $y^* = \frac{1}{|Z \cap S| - |Z \cap T|} y$ is an optimal solution of the dual problem.*

Proof. Let $X = Z \cup S$, $Y = Z \cap T$ be node sets, and $L \subset E$ be defined like in the previous lemma.

$e \in L \Leftrightarrow e \in \delta(Z)$, since $|Z \cup S| > |Z \cup T|$ holds for optimal Z. This means that y is the incidence vector of L.
Every perfect matching has to intersect L in at least $|X| - |Y|$ edge by lemma 6.1, this implies that $y^* a^i = \frac{1}{|Z \cap S| - |Z \cap T|} y a^i \geq \frac{1}{|Z \cap S| - |Z \cap T|}(|Z \cap S| - |Z \cap T|) = 1$ holds, where a^i denotes i-th column in A. Thus y^* satisfies the dual constraints.

We showed that $\frac{\delta_u(Z)}{(|Z \cap S| - |Z \cap T|)}$ is equal to the optimum, and this value is equal to the objective function value of y^*. Therefore y^* is a dual optimal solution. □

In the optimal fractional packing of perfect matching every positive weighted perfect matching has to intersect in exactly $|Z \cap S| - |Z \cap T|$ edge every dual optimal cut Z, since the corresponding dual constraint has to be tight.

We use $k(X)$ to denote $k(X) = |X \cap S| - |X \cap T|$. Before we continue examining dual optimal solutions, we make some very simple observations:

Statement 6.1. *$k(X)+k(Y) = k(X \cap Y)+k(X \cup Y)$ holds for every pair $X, Y \subset V$.*

Proof.

$$
\begin{aligned}
&k(X \cup Y) = |(X \cup Y) \cap S| - |(X \cup Y) \cap T| = \\
&= |X \cap S| + |Y \cap S| - |(X \cap Y) \cap S| - (|X \cap T| + |Y \cap T| - |(X \cap Y) \cap T|) = \\
&= k(X) + k(Y) - k(X \cap Y) \\
&k(X \cap Y) + k(X \cup Y) = k(X) + k(Y)
\end{aligned}
$$

□

Statement 6.2. *Let d_1, d_2, d_3, d_4 and k_1, k_2, k_3, k_4 be non-negative numbers, that satisfy the next inequalities: $d_1 + d_2 \geq d_3 + d_4$, $k_1 + k_2 = k_3 + k_4$ and $\frac{d_1}{k_1} = \frac{d_2}{k_2}$. Then at least one of the following holds: $\frac{d_1}{k_1} \geq \frac{d_3}{k_3}$ or $\frac{d_1}{k_1} \geq \frac{d_4}{k_4}$*

Proof. It can be proved by contradiction. Suppose that neither of them holds, we have that:

$$
\begin{aligned}
&\frac{d_1}{k_1} < \frac{d_3}{k_3} \rightarrow d_1 k_3 < d_3 k_1 \\
&\frac{d_2}{k_2} < \frac{d_3}{k_3} \rightarrow d_2 k_3 < d_3 k_2 \\
&\frac{d_1}{k_1} < \frac{d_4}{k_4} \rightarrow d_1 k_4 < d_4 k_1 \\
&\frac{d_2}{k_2} < \frac{d_4}{k_4} \rightarrow d_2 k_4 < d_4 k_2
\end{aligned}
$$

These are implies that: $(d_1 + d_2)(k_3 + k_4) < (d_3 + d_4)(k_1 + k_2)$.
This is an contradiction, because of $k_1 + k_2 = k_3 + k_4$. □

Now we can continue considering dual optimal solutions:

Lemma 6.3. *Let Z_1, Z_2 be cuts in G, that define dual optimal solutions. Let us define the following notations: $Z_3 = Z_1 \cap Z_2$, $Z_4 = Z_1 \cup Z_2$. Then $k(Z_3), k(Z_4)$ have to be non-negative.*

Proof. We prove it by contradiction. We assume that $k(Z_3) = k(Z_1 \cap Z_2) < 0$. Because of Z_i, Z_2 define optimal solutions, every positive weighted perfect matching in the optimal packing has to intersect them in exactly $k(Z_1)$ and $k(Z_2)$ edges. As well as every perfect matching has to intersect every cut Z in at least $k(Z)$ edges. By using statement 6.1, then a perfect matching intersects Z_4 in at least $k(Z_4) = k(Z_1) + k(Z_2) - k(Z_3) > k(Z_1) + k(Z_2)$ edges. This means that a positive weighted perfect matching contains exactly $k(Z_1)$ and $k(Z_2)$ edge in $\delta(Z_1), \delta(Z_2)$ and also contains more than $k(Z_1) + k(Z_2)$ edges in their union. This is a contradiction, so $k(Z_3) > 0$.
In the case where $k(Z_4) = k(= Z_1 \cup Z_2) < 0$, it can be proved similarly. □

Lemma 6.4. *Let Z_1, Z_2 be cuts in G, that define dual optimal solutions. Let us define the following notations: $Z_3 = Z_1 \cap Z_2$, $Z_4 = Z_1 \cup Z_2$. Assume that $k(Z_3)$, $k(Z_4)$ are non-negative. Then at least one of $Z_1 \cup Z_2, Z_1 \cap Z_2$ defines an optimal dual solution.*

Proof. Since δ is an submodular set function: $\delta(Z_1) + \delta(Z_2) \geq \delta(Z_3) + \delta(Z_4)$.
By statement 6.1, we have that $k(Z_1) + k(Z_2) = k(Z_3) + k(Z_4)$. Because of Z_1, Z_2 define optimal dual solutions: $\frac{\delta(Z_1)}{k(Z_1)} = \frac{\delta(Z_2)}{k(Z_2)}$
So we can apply statement 6.2 for $d_i = \delta(Z_i)$ and $k_i = k(Z_i)$.
Then at least one of the following holds: $\frac{\delta(Z_1)}{k(Z_1)} \geq \frac{\delta(Z_3)}{k(Z_3)}$ or $\frac{\delta(Z_1)}{k(Z_1)} \geq \frac{\delta(Z_4)}{k(Z_4)}$.
This means that $Z_3 = Z_1 \cap Z_2$ or $Z_4 = Z_1 \cup Z_2$ defines a dual optimal solution. □

We use the following notion:
A cut Z is called a tight cut, if $k(Z) \geq 0$ and every positive weighted perfect matching in the optimal packing intersects in exactly $k(Z)$ edges. It follows from the definition that if a cut Z defines an optimal solution, then Z is tight. Let us see a lemma of tight cuts:

Lemma 6.5. *Let Z_1, Z_2 be cuts in G, that define dual optimal solutions. Then $Z_1 \cup Z_2, Z_1 \cap Z_2$ are tight cuts.*

Proof. Let M be a perfect matching with positive weight in the optimal fractional packing. By a counting argument, it is easy to see the following: there is no edge in M between $Z_1 \setminus Z_2$ and $Z_2 \setminus Z_1$.
The statement of this lemma is implied by the next equality:
$|M \cap \delta(Z_1)| + |M \cap \delta(Z_2)| = |M \cap \delta(Z_1 \cap Z_2)| + |M \cap \delta(Z_1 \cup Z_2)|$.
(Due to the fact that $|M \cap \delta(Z_1)| \geq k(Z_1)$ and lemma 6.1) □

The theorem below show that we only need to impose this for a laminar family of tight cuts.

Theorem 6.1. *Let Z_1, Z_2 be cuts, that define optimal solutions. If $Z_1 \cup Z_2, Z_1 \cap Z_2$ are tights, then Z_1, Z_2 are tights cuts.*

Proof. Due to the previous lemma and the fact that a positive weighted perfect matching has no edge between $Z_1 \setminus Z_2$ and $Z_2 \setminus Z_1$, we get this theorem. □

We say that a perfect matching M satisfy the tightness condition, if M intersects every tight cut Z in exactly $k(Z)$ edges. A positive weighted perfect matching in the optimal packing must satisfy the tightness condition. Therefore, it is enough to keep a laminar family of tight sets $\mathcal{F}$, if we want to check that a perfect matching satisfy the tightness condition or not.

During the algorithm we will keep a laminar family of tight sets. If we add a new cut to $\mathcal{F}$, then we convert it into a laminar family by using the following simple uncrossing procedure: If there are two intersecting cut X, Y in $\mathcal{F}$, then we replace them by their union and intersection. It is easy to see that at each uncrossing step the number of crossing pairs decreases by at least one.

Let be $\lambda(G)$ the optimal value in graph G, that can be computed with the algorithm 7.
For arbitrary $M \subseteq E$ and $\alpha > 0$, let be $G - \alpha M$ a graph, that is obtained by the following way: capacities of edges in M is reduced to $u(e) - \alpha$. If capacity of an edge becomes 0, then we remove that edge from the graph.
Let us be $\mu(M) = \min_{e \in M} u(E)$.
For $\forall M$ perfect matching, we define α_M value, such that $\alpha_M = \max(\alpha \mid \lambda(G - \alpha M) = \lambda(G) - \alpha, 0 \leq \alpha \leq \mu(M))$

With these notations, the problem can be solved recursively: Given an M perfect matching with a positive weight in an optimal packing, we can compute α_M. Then $G' = G - \alpha_M M$ graph's optimal packing is completed with the α_M weighted M perfect matching. This way we get the optimal packing of graph G.

Now we can describe the algorithm for solving maximum packing of perfect matchings problem.

Algorithm 8 Maximum packing of perfect matchings (G)

1: $\mathcal{F} = \emptyset$
2: Search M perfect matching, such that $|M \cap \delta(Z)| = k(Z)\ \forall Z \in \mathcal{F}$
3: Compute α_M
4: **if** $\alpha_M < \mu(M)$ **then**
5: Find $Z \notin \mathcal{F}$ set, such that $G - \alpha_M M$-ban $|M \cap \delta(Z)| > k(Z)$
6: Add Z to $\mathcal{F}$
7: Run Uncrossing procedure($\mathcal{F} \cup Z$)
8: $G = G - \alpha_Z Z$
9: **if** $\lambda(G) > 0$ **then**
10: Go to 2. step

Let M be a perfect matching, which is satisfy the tightness condition for all $Z \in \mathcal{F}$. Then there are two possible case:
1. The first case when $\alpha_M = \mu(M)$: As we can add M to the packing and its weight is equal to its capacity. So at least one edge capacity decrease to 0 and this edge is deleted from G. Therefore in this case, the number of edges in $G - \alpha_M M$ is less, then number of edges in G.
2. The second case when $\alpha_M < \mu(M)$: as M can't be packed greater weight than α_M. This means that in $G - \alpha_M M$ there is a cut Z such that $|M \cap \delta(Z)| > k(Z)$, that is $\mathcal{F}$ doesn't contain Z. Adding Z to $\mathcal{F}$ and applying the uncrossing procedure to convert it into a laminar family, we get a family of tight cuts, which is greater then the original family.

Therefore the algorithm does polynomial repetitions, because at every iteration either number of edges in G decreases or size of $\mathcal{F}$ increases. So the algorithm does at most $m + 2n - 1$ iteration. (Because of a laminar set family contains at most $2n - 1$ sets over n elements.)

In the second step of algorithm 8 we would like to find a perfect matching, that intersects every cut $Z \in \mathcal{F}$ in exactly $k(Z)$ edges. We can manage it by the algorithm for find a minimum cost perfect matching [13]. A new arc cost c is defined as follows: $c(e)$ is equal to the number of cuts $Z \in \mathcal{F}$, such that $e \in \delta(Z)$. Now we search a minimum cost perfect matching M. M is a good candidate for having positive weight in the optimal packing. (Cost of M is equal to $\sum_{Z \in \mathcal{F}} k(Z)$.)

Let describe the third step of the algorithm, in which α_M is computed. Let M be a perfect matching, that satisfy the tightness condition for all $Z \in \mathcal{F}$. We use the new notation: $f(\alpha) = \lambda(G - \alpha M)$. Then f is a piece-wise linear function and we want to find its first break-point. By the definition of $\mu(M)$, $\alpha_M \leq \mu(M)$ holds. We start the process with initial value $\alpha_0 = \mu(M)$. Then value of α is decreased

until we find the biggest α, that $f(\alpha) = \lambda(G) - \alpha$ holds. ($f(\alpha) \leq \lambda(G) - \alpha$ for an arbitrary α.)

Algorithm 9 Compute α_M

1: $\alpha = \mu(M)$
2: Run compute optimal value$(G - \alpha_M M,\, u) \to f(\alpha), Z$
3: **if** $f(\alpha) = \lambda(G) - \alpha$ **then**
4: Return α, Z
5: **if** $f(\alpha) < \lambda(G) - \alpha$ **then**
6: Let us α' be the solution of $\lambda(G) - \alpha = u(Z) - k\alpha$, such that $k = |M \cap \delta(Z)|$
7: $\alpha_U = \alpha'$ and go to 2. step

The value of k is decreased at every iteration and k is an integer. Thus the procedure takes at most $|M| = n$ iterations. The running time of an iteration in this algorithm is dominated by the running time of Steps 2. As we said compute optimal value algorithm in the 2. step runs $\mathcal{O}(mn^3)$ time. Therefore, if $\alpha = \mu(M)$, then algorithm 9 runs in $\Updownarrow\backslash^{\ni}$ time. If $\alpha < \mu(M)$, then algorithm 9 runs in $\Updownarrow\backslash^{\triangle}$ time.

Now let us examine the running time of algorithm 8. Step 2 is run $\mathcal{O}(n^3)$ at every iteration. After finding M perfect matching, there are two possible case:
1. $\alpha_M = \mu(M)$: in this case compute α_M algorithm runs in $\mathcal{O}(mn^3)$ time. As well as there are at most m this kind of iteration, because number of edges in G decreases at each repetition. So the total running time in this case is $\mathcal{O}(n^3) + \mathcal{O}(m^2n^3)$.
2. $\alpha_M < \mu(M)$: in this case compute α_M algorithm runs in $\mathcal{O}(mn^4)$ time. As well as there are at most $2n - 1$ iterations, because size of $\mathcal{F}$ increases at each this kind of repetition. So the total running time in this case is $\mathcal{O}(n^3) + \mathcal{O}(mn^5)$.

Thus the complexity of the packing algorithm is $\max\{\mathcal{O}(m^2n^3), \mathcal{O}(mn^5)\} = \mathcal{O}(n^7)$. Then we get the following theorem:

Theorem 6.2. *Let us given a bipartite graph $G = (S, T, E)$ and every edge e has a non-negative capacity $u(e)$. There is an algorithm, that solves maximum fractional packing of perfect matchings problem in $\mathcal{O}(n^7)$ time.*

7 Problem relations

In this section we consider relationships between problems related to packing of T-joins and perfect matchings. First we will formulate three problems about T-joins packing and we will describe relations between these problems. Then we will see same problems with perfect matchings and their relationships. In the end of this section, we will show that matchings problems are reducible to minimum cost 1-packing of T-joins problem.

In every problem, graph $G = (V, E)$ is given and every edge e has a non-negative capacity $u(e)$. We will use the next matrices for LP-formulations. Let A be a matrix whose columns are the incidence vectors of T-joins. As well as let B be a matrix whose columns are the incidence vectors of T-cuts.

First let us see the maximum packing of T-joins problem. We would like to find the maximum fractional packing of T-joins, such that e edge is used at most $u(e)$ times by the packing. In this case every T-join has a unit weight and we want to maximize the sum of their weight in the packing. This problem can be formulated as follows:

$$\max(1x) \tag{28}$$

$$Ax \leq u \tag{29}$$

$$x \geq 0 \tag{30}$$

The second problem is 1-packing problem. We want to decide either there is a packing with 1 weight or not. The answer of this decision problem is yes, if a T-joins packing can be found, where total weights is 1. This problem is decidable with the next LP-formula. (The following problem is feasible if and only if the answer is yes for the decision problem.)

$$Ax \leq u \tag{31}$$

$$1x = 1 \tag{32}$$

$$x \geq 0 \tag{33}$$

Minimum cost 1-packing problem is very similar to the second one. A non-negative cost c is given on the edges. A cost for T-joins is defined by c, that is $c(T) = \sum_{e \in T} c(e)$. We would like to find a minimum cost packing, which total weight is one. In this problem every T-join has different cost, but this cost is come from edge's cost. This problem is the special case of the more general minimum cost

1-packing problem. This problem can be written as follows:

$$\min(cx) \tag{34}$$
$$x \leq u \tag{35}$$
$$xB \geq 1 \tag{36}$$
$$x \geq 0 \tag{37}$$

Edmonds and Johnson proved that if a polyhedron defined by (36)-(37), then it is the dominant of the T-join polytope of G [5]. We use P to denote the T-join polytope. If x satisfies (36)-(37), then $x = x_1 + x_2$, where $x_1 \in P$ and $x_2 \geq 0$. Let x^* be an optimal solution of (34)-(37). Then x^* can be written as a sum $x_1^* + x_2^*$, where $x_1^* \in P$ and $x_2^* \geq 0$. Because of c is non-negative, we get $cx^* \geq cx_1^*$. Thus we can assume that x^* is contained in P. Then it can be written as a convex combination of incidence vectors of T-join. This gives a packing of T-joins in G, such that satisfies all above conditions (34)-(37).

Now we examine the relations between these problems. First two problems are equivalent.

If we can solve maximum packing problem, then 1-packing problem can be decided. This case is very simple, because we can just check whether the maximum packing is greater than one or not.

Let see the other direction. Suppose that there is an algorithm for solve 1-packing problem in polynomial time, we can decide α-packing problem for an arbitrary α. For this we make a new capacity function $u'(e) = \frac{u(e)}{\alpha}$ and apply 1-packing algorithm to get the answer α-packing problem. With this subroutine to α-packing problem, Megiddo's technique can be used for solve (28)-(30) [16].

Finally, the minimum cost 1-packing problem implies the first two problem. If we can solve (34)-(37) with $c \equiv 0$, then we can decide (31)-(33).

Let us consider same problems with perfect matchings. Let C be a matrix whose columns are the incidence vectors of perfect matchings in G.

Fourth problem is maximum packing of perfect matchings problem. We would like to find the maximum fractional packing of perfect matchings, such that e edge is used at most $u(e)$ times. This problem is formulated in the following way:

$$\max(1x) \tag{38}$$
$$Cx \leq u \tag{39}$$
$$x \geq 0 \tag{40}$$

Then let us see 1-packing of perfect matching problem. We want to decide either there is a packing with 1 weight or not. This problem is decidable with the next

LP-formula. (The following problem is feasible if and only if the answer is yes for the decision problem.)

$$Cx \leq u \tag{41}$$
$$1x = 1 \tag{42}$$
$$x \geq 0 \tag{43}$$

The last problem is minimum cost 1-packing problem. A non-negative c cost is given on the edges and every perfect matching M has a cost, such that is $c(M) = \sum_{e \in M} c(e)$. We would like to find a minimum cost packing, which total weight is one. This problem can be written as follows:

$$\min(cx) \tag{44}$$
$$Cx \leq u \tag{45}$$
$$1x = 1 \tag{46}$$
$$x \geq 0 \tag{47}$$

The relationships between problems with perfect matchings are very similar, than between related problems with T-joins.

We can reduce problem with perfect matchings to minimum cost 1-packing of T joins problem.
If we can solve minimum cost 1-packing of T-joins problem, then we can decide (41)-(43). For this we choose $T = V$ and $c \equiv 1$ cost in (34)-(37). Then we compute its optimal value, that can't be less than $\frac{n}{2}$. If it is equal to $\frac{n}{2}$, then every T-join with a positive weight in the optimal packing has $\frac{n}{2}$ edges. Otherwise, let U' be a T-joins with positive weight $w(U')$ and it has more than $\frac{n}{2}$ edges. This implies that $cx = \sum_{UT-join} c(U)W(U) = \sum_{U \neq U'} c(U)W(U) + c(U')w(U') \geq \frac{n}{2}(1 - W(U')) + c(U')w(U') > \frac{n}{2}$. This means that the optimal objective value of (34)-(37) is equal to $\frac{n}{2}$ if and only if there is a packing of perfect matchings in G, such that total weight is 1.

As well as the last problem is reducible to minimum cost 1-packing if T-joins problem. Let c be the cost, that is used in (44)-(47). Then we make a cost c' for the third problem. $c'(e) = c(e) + N$ is a good chose, where N is a big number. We say that $a_1 + b_1 N$ is less than $a_2 + b_2 N$, if $b_1 < b_2$ or $b_1 = b_2, a_1 < a_2$. Then we compute the optimal value of (34)-(37) with c'. If it is greater than $\sum_E c(e) + \frac{n}{2}N$, then the optimal solution uses at least one T-join in the optimal packing, which is not a perfect matching. Let U' be T-join with positive weight in the optimal packing of (34)-(37), such that U' has more than $\frac{n}{2}$ edges. Then for the value of this optimal

solution the following holds:

$$c'x = \sum_U c'(U)W(U) = \sum_{U \neq U'} (c(U) + |U|N)W(U) + (c(U') + |U'|N)w(U') \geq$$
$$\geq \sum_{U \neq U'} \left(c(U) + \frac{n}{2}N\right)W(U) + \left(c(U') + (\frac{n}{2} + 1)N\right)w(U') =$$
$$= \sum_{U \neq U'} c(U)W(U) + \frac{n}{2}N(1 - W(U')) + c(U')w(U') + (\frac{n}{2} + 1)Nw(U') =$$
$$= \sum_U c(U)W(U) + \left(\frac{n}{2} + w(U')\right)N > \sum_e c(e) + \left(\frac{n}{2}\right)N$$

This implies that the optimal value isn't greater than $\sum_E c(e) + \frac{n}{2}N$, then the optimal solution uses only perfect matchings in the optimal packing. This means that the minimal cost packing of (34)-(37) is a 1-packing of perfect matchings, if there exists a packing of perfect matchings in G with 1 total weight. Let x^* be a packing of perfect matchings in G. Then we have that:

$$\sum_M c'(M)x^*(M) = \sum_M (c(M) + \frac{n}{2}N)x^*(M) = \sum_M c(M)x^*(M) + \frac{n}{2}N$$

This implies that optimal solutions of (34)-(37) are the optimal solutions of (44)-(47) and the optimal value of (34)-(37) is $\frac{n}{2}N$ more than the optimal value of (34)-(37).

8 Conclusion and future work

This thesis introduced some algorithm for two kinds of problems. In the first part, we discussed balanced network problems, then we presented a strongly polynomial algorithm for balanced submodular flow problem. Thereafter in the second part of thesis, we consider some different packing problems and we showed a polynomial algorithm to find maximum packing of perfect matchings in bipartite graph.

We close this work with some question, that remain open and would be interesting to examine in the future.

Can we find the optimal balanced network flow in a graph, if an upper capacity is given along the arcs?

How can we compute the optimal value of balanced submodular flow problem with an intersecting submodular function?

How can we finish the algorithm in this case?

A network flow and submodular flow problem can be consider as a packing problem. This idea provides the next problem: We search a 1-valued packing in the graph, such that the difference between maximum edge weight and minimum edge weight is minimal. In this case capacity constraints can be given in arcs and the the optimal balanced 1-packing have to find. We can phrase questions about this kind of problems.

Can we solve different balanced 1-packing problems without capacity constraints, like balanced spanning tree problem?

What can we say about balanced packing problems with an edge capacity?

Let us see some question related to the second part of this thesis:

Can we compute the optimal value of maximum packing of perfect matching problem in a general graph?

Can we give a good characterization its dual optimal solutions?

Can we solve minimum cost 1-packing of T-joins problem, that is described in the previous section?

Can we find the optimal packing of perfect matching in a general graph?

9 References

[1] Ravindra K. Ahuja. The balanced linear programming problem. *European Journal of Operational Research*, 101(1):29–38, 1997.

[2] Francisco Barahona. Planar multicommodity flows, max cut and the chinese postman problem. *Polyhedral Combinatorics, DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, 1:189–202, 1990.

[3] Francisco Barahona. Fractional packing of t -joins. *SIAM J. Discrete Math.*, 17:661–669, 04 2004.

[4] Paolo M. Camerini, Francesco Maffioli, Silvano Martello, and Paolo Toth. Most and least uniform spanning trees. *Discrete Applied Mathematics*, 15(2):181–197, 1986.

[5] Jack Edmonds and Ellis Johnson. Matching, euler tours and the chinese postman. *Mathematical Programming*, 5:88–124, 12 1973.

[6] D. R. Fulkerson. Blocking and anti-blocking pairs of polyhedra. *Mathematical Programming*, 1:168–194, 1971.

[7] D.R Fulkerson. Anti-blocking polyhedra. *Journal of Combinatorial Theory, Series B*, 12(1):50–71, 1972.

[8] H. Gabow and K. Manu. Packing algorithms for arborescences (and spanning trees) in capacitated graphs. *Mathematical Programming*, 82:83–109, 1998.

[9] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

[10] P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, s1-10:26–30, 01 1935.

[11] A. J. Hoffman. Some recent applications of the theory of linear inequalities to extremal combinatorial analysis. *Proc. Symp. in Applied Mathematics*, pages 113–127, 1960.

[12] Maria Grazia Scutellà Klinz Bettina. A strongly polynomial algorithm for the balanced network flow problem. Technical report, TU Graz, Austria, University of Pisa, Italy, 01 2000.

[13] H. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2, 05 2012.

[14] S Martello, W.R Pulleyblank, P Toth, and D de Werra. Balanced optimization problems. *Operations Research Letters*, 3(5):275–278, 1984.

[15] S.Thomas McCormick and Thomas R. Ervolina. Computing maximum mean cuts. *Discrete Applied Mathematics*, 52:53 – 70, 1994.

[16] Nimrod Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4(4):414–424, 1979.

[17] Manfred W. Padberg and M. R. Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7(1):67–80, 1982.

[18] T. Radzik. Parametric flows, weighted means of cuts, and fractional combinatorial optimization. In *Complexity in Numerical Optimization*, 1993.

[19] Éva Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.