Algorithms and Complexity Analysis for Some Flow Problems

Edith Cohen* and Nimrod Megiddo†

Revised; November 1991

Abstract

Abstract. Several network flow problems with additional constraints are considered. They are all special cases of the linear programming problem and are shown to be $\mathcal{P}$-complete. It is shown that the existence of a strongly polynomial time algorithm for any of these problems implies the existence of such an algorithm for the general linear programming problem. On the positive side, strongly polynomial algorithms for some parametric flow problems are given, when the number of parameters is fixed. These algorithms are applicable to constrained flow problems when the number of additional constraints is fixed.

1. Introduction

An algorithm for the linear programming problem over the real numbers is called *strongly polynomial* if it performs no more than a polynomial number of elementary operations (additions, subtraction, multiplications, divisions, comparisons, and data transfers) in terms of the number of variables and constraints. If the problem is posed over the rationals, then it is also required that the algorithm be polynomial in the usual sense.

Strongly polynomial algorithms are known only for special cases of the linear programming problem. For example, Megiddo [24] gave a strongly polynomial algorithm for linear programming problems with inequality constraints, where the objective function and each of the constraints depend on at most two variables. Tardos [28] gave a strongly polynomial algorithm for linear programming problems where the entries of the constraints matrix (but not necessarily those of the objective function and the right-hand side vector) are integers bounded by a polynomial in terms of the number of variables and constraints. Her algorithm applies to many network flows problems. It is still not known, however, whether the generalized max-flow problem [16] can be solved in strongly polynomial time.

*Currently at: AT&T Bell Laboratories, 600 Mountain Av. rm. 2D-146, Murray Hill, NJ 07974. Work on the paper was done while the first author was at Stanford University and IBM Almaden Research Center.
Research partially supported by NSF PYI Grant CCR-8858097.
†IBM Almaden Research Center, San Jose, CA 95120-6099, and School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel.

Since the general linear programming problem is known to be in the class $\mathcal{P}$, it is not interesting anymore to consider polynomial time reductions of the general problem to various special cases [1, 10, 20], unless the reduction runs in strongly polynomial time. It is interesting to consider strongly polynomial reductions of the general linear programming problem to various special cases. Also, since the linear programming problem is known to be $\mathcal{P}$-complete[1] [9], it is interesting to consider logspace reductions. In this note we consider the following network flows problems, and give strongly polynomial and logspace reductions from the general linear programming problem to these problems. On the positive side, we show that some parametric flow problems with a fixed number of parameters have strongly polynomial algorithms. In particular, we discuss strongly polynomial special cases of problems described below.

Problem 1.1 [Balanced Costs Circulation] Given is a digraph $G = (V, E)$ with real costs per unit of flow $c_e \in R$ $(e \in E)$ associated with the edges. Find a nontrivial circulation $\boldsymbol{x} = (x_e)_{e\in E}$, i.e., $x_e \geq 0$ $(e \in E)$, $\boldsymbol{x} \neq \mathbf{0}$, and for every $v \in V$,

$$\sum_{i:(i,v)\in E} x_{iv} = \sum_{i:(v,i)\in E} x_{vi} \ ,$$

such that the "cost" is balanced at the vertices, that is, for every vertex $v \in V$,

$$\sum_{i:(i,v)\in E} x_{iv}c_{iv} = \sum_{i:(v,i)\in E} x_{vi}c_{vi} \ .$$

The set of circulations with balanced costs is invariant under the operation of adding the same constant to each of the costs.

The problem of a flow in a network with pairs of *homologous edges* is defined as follows. A set of pairs of edges is given and one has to find a flow where members of each pair carry equal amounts of flow. The latter problem and a related one of networks with bundles were studied by Berge and Ghouila-Houri [2], Ghouila-Houri [14, 15], and Hoffman [19]. The result of [28] implies that these problems are solvable in strongly polynomial time. The problem of a flow in a network with pairs of homologous edges can be generalized as follows into a problem for which it is still not known whether a strongly polynomial time algorithm exists.

Problem 1.2 [Fixed Ratios Circulation] Let $G = (V, E)$ be a digraph, let $\ell_e, u_e \in R_+$ $(e \in E)$ be lower and upper bounds, respectively, on the flow in an edge e, and let $\alpha : S \to R_+$, where $S \subset E \times E$. Find a nontrivial circulation $\boldsymbol{x} = (x_e)_{e\in E}$, i.e., $\ell_e \leq x_e \leq u_e$, and for every $v \in V \setminus \{s, t\}$,

$$\sum_{i:(i,v)\in E} x_{iv} = \sum_{i:(v,i)\in E} x_{vi} \quad ,$$

such that for every pair $(e_1, e_2) \in S$, $x_{e_1} = \alpha(e_1, e_2)x_{e_2}$.

[1] complete for the class $\mathcal{P}$ under logspace reductions

Remark 1.3 The constraints of the form $x_{e_1} = \alpha(e_1, e_2)x_{e_2}$ imply by transitivity more constraints of this type, so we may assume without loss of generality that S is transitively closed, so that if $\{(e_1, e_2), (e_2, e_3)\} \subset S$, then $(e_1, e_3) \in S$ and $\alpha(e_1, e_3) = \alpha(e_1, e_2)\alpha(e_2, e_3)$. Similarly, we may assume without loss of generality that if $(e_1, e_2) \in S$, then $(e_2, e_1) \in S$ and $\alpha(e_2, e_1) = 1/\alpha(e_1, e_2)$. Thus, we assume S is an equivalence relation over E.

Problem 1.4 [Circulation with Fixed Forks] Given is a digraph $G = (V, E)$, where V is partitioned into three sets U, F_{in} and F_{out}. The vertices in F_{in} (resp., F_{out}) have in-degree 2 and out-degree 1 (resp., out-degree 2 and in-degree 1). Also given is a function $\alpha : F_{in} \cup F_{out} \to [0, 1]$. The vertices F_{in} (resp., F_{out}) are called *in-forks* (resp., *out-forks*). Find a nontrivial circulation $\boldsymbol{x}$ such that

(i). if $v \in F_{out}$ and $\{(i, v), (v, j), (v, k)\} \subset E$, and $j < k$, then

$$x_{vj} = \alpha(v)x_{iv} \quad \text{and} \quad x_{vk} = (1 - \alpha(v))x_{iv} \ ,$$

(ii). if $v \in F_{in}$ and $\{(v, i), (j, v), (k, v)\} \subset E$ and $j < k$, then

$$x_{jv} = \alpha(v)x_{vi} \quad \text{and} \quad x_{kv} = (1 - \alpha(v))x_{vi} \ .$$

Problem 1.5 If the graph G is bipartite and all the edges are between U and $F = F_{in} \cup F_{out}$, then we refer to the problem as the bipartite version of Problem 1.4.

We show that the existence of a strongly polynomial algorithm for any of the problems stated above implies the existence of a strongly polynomial algorithm for Problem 1.5. Also, these problems are $\mathcal{P}$-complete. This is clearly the case for Fixed Forks. It is also easy to see that a circulation with fixed forks is a special case of a circulation with fixed ratios. In Section 2 we give a strongly polynomial time and $\mathcal{NC}$ reduction[2] from Problem 1.5 to the Balanced Costs Circulation problem. In Section 3 we show that the existence of a strongly polynomial algorithm for Problem 1.5 implies the existence of such an algorithm for the general linear programming problem. Also, Problem 1.5 is $\mathcal{P}$-complete. The combination of these results implies the following theorem.

Theorem 1.6 *For each of the following: (i) Circulation with Balanced Costs (Problem 1.1), (ii) Circulation with Fixed Ratios (Problem 1.2), and (iii) Circulation with Fixed Forks (Problems 1.4 and 1.5), the problem has a strongly polynomial algorithm if and only if the general linear programming problem has one. Moreover, all these problems are $\mathcal{P}$-complete.*

In Section 4 we review a technique developed by the authors in [6] (see also [4, 5, 7]). We apply the technique to obtain strongly polynomial algorithms for parametric flow problems when the number of parameters is fixed. If the number of parameters is not fixed then these parametric problems are $\mathcal{P}$-complete, and the existence of a strongly polynomial algorithm for any of them is equivalent to existence of such an algorithm for the general linear

[2]An $\mathcal{NC}$ reduction is one that can be carried out in polylogarithmic time with a polynomial number of processors.

programming problem. The existence of strongly polynomial algorithms for parametric problems with a fixed number of parameters has consequences as follows. If either the number of "forks" in Problem 1.4 is fixed, or the number of equivalence classes in S of Problem 1.2 is fixed, then the respective problems can be solved in strongly polynomial time. Moreover, similar generalizations of the maximum flow and the minimum-cost flow problems also have strongly polynomial algorithms. Some of the results of Section 4 also follow from a scheme very similar to that of [6], which was introduced later by Norton, Plotkin, and Tardos [26].

2. Bipartite Fixed Forks is reducible to Balanced Costs

Proposition 2.1 *The Bipartite Fixed Forks problem is reducible both in strongly polynomial time and in $\mathcal{NC}$ to the Balanced Costs problem.*

Proof: Consider an instance of Problem 1.4 on a graph $G = (V, E)$ with costs c_e $(e \in E)$, where $V = U \cup F_{in} \cup F_{out}$. We reduce the problem to an instance of Problem 1.1. Define $G' = (V', E')$ as follows. Let V^1 and V^2 be two copies of V, and let W^1 and W^2 be two copies of E. Let

$$V' = V^1 \cup V^2 \cup W^1 \cup W^2 \ .$$

For every $v \in V$ we denote by $v^1 \in V^1$ and $v^2 \in V^2$ the corresponding vertices. For every $e \in E$ we denote by $w^1_e \in W^1$ and $w^2_e \in W^2$ the vertices that correspond to e.

We now associate weights c^*_e $(e \in E)$ with the edges of G. Consider a vertex $v \in F_{in}$ (resp., $v \in F_{out}$) with incident edges (v, i), (j, v), and (k, v) (resp., (i, v), (v, j), and (v, k)), where $j > k$. Let $c^*_{vi} = 1$, $c^*_{jv} = \alpha(v)$, and $c^*_{kv} = 1 + \alpha(v)$ (resp., $c^*_{iv} = 1$, $c^*_{vj} = \alpha(v)$, and $c^*_{vk} = 1 + \alpha(v)$).

To each edge $e \in E$ there correspond five edges in E'. If $e = (i, v)$ where $i \in U$ and $v \in F = F_{in} \cup F_{out}$, then the corresponding edges are (See Figure 1.):

(i). $e_1 = (w^1_e, v^1)$ with cost c^*_e,

(ii). $e_2 = (v_2, w^2_e)$ with cost c^*_e,

(iii). $e_3 = (i^1, w^1_e)$ with cost 0,

(iv). $e_4 = (w^2_e, i^2)$ with cost 0,

(v). $e_5 = (w^2_e, w^1_e)$ with cost $2c^*_e$.

If $e = (v, i)$ with $i \in U$ and $v \in F$, then the corresponding edges are:

(i). $e_1 = (v^1, w^1_e)$ with cost c^*_e,

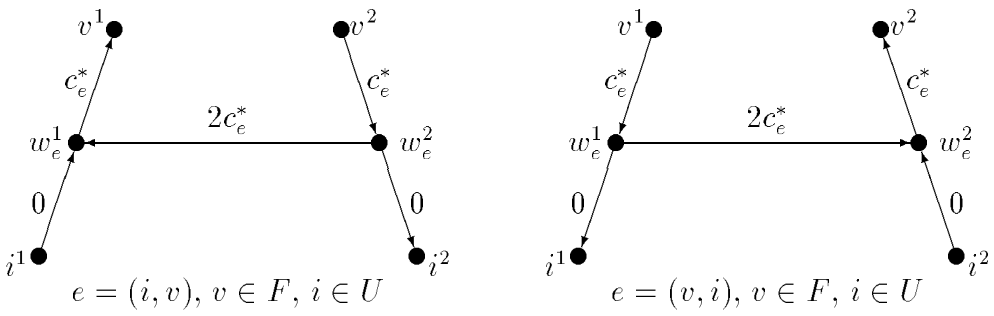

Figure 1: The replacement in G' of an edge $e \in E$.

(ii). $e_2 = (w_e^2, v^2)$ with cost c_e^*,

(iii). $e_3 = (w_e^1, i^1)$ with cost 0,

(iv). $e_4 = (i^2, w_e^2)$ with cost 0,

(v). $e_5 = (w_e^1, w_e^2)$ with cost $2c_e^*$.

It is easy to see that a flow $\boldsymbol{x}$ has balanced costs at the vertices w_e^1 and w_e^2 if and only if $x_{e_2} = x_{e_1}$ and $x_{e_3} = x_{e_4} = x_{e_5} = \frac{1}{2}x_{e_1}$.

If $u \in U$, then all the edges of E' incident on either u^1 or u^2 have zero cost, and hence every flow has balanced costs at the vertices u^1 and u^2.

Consider a vertex $v \in F_{in}$ (resp., $v \in F_{out}$) with incident edges $e = (v, i)$, $e' = (j, v)$, and $e'' = (k, v)$ (resp., $e = (i, v)$, $e' = (v, j)$, and $e'' = (v, k)$), where $j > k$. A flow $\boldsymbol{x}$ has balanced costs at the vertices v^1 and v^2 if and only if

$$x(e_1) = \alpha(v)x(e_1') + (1 - \alpha(v))x(e_1'') \quad \text{and} \quad x(e_2) = \alpha(v)x(e_2') + (1 - \alpha(v))x(e_2'') \ .$$

It is easy to see that a flow $\boldsymbol{x} = (x_e)$ is feasible for Problem 1.4 in G if and only if the flow $\boldsymbol{x}'$ defined by $x'_{e_1} = x_e$ $(e \in E)$ has balanced costs in G'. ∎

3. Linear Programming reduces to Bipartite Fixed Forks

In this section we present the following theorem.

Theorem 3.1 *The Bipartite Fixed Forks problem (Problem 1.5) has a strongly polynomial algorithm if and only if the general linear programming problem has one. Moreover, the former is $\mathcal{P}$-complete.*

For the proof of the theorem, we show that all the following problems have logspace and strongly polynomial time reductions to each other:

(i). Given a matrix $\boldsymbol{A} \in R^{m\times d}$ and a vector $\boldsymbol{b} \in R^m$, decide whether there exists an $\boldsymbol{x}$ such that $\boldsymbol{Ax} \le \boldsymbol{b}$.

(ii). Given $\boldsymbol{A}$ and $\boldsymbol{b}$ as above, decide whether there exists an $\boldsymbol{x}$ such that $\boldsymbol{Ax} < \boldsymbol{b}$.

(iii). Given $\boldsymbol{A}$ as in (i), where for $i = 1, \dots, m$, $\sum_{j=1}^{d} A_{ij} = 0$, decide whether there is an $\boldsymbol{x}$ such that $\boldsymbol{Ax} < \boldsymbol{0}$.

(iv). Given $\boldsymbol{A}$ as in (iii), where in addition, for $i = 1, \dots, m$, $\max_{1\le j\le d} |A_{ij}| = 1$, and each row has at most three nonzero entries, decide whether there is an $\boldsymbol{x}$ such that $\boldsymbol{Ax} < \boldsymbol{0}$.

(v). Given $\boldsymbol{A}$ as in (iv), decide whether there is a $\boldsymbol{y}$ such that $\boldsymbol{A}^T\boldsymbol{y} = \boldsymbol{0}$, $\boldsymbol{y} \ge \boldsymbol{0}$, and $\boldsymbol{y} \ne \boldsymbol{0}$.

(vi). The Bipartite Fixed Forks problem (Problem 1.5).

Problem 1.4 is a special case of the linear programming problem, and hence (vi), as a special case of Problem 1.4, is reducible to (i). Proposition 3.2 gives the reduction from (i) to (ii), Proposition 3.3 reduces (ii) to (iii), Proposition 3.4 reduces (iii) to (iv), Proposition 3.5 (Gordan [17]) reduces (iv) to (v), and Proposition 3.6 reduces (v) to (vi).

Proposition 3.2 *The problem of deciding feasibility of a system of weak linear inequalities $\boldsymbol{Ax} \le \boldsymbol{b}$ can be reduced in strongly polynomial time to the problem of deciding feasibility of a system of strict linear inequalities inequalities $\boldsymbol{A}'\boldsymbol{x} < \boldsymbol{b}'$ Also, the latter is $\mathcal{P}$-complete.*

Proof: Given $\boldsymbol{A} \in R^{m\times d}$ and $\boldsymbol{b} \in R^m$, denote $P = \{\boldsymbol{x} \in R^d \mid \boldsymbol{Ax} \le \boldsymbol{b}\}$. Suppose we have an oracle which decides for any matrix $\boldsymbol{A}'$ and any vector $\boldsymbol{b}'$ whether $\boldsymbol{A}'\boldsymbol{x} < \boldsymbol{b}'$ is feasible. We use the oracle to decide whether $\boldsymbol{Ax} \le \boldsymbol{b}$ is feasible. Consider the following iterative step. If $\boldsymbol{Ax} < \boldsymbol{b}$ is feasible then obviously so is $\boldsymbol{Ax} \le \boldsymbol{b}$. Otherwise, there exists a row $A_{j\bullet}$ such that $A_{j\bullet}\boldsymbol{x} = b_j$ for all $\boldsymbol{x} \in P$, which can be found as follows. For every k ($k = 1, \dots, m$), consider the system S_k, consisting of the inequalities $A_{i\bullet}\boldsymbol{x} < b_i$ ($i = 1, \dots, k$). Let j be the first index such that S_j is infeasible.

Let $\boldsymbol{A}' \in R^{(m-1)\times(d-1)}$ and $\boldsymbol{b}' \in R^{m-1}$ be the system generated when one variable is eliminated, using the equality $A_{j\bullet}\boldsymbol{x} = b_j$. The system $\boldsymbol{A}'\boldsymbol{x} \le \boldsymbol{b}'$ is feasible if and only if the system $\boldsymbol{Ax} \le \boldsymbol{b}$ is. We repeat the same process with $\boldsymbol{A}'$ and $\boldsymbol{b}'$. If only one variable remains, we test the feasibility of the system directly. Otherwise, repeat the iterative step. The $\mathcal{P}$-completeness of the problem of deciding a system of strict linear inequalities follows by a simple adaptation of the proof of [9]. ∎

Proposition 3.3 *Given are a matrix $\boldsymbol{A} \in R^{m\times d}$ and a vector $\boldsymbol{b} \in R^m$. In $O(md)$ time (and in $\mathcal{NC}$) we can compute a matrix $\boldsymbol{A}' \in R^{(m+1)\times(d+2)}$ with the following properties:*

(i). $\sum_{j=1}^{d+2} A'_{ij} = 0$ for $i = 1, \dots, m$.

(ii). There exists an $\boldsymbol{x} \in R^{d+2}$ such that $\boldsymbol{A}'\boldsymbol{x} < \boldsymbol{0}$ if and only if there exists an $\boldsymbol{x} \in R^d$ such that $\boldsymbol{Ax} < \boldsymbol{b}$.

Proof: Let $F \subset R^{d+2}$ be the d-dimensional flat

$$F = \{\boldsymbol{x} \in R^{d+2} \mid x_{d+2} - x_{d+1} = 1 \ , \ \sum_{i=1}^{d+2} x_i = 1\} \ .$$

Let $M : R^d \to F$ be the affine transformation defined by

$$(M(\boldsymbol{x}))_i = \begin{cases} x_i & \text{if} \quad 1 \le i \le d \\ -\frac{1}{2}\mathbf{e}^T\boldsymbol{x} & \text{if} \quad i = d+1 \\ 1 - \frac{1}{2}\mathbf{e}^T\boldsymbol{x} & \text{if} \quad i = d+2 \end{cases}$$

Denote by $M^{-1} : F \to R^d$ the inverse transformation, i.e., $(M^{-1}(\boldsymbol{x}))_i = x_i$ $(i = 1, \ldots, d)$. Let $L : R^d \times R \to R^{d+2}$ be defined as follows.

$$(L(\boldsymbol{a}, b))_i = \begin{cases} a_i + \frac{1}{2}w - \frac{1}{2}b & \text{if} \quad 1 \le i \le d \\ w & \text{if} \quad i = d+1 \\ -b & \text{if} \quad i = d+2 \end{cases}$$

where $w = b - \frac{2}{d+2}\mathbf{e}^T\boldsymbol{a}$. It is easy to verify that $(L(\boldsymbol{a}, b))^T M(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x} - b$, and $\mathbf{e}^T L(\boldsymbol{a}, b) = 0$. Therefore, for all vectors $\boldsymbol{x}, \boldsymbol{a} \in R^d$ and $b \in R$ we have $\boldsymbol{a}^T\boldsymbol{x} < b$ if and only if $(L(\boldsymbol{a}, b))^T M(\boldsymbol{x}) < 0$. Define $\boldsymbol{A}'$ as follows. For $i = 1, \ldots, m$, let $A'_{i\bullet} = (L(A_{i\bullet}, b_i))^T$. The $(m+1)$-st row of $\boldsymbol{A}'$ is defined so as to represent the constraint $x_{d+2} > x_{d+1}$.

Denote

$$P = \{\boldsymbol{x} \in R^d \mid \boldsymbol{A}\boldsymbol{x} < \boldsymbol{b}\}$$
$$P' = \{\boldsymbol{x} \in R^{d+2} \mid \boldsymbol{A}'\boldsymbol{x} < \boldsymbol{0}\} \ .$$

We need to show that $P = \emptyset$ if and only if $P' = \emptyset$. It is easy to verify that $P = \emptyset$ if and only if $P' \cap F = \emptyset$. It remains to be shown that if $P' \neq \emptyset$ then $F \cap P' \neq \emptyset$. Note that if $\boldsymbol{x} \in P'$, then for every $\alpha > 0$, $\alpha\boldsymbol{x} \in P'$, and for any α, $\boldsymbol{x} - \alpha\boldsymbol{e} \in P'$. Suppose that $\boldsymbol{y} \in P'$, it follows that for $\boldsymbol{y}' = 1/(y_{d+2} - y_{d+1})\boldsymbol{y}$, $y'_{d+2} - y'_{d+1} = 1$. It is easy to verify that $\boldsymbol{y}'' = \boldsymbol{y}' + ((\mathbf{e}^T\boldsymbol{y}' - 1)/d)\boldsymbol{e} \in P'$, $\mathbf{e}^T\boldsymbol{y}'' = 1$, and $y''_{d+2} - y''_{d+1} = 1$. Hence, $\boldsymbol{y}'' \in P' \cap F$. ∎

Proposition 3.4 *Suppose $\boldsymbol{A} \in R^{m \times d}$ is a matrix where $\sum_{j=1}^d A_{ij} = 0$, for $i = 1, \ldots, m$. In $O(md)$ time (and in $\mathcal{NC}$), we can compute a matrix $\boldsymbol{A}' \in R^{m' \times d}$, where $m' = O(md)$, such that*

(i). $\sum_{j=1}^d A'_{ij} = 0$ $(i = 1, \ldots, m')$,

(ii). each row of $\boldsymbol{A}'$ have at most three nonzero entries,

(iii). for every row of $\boldsymbol{A}'$, the maximum absolute value of an entry in the row is 1, and

(iv). there exists an $\boldsymbol{x}$ such that $\boldsymbol{A}'\boldsymbol{x} < \boldsymbol{0}$ if and only if there exists an $\boldsymbol{x}$ such that $\boldsymbol{A}\boldsymbol{x} < \boldsymbol{0}$.

Proof: We first show how to convert the system to an equivalent one where each constraint involves at most three variables, and the sum of the coefficients is 0. Consider an inequality of the form $\sum_{i=1}^{k} a_i x_i < 0$ where $a_i \neq 0$ $(i = 1, \ldots, k)$ and $\sum_{i=1}^{k} a_i = 0$. Without loss of generality, assume $\sum_{i=3}^{k} \neq 0$. It is easy to verify that $(x_1, \ldots, x_k)$ satisfies the inequality $\sum_{i=1}^{k} a_i x_i < 0$ if and only if there exists a scalar ξ such that

$$a_1 x_1 + a_2 x_2 + \xi \sum_{i=3}^{k} a_i < 0 \quad \text{and} \quad \xi \sum_{i=3}^{k} a_i > \sum_{i=3}^{k} a_i x_i \ .$$

These two latter inequalities have 3 and $k-1$ variables respectively, and the sum of the coefficients in each of them is 0. Hence, by considering all constraints with $k > 3$ variables and repeating this step $k-3$ times we get an equivalent system with properties (i) and (ii). This can be done in a poly-logarithmic number of phases. To conclude, we note that by multiplying a row by a positive constant, or omitting rows where all entries are 0, we do not alter the feasibility of the system. Thus, we may divide any row by the largest absolute value of an entry in the row. ∎

Proposition 3.5 [Gordan [17]] *For any $\boldsymbol{A} \in R^{m \times d}$, there exists an $\boldsymbol{x} \geq \boldsymbol{0}$, $\boldsymbol{x} \neq \boldsymbol{0}$, such that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$, if and only if there is no $\boldsymbol{y}$ such that $\boldsymbol{A}^T \boldsymbol{y} > \boldsymbol{0}$.*

Proposition 3.6 *Suppose $\boldsymbol{A} \in R^{d \times m}$ satisfies*

(i). $\sum_{i=1}^{d} A_{ij} = 0$, for $j = 1, \ldots, m$,

(ii). each column of $\boldsymbol{A}$ has at most three nonzero entries,

(iii). in each column, the largest absolute value of an entry in the column is 1.

Under these conditions, we can construct in $O(m+d)$ time an instance BFF($\boldsymbol{A}$) *of the Bipartite Fixed Forks Problem (Problem 1.5) such that there is a one-to-one correspondence between the circulations that solve* BFF($\boldsymbol{A}$) *and vectors $\boldsymbol{y} \geq \boldsymbol{0}$ for which $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{0}$.*

Proof: With each matrix $\boldsymbol{A}$ as above we associate an instance BFF($\boldsymbol{A}$) of Problem 1.5 as follows. The vertices of the bipartite digraph correspond to the columns and rows of $\boldsymbol{A}$, and the edges correspond to the nonzero entries of $\boldsymbol{A}$. The set U ($|U| = d$) consists of vertices which correspond to the rows, and the set $F = F_{in} \cup F_{out}$ ($|F| = m$) consists of vertices which correspond to the columns. If a vertex $v \in F$ corresponds to a column of $\boldsymbol{A}$ with exactly one positive entry, then $v \in F_{out}$; otherwise, $v \in F_{in}$. If the entry A_{ij} is positive, (resp., negative) place an edge from the vertex which corresponds to the ith row (resp., jth column) to the vertex corresponding to the jth column (resp., ith row). Construct the function $\alpha : F \to [0,1]$ as follows. Consider a column $A_{\bullet j}$ and the corresponding vertex v. Let $\alpha(v) = |A_{kj}|$ where k is the smallest number such that $|A_{kj}| \notin \{0,1\}$. If such a k does not exist, then the column has two nonzero entries $\{-1, 1\}$, and we choose $\alpha(v) = 1$.
What remains is to verify that there is a one-to-one correspondence between circulations $x : E \to R_+$ which solve BFF($\boldsymbol{A}$) and vectors $\boldsymbol{0} \leq \boldsymbol{y} \in R^m$ such that $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{0}$. Choose edges

$e_1, \ldots, e_m$ such that e_j corresponds to an entry with absolute value 1 in the jth column ($j = 1, \ldots, m$). (If there are two such entries take either one.) Consider a circulation x which solves BFF($\boldsymbol{A}$). It is easy to see that the flow values on the edges $e_1, \ldots, e_m$ uniquely determine the flow on the rest of the graph.
To conclude the proof, note the following facts: (i) The vector $\boldsymbol{y}$, defined by $y_j = x(e_j)$ ($j = 1, \ldots, m$), satisfies $\boldsymbol{Ay} = \boldsymbol{0}$. (ii) If $\boldsymbol{y} \geq \boldsymbol{0}$ is a vector such that $\boldsymbol{Ay} = \boldsymbol{0}$, then there exists a unique solution $\boldsymbol{x}$ for BFF($\boldsymbol{A}$) such that $x(e_j) = y_j$ ($j = 1, \ldots, m$). ∎

4. Algorithms for parametric flow problems

The authors have obtained in [6] a strongly polynomial time algorithm for the parametric minimum cycle problem with a fixed number of parameters (see [5, 4] for a full version). The scheme used in [6] is fairly general and applies to parametric extensions of other problems. Norton, Plotkin, and Tardos [26] used a similar scheme to obtain strongly polynomial time algorithms for various other problems. In particular, they showed how under certain conditions, a strongly polynomial algorithm for a class of linear programming problems can be extended to handle a fixed number of additional constraints and variables. When additional variables are considered, both the scheme of [26] and a direct application of the scheme of [6], assume the existence of a strongly polynomial algorithm that works for arbitrary right-hand side vectors. The scheme of [26] requires, in addition, a strongly polynomial algorithm for the dual problem for any right-hand side vector of the dual, i.e., any objective function of the primal problem. If a fixed number of constraints are added, the scheme of [6] also requires a strongly polynomial algorithm for the dual problem. Consider, for example, the linear programming formulation of the max-flow problem. An arbitrary right-hand side vector can be interpreted as a set of arbitrary supply and demand constraints associated with the vertices. Assuming an arbitrary objective function means we are looking at the general *min-cost* flow problem. In subsection 4.1 we summarize the scheme of [6] as presented in [5]. In subsection 4.2 we apply this scheme to get strongly polynomial time algorithms for some parametric extensions of flow problems with a fixed number of parameters. The linear programming formulations of these parametric problems yield flow problems with additional variables.

The extensions of algorithms have interesting implications with regard to the Fixed Ratios and the generalized flow problems. In subsection 4.3 we show that strongly polynomial algorithms exist for the generalized flow and min-cost generalized flow problems, when the number of edges with gains or losses is fixed, and for the Fixed Ratios and min-cost with Fixed Ratios problems, when the number of equivalence classes of edges is fixed. The max-flow problem with a fixed number of forks is a special case of the problem of max-flow with fixed. These problems can be represented as linear programs of max-flow with a fixed number of additional variables. Strongly polynomial time algorithms for these problems can also be derived using Theorem 3.1 of [26].

4.1. Review of a scheme for strongly polynomial algorithms

For the sake of completeness, we review the scheme used in [6, 5]. We also discuss how it applies to obtain strongly polynomial algorithms for parametric extensions of problems.

An algorithm that computes a function $g : R^d \to R$ is called *piecewise affine* if all the operations it performs on intermediate values that depend on the input vector are additions, multiplications by constants, comparisons, and making copies.

The main tool we use can be stated as follows. Consider a piecewise affine algorithm $\mathcal{A}$ that computes values of a concave function $g : \mathcal{Q} \to R$, where the domain $\mathcal{Q} \subset R^d$ is given as an intersection of k halfspaces. We assume that $\mathcal{A}$ is accessible in a way which allows us to follow the computation path for any given input. We also assume that $\mathcal{A}$ performs at most T operations including C comparisons. Furthermore, the C comparisons can be organized into r phases, where all the comparisons within a single phase are independent. We denote the number of comparisons in phase i by C_i ($i = 1, \ldots, r$).

Theorem 4.1 *The function g can be maximized using*

$$kT \left(\sum_{i=1}^{r} \lceil \log C_i \rceil \right)^d$$

operations.

The scheme given in [6, 5] integrates techniques from [23, 25].

Theorem 4.1 also holds when the range of g is R^ℓ ($\ell > 1$) and the notions of maximum and concavity are defined with respect to the lexicographic order as follows. The function g is concave with respect to the lexicographic order if for every $\alpha \in [0, 1]$ and $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{Q}$, $\alpha g(\boldsymbol{x}) + (1 - \alpha) g(\boldsymbol{y}) \leq_{\text{lex}} g(\alpha \boldsymbol{x} + (1 - \alpha)\boldsymbol{y}))$.

Remark 4.2 Note that if the conditions hold for $g : \mathcal{Q}$, then they also hold for a restriction of g to a polyhedron $\mathcal{Q}' \subset \mathcal{Q}$.

We now discuss how and when Theorem 4.1 can be applied to obtain strongly polynomial algorithms for parametric extensions of problem (with a fixed number of parameters). We view a *problem* $S : \mathcal{P} \to R^\ell$ as a mapping from a set $\mathcal{P}$ of instances into ℓ-tuples of real numbers. We say that $S(P)$ is the *solution* of the problem for the instance $P \in \mathcal{P}$. A *d-parametric extension* of $\mathcal{P}$ has the form $\mathcal{P}^d = (\mathcal{M}, \mathcal{Q})$ where (i) $\mathcal{Q} \subset R^d$ is a polyhedron given as an intersection of halfspaces, and (ii) $\mathcal{M} : \mathcal{Q} \to \mathcal{P}$ is a mapping from points $\boldsymbol{\lambda} \in \mathcal{Q} \subset R^d$ to instances of $\mathcal{P}$. Each parametric instance $P^d \in \mathcal{P}^d$ corresponds to a subset of instances $\{\mathcal{M}(\boldsymbol{\lambda}) \mid \boldsymbol{\lambda} \in \mathcal{Q}\} \subseteq \mathcal{P}$. We refer to $\mathcal{M}(\boldsymbol{\lambda}) \in \mathcal{P}$ as the *instance induced by* $\boldsymbol{\lambda}$. By a *solution of the parametric problem* for an instance $P^d = (\mathcal{M}, \mathcal{Q}) \in \mathcal{P}^d$ we mean as follows. Consider the maximum (relative to the lexicographic order) over all possible values of the parameters $\boldsymbol{\lambda} \in \mathcal{Q}$, of the solutions of the induced instance $\mathcal{M}(\boldsymbol{\lambda})$. If the maximum is finite, then a solution consists of the maximum and a vector $\boldsymbol{\lambda} \in R^d$ that belongs to the relative interior of the set of parameter values which maximize S. Formally, to solve an instance of $P^d \in \mathcal{P}^d$ we have to do the following: if either $\mathcal{Q} = \emptyset$ or $S(\mathcal{M}(\boldsymbol{\lambda}))$ is unbounded on $\mathcal{Q}$, then these facts have to be recognized; otherwise, a pair $(m, \boldsymbol{\lambda}^*) \in R \times R^d$, where $m = \max_{\lambda \in \mathcal{Q}} S(\mathcal{M}(\boldsymbol{\lambda}))$ and $\boldsymbol{\lambda}^* \in \operatorname{rel\,int}\{\boldsymbol{\lambda} \mid S(\mathcal{M}(\boldsymbol{\lambda})) = m\}$, has to be computed.

For a parametric instance P^d, consider the function $g : \mathcal{Q} \to R^\ell$ defined as $g(\boldsymbol{\lambda}) = S(\mathcal{M}(\boldsymbol{\lambda}))$. Note that the solution of P^d is a vector which maximizes g. Theorem 4.1 can be applied for solving P^d if we are given an algorithm $\mathcal{A}$ which computes $S(\mathcal{M}(\boldsymbol{\lambda}))$, and the conditions of the theorem hold for $\mathcal{A}$ and g.

Remark 4.3 When we attempt to apply the method described above to obtain strongly polynomial time bounds, we may encounter a difficulty as follows. Sometimes we may have a problem that satisfies the conditions of Theorem 4.1, but the polyhedron $\mathcal{Q}$ (i.e., the domain of g) is either not given explicitly or has a super-polynomial number of facets. The latter occurs in the parametric minimum cycle problem. A simplified version of the parametric minimum cycle problem used in [6] is as follows.

> Let $G = (V, E)$ be a digraph where affine forms of d variables are associated with the edges, so the corresponding function g maps sets of values for the parameters $\boldsymbol{\lambda} \in R^d$ to the value of the minimum weight cycle in the graph with the induced scalar weights. Compute a $\boldsymbol{\lambda} \in R^d$ at which the value of the minimum weight cycle relative to the induced scalar weights is maximized.

The function g is not defined at vectors $\boldsymbol{\lambda}$ for which the graph has negative cycles relative to the induced weights. The weight of a cycle c is an affine form. Hence, we can compute a halfspace H_c such that the weight of the cycle relative to $\boldsymbol{\lambda}$ is nonnegative if and only if $\boldsymbol{\lambda} \in H_c$. The domain of g is the intersection of the halfspaces H_c, where c is a simple cycle in G. We now show that the number of facets in the domain of g may be $n^{\Omega(\log n)}$. Carstensen [3] constructed a family of acyclic graphs, with affine forms of one parameter λ associated with the edges, such that the parametric shortest (s,t)-path function has $n^{\Omega(\log n)}$ breakpoints. We consider the graphs of Carstensen's construction, with an additional edge (t,s), with the affine form $-\mu + c$ (where c is some constant) associated with it. The resulting graphs have two-parameter affine forms associated with the edges and cycles which consist of an (s,t)-path and the edge (t,s). For each graph, consider the polygon consisting of all the values of λ and μ for which there are no negative cycles. It is easy to see that we can choose, independently for each graph, a sufficiently large constant c such that these polygons have $n^{\Omega(\log n)}$ edges. Gusfield [18] showed that the number of facets in a parametric shortest (s,t)-path function is $n^{O(\log n)}$. Using a similar argument, it can be shown that the same holds for parametric minimum cycle. Obviously, we want to avoid computing the domain of g. In order to overcome this difficulty, we "extended" the domain of the function g to R^{d-1}. We note that in [6] we asked for a minimum cycle of at most n edges in the graph. This parametric problem is well defined for any $\boldsymbol{\lambda} \in R^d$. The corresponding function g' coincides with g on the domain of g and is concave. Moreover, g' is maximized in the domain of g if the latter is not empty. It follows that we can apply Theorem 4.1 to maximize g'. Below we consider concave extensions of functions where the extended domains are polyhedra with polynomial numbers of facets. Another way to optimize a function g over a domain with superpolynomially many facets is to construct a "separation oracle", i.e., an algorithm that for a given $\boldsymbol{\lambda}$ returns a halfspace that (i) contains the domain of g and (ii) has $\boldsymbol{\lambda}$ on its boundary. The use of a separation oracle is required by the scheme of [26] and can be easily incorporated into the scheme of [6].

4.2. Parametric extensions of flow problems

The max-flow problem [12] has well-known strongly polynomial algorithms [8, 11]. We consider parametric extensions of the max-flow problem where the capacities and the supplies and demands at the vertices are replaced by affine forms of d variables. A vector $\boldsymbol{\lambda} \in R^d$ corresponds to a set of values of the parameters. We refer to the resulting capacities as the *capacities induced by* $\boldsymbol{\lambda}$. We discuss problems such as finding $\boldsymbol{\lambda} \in R^d$ which (i) maximizes the max flow relative to the induced capacities, or (ii) allows for a feasible circulation subject to the induced capacities. Parametric flow problems were previously considered by Gusfield [18], and by Gallo, Grigoriadis and Tarjan [13].

We first consider the max-flow problem where the capacities are affine forms with d variables and we want to find a $\boldsymbol{\lambda} \in R^d$ that maximizes the max-flow relative to the induced capacities.

Problem 4.4 [Max-Flow with Parametric Capacities] Let $G = (V, E)$ be a digraph, let $s, t \in V$ be two distinguished vertices, and let c_e ($e \in E$) be d-variable affine forms ("parametric capacities") associated with the edges. Find a $\boldsymbol{\lambda} \in R^d$ which maximizes the maximum (s, t)-flow relative to the induced capacities.

Proposition 4.5 *The parametric max-flow problem (Problem 4.4) can be solved in strongly polynomial time for any fixed number of parameters.*

Proof: Let $g(\boldsymbol{\lambda})$ be the value of the maximum flow relative to the capacities induced by $\boldsymbol{\lambda}$. The domain of g is the intersection of the halfspaces which guarantee that all the edges have nonnegative capacities. Denote the domain of g by $\mathcal{Q}$. We show that the function g is concave. The maximum flow equals the minimum capacity of a cut. The capacity of each cut is an affine form of $\boldsymbol{\lambda}$. Hence, the minimum cut is a piecewise linear and concave function of $\boldsymbol{\lambda}$. It is easy to verify that all the other conditions of Theorem 4.1 are satisfied as well. ∎

Consider a network with both upper and lower bounds on the flow in each edge. Both bounds are replaced by affine forms. As in Problem 4.4, the goal is to find a vector $\boldsymbol{\lambda} \in R^d$ relative to which the max-flow in the induced network is maximized. This problem is called *max-flow with parametric bounds.*

We wish to find a strongly polynomial time algorithm for the problem that is based on a piecewise affine max-flow algorithm. This problem, however, suffers from a difficulty similar to the one described in Remark 4.3. The domain of the search consists of values of $\boldsymbol{\lambda} \in R^d$ for which the induced network has a feasible circulation. We require that for all edges, the upper bounds are not smaller than the lower bounds, and they are both nonnegative. To guarantee feasibility, we need additional requirements (see Hoffman [19]) as follows. We need to assure that all the (s,t)-cuts allow nonnegative flow, and all other cuts allow a zero flow. The number of cuts (and hence the number of halfspaces whose intersection defines the domain of g) may be exponential. We overcome this difficulty by considering a concave extension of the function g to a domain with a polynomial number of facets. Later in this subsection, we define the parametric min-discrepancy max-flow problem which is a generalization of the problem of max-flow with parametric bounds. We then proceed to show that the latter has a strongly polynomial time algorithm.

Definition 4.6 Let $G = (V, E)$ be a digraph, let $u_e \in R_+$ $(e \in E)$ be capacities on the edges, and let $\sigma_v \in R$ $(v \in V)$ be supplies or demands associated with the vertices. A vector $\boldsymbol{x} = (x_e)_{e \in E}$ such that $0 \leq x_e \leq u_e$ $(e \in E)$ is called *a pseudoflow*. The *excess* of the pseudoflow $\boldsymbol{x}$ at a vertex $v \in V$ is

$$\operatorname{excess}(v) = \sum_{i:(i,v)\in E} x_{iv} - \sum_{i:(v,i)\in E} x_{vi}.$$

The *discrepancy* of the pseudoflow $\boldsymbol{x}$ is

$$\Delta(\boldsymbol{x}) = \sum_{v \in V} |\sigma_v - \operatorname{excess}(v)| \ .$$

We seek a pseudoflow which minimizes the discrepancy, and refer to its discrepancy as the *minimum discrepancy* of the network.

Problem 4.7 [Minimum Discrepancy Pseudoflow]
Given a network $G = (V, E)$ with capacities $u_e \in R_+$ $(e \in E)$ on the edges, and supplies or demands $\sigma_v \in R$ $(v \in V)$ associated with the vertices, find a minimum discrepancy pseudoflow (see Definition 4.6) in G.

Remark 4.8 Given an instance of Problem 4.7, we construct an instance of the max-flow problem on a network $G' = (V', E')$ with two distinguished vertices $s', t' \in V'$ and capacities u'_e $(e \in E')$, where

$$V' = V \cup \{s', t'\} \ ,$$

$$E' = \{(v, t') \mid v \in V, \sigma_i < 0\} \cup \{(s', v) \mid v \in V, \sigma_i > 0\} \cup E \ ,$$

and

$$u'_e = \begin{cases} u_e & \text{if} \quad e \in E \\ \sigma_v & \text{if} \quad e = (s', v) \\ -\sigma_v & \text{if} \quad e = (v, t') \ . \end{cases}$$

It is easy to verify that the sum of twice the value of the max-flow in G' plus the minimum discrepancy in G equals $\sum_{i \in V} |\sigma_i|$. Furthermore, it is easy to compute a min-disc pseudoflow in G from a max-flow in G'.

Consider the parametric version of problem 4.7 where capacities, supplies, and demands are replaced by affine forms. The algorithm that computes the minimum discrepancy relative to the values induced by a given $\boldsymbol{\lambda}$ is piecewise affine. This follows from the fact that the construction of G' is piecewise affine and leads to a max-flow problem in a network with capacities that are affine forms of $\boldsymbol{\lambda}$. The final step is to apply the parametric max-flow algorithm which is piecewise affine as well. The domain of $\boldsymbol{\lambda}$'s for which the problem is well-defined has a polynomial number of facets. It follows from Theorem 4.1 that the parametric problem can be solved in strongly polynomial time for any fixed number of parameters. A consequence of this claim is that the problem of a feasible circulation with parametric bounds can be solved in strongly polynomial time for any fixed number of parameters.

We now consider a generalization of Problem 4.7 where we compute a maximum flow which minimizes the discrepancy.

Definition 4.9 Suppose a network G is as in Definition 4.6, and $s, t \in V$ are two distinguished vertices. The discrepancy of a pseudoflow in a network with distinguished vertices, is redefined to be

$$\Delta(\boldsymbol{x}) = - \sum_{v \in V \setminus \{s,t\}} |\sigma_v - \mathrm{excess}(v)| \ .$$

A *min-disc maximum (s,t)-flow* in G is a pseudoflow $\boldsymbol{x} = (x_e)_{e \in E}$ such that $\mathrm{excess}(s) + \mathrm{excess}(t) = 0$ and $\boldsymbol{x}$ maximizes the pair $(-\Delta(\boldsymbol{x}), f(\boldsymbol{x}))$ (where $f(\boldsymbol{x}) = \mathrm{excess}(s)$) relative to the lexicographic order. The *value* of the min-disc maximum (s,t)-flow is the pair $(-\Delta, f)$ which corresponds to the optimal pseudoflow.

Problem 4.10 [Min-Disc Max-Flow] Given a network G as in Definition 4.9, compute a min-disc maximum (s,t)-flow in G.

Remark 4.11 Problem 4.10 can be solved by two applications of a max-flow algorithm. First, compute a minimum discrepancy pseudoflow $\boldsymbol{x}$ in the network G with the edge (t,s) added to E (see Remark 4.8). Consider the pseudoflow $\boldsymbol{x}$ on the network G. It is easy to verify that (i) $\mathrm{excess}(s) + \mathrm{excess}(t) = 0$, and (ii) the pseudoflow $\boldsymbol{x}$ minimizes $\Delta(\boldsymbol{x})$. Construct the residual network G' with residual capacities relative to the pseudoflow $\boldsymbol{x}$. Compute a maximum (s,t)-flow x' in G'. The combined pseudoflow $x + x'$ is a min-disc maximum (s,t)-flow in G.

The claim made in Remark 4.8 carries over to Problem 4.10. When the capacities and demands are replaced by d-variable affine forms, the algorithm that computes the solution of Problem 4.10 for a fixed $\boldsymbol{\lambda} \in R^d$ is piecewise affine. See [22] for a related problem as follows. Given a network and k pairs of sources and sinks (σ_i, t_i) $(i = 1, \ldots k)$, compute a pseudoflow $\boldsymbol{x}$ such that (i) $\mathrm{excess}(v) = 0$ if $v \notin \{\sigma_1, \ldots, \sigma_k, t_1, \ldots, t_k\}$, (ii) $\mathrm{excess}(\sigma_i) + \mathrm{excess}(t_i) = 0$ for every i $(i = 1, \ldots, k)$, and (iii) $\boldsymbol{x}$ maximizes the k-tuple $(\mathrm{excess}(\sigma_1), \ldots, \mathrm{excess}(\sigma_k))$ relative to the lexicographic order.

Consider the parametric version of the problem, where the capacities and the supplies and demands at the vertices are replaced by affine forms:

Problem 4.12 [Parametric Min-Disc Max-Flow] Let $G = (V, E)$ be a digraph, let u_e for $e \in E$ and σ_v for $v \in V$ be d-variable affine forms corresponding to capacities on the edges and demands or supplies at the vertices, respectively. Find a $\boldsymbol{\lambda} \in R^d$ which maximizes the value of the min-disc maximum (s,t)-flow in the induced network.

Proposition 4.13 *The problem of parametric min-disc max-flow (Problem 4.12) can solved in strongly polynomial time for any fixed number of parameters.*

Proof: We limit the domain of the search to values of $\boldsymbol{\lambda} \in R^d$ for which the induced capacities are nonnegative. The polyhedron $\mathcal{Q}$ is the intersection of the m halfspaces $\{u_e \geq 0\}$ $(e \in E)$.
Denote by $g : \mathcal{Q} \to R^2$ the function which maps $\boldsymbol{\lambda} \in \mathcal{Q}$ to the solution of Problem 4.10 on the induced network. In order to apply Theorem 4.1, we need to show that the function g is concave. Observe that Remark 4.11 implies that the other conditions of Theorem 4.1 are satisfied. Suppose that $\{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2\} \subset \mathcal{Q}$. Let $x_e^{(i)} \in R_+$, $(e \in E,\ i = 1, 2)$ be the min-disc max (s,t)-flows in the induced network relative to $\boldsymbol{\lambda}_i$, with values $(-\Delta_i, f_i)$. Let $\sigma_v^{(i)}$ $(i = 1, 2,\ v \in V \setminus \{s,t\})$ be the demands induced by $\boldsymbol{\lambda}_i$, and let $u_e^{(i)}$ $(e \in E,\ i = 1, 2)$ be the capacities induced by $\boldsymbol{\lambda}_i$. Let $\alpha \in [0, 1]$ be any number. Let $\boldsymbol{\lambda}' = \alpha\boldsymbol{\lambda}_1 + (1-\alpha)\boldsymbol{\lambda}_2$. The capacities induced by $\boldsymbol{\lambda}'$ are $u'_e = \alpha u_e^{(1)} + (1-\alpha)u_e^{(2)}$, and the induced demands are $\sigma'_v = \alpha\sigma_v^{(1)} + (1-\alpha)\sigma_v^{(2)}$. We need to show that $g(\boldsymbol{\lambda}') \geq_{\rm lex} \alpha g(\boldsymbol{\lambda}_1) + (1-\alpha) g(\boldsymbol{\lambda}_2)$. Let x'_e $(e \in E)$ be the pseudoflow $x'_e = \alpha x_e^{(1)} + (1-\alpha)x_e^{(2)}$. Denote by $\text{excess}'(v)$, $\text{excess}_1(v)$, and $\text{excess}_2(v)$ the flow excesses at a vertex $v \in V$ relative to the pseudoflows $\boldsymbol{x}'$, $\boldsymbol{x}^{(1)}$, and $\boldsymbol{x}^{(2)}$ respectively. Let $g(\boldsymbol{\lambda}_1) = (-\Delta_1, f_1)$, $g(\boldsymbol{\lambda}_2) = (-\Delta_2, f_2)$, $\Delta' = \sum_{v \in V\setminus\{s,t\}} |\sigma'_v - \text{excess}'(v)|$ and $f' = \text{excess}'(s)$. It is easy to verify that $\boldsymbol{x}'$ is a feasible pseudoflow relative to the capacities u' and that $\text{excess}'(s) = \text{excess}'(t)$. Hence, $g(\boldsymbol{\lambda}') \geq_{\rm lex} (-\Delta', f')$. Note that $\text{excess}'(v) = \alpha\ \text{excess}_1(v) + (1-\alpha)\ \text{excess}_2(v)$, and $f' = \text{excess}'(s) = \alpha f_1 + (1-\alpha) f_2$. It follows that

$$
\begin{aligned}
\Delta' &= \sum_{v \in V\setminus\{s,t\}} |\sigma'_v - \text{excess}'(v)| \\
&= \sum_{v \in V\setminus\{s,t\}} |\alpha\sigma_v^{(1)} + (1-\alpha)\sigma_v^{(2)} - \alpha\text{excess}_1(v) - (1-\alpha)\text{excess}_2(v)| \\
&\leq \sum_{v \in V\setminus\{s,t\}} \left(\alpha|\sigma_v^{(1)} - \text{excess}_1(v)| + (1-\alpha)|\sigma_v^{(2)} - \text{excess}_2(v)|\right) \\
&= \alpha\Delta_1 + (1-\alpha)\Delta_2 \ .
\end{aligned}
$$

Now we can show that g is concave:

$$g(\boldsymbol{\lambda}') \geq_{\rm lex} (-\Delta', f') \geq_{\rm lex} (-\alpha\Delta_1 - (1-\alpha)\Delta_2, \alpha f_1 + (1-\alpha) f_2) = \alpha g(\boldsymbol{\lambda}_1) + (1-\alpha) g(\boldsymbol{\lambda}_2) \ .$$

∎

Corollary 4.14 *The following parametric problems can be solved in strongly polynomial time for any fixed number of parameters:*

(i). Problem 4.12 with a distiguished vertices with demand or supply, and the value of $\boldsymbol{\lambda} \in R^d$ is constrained so that the induced demands are nonnegative and the induced supplies are nonpositive.

(ii). Max-flow with parametric bounds.

(iii). Feasible circulation with parametric bounds.

Proof: Part (i) follows immediately from Remark 4.2. We need to replace the search domain $\mathcal{Q}$ by its intersection with the $O(n)$ halfspaces $\{\sigma_v \geq 0\}$ if v is a demand vertex, and $\sigma_v \leq 0$ if v is a supply vertex. It is easy to see that the intersection of these halfspaces equals the set of vectors relative to which the induced demands are nonnegative and the induced supplies are nonpositive.
To prove part (ii), consider an instance of the parametric bounds max-flow problem. Let $G = (V, E)$ be a network, let $s, t \in V$ be two distinguished vertices, and let ℓ_e, u_e $(e \in E)$ be d-variable affine forms. Assume, without loss of generality, that the edges incident on the source and the sink have zero lower bounds on the flow. We define a corresponding instance of the min-disc max (s,t)-flow problem on a network $G = (V, E)$, where u'_e and σ_v are capacities and demands. The capacities are $u'_e = u_e - \ell_e$ $(e \in E)$, and the demands are $\sigma_v = \sum_{\{u|(u,v)\in E\}} \ell_{uv} - \sum_{\{u|(v,u)\in E\}} \ell_{vu}$ $(v \in V \setminus \{s,t\})$.
The search domain is the intersection of the halfspaces that guarantee that $u'_e \geq \ell_e \geq 0$ $(e \in E)$. Consider Problem 4.12 on the network with capacities u'_e and demands σ_v. Suppose that $\boldsymbol{\lambda} \in \mathcal{Q}$ maximizes the min-disc maximum (s,t)-flow in the induced network, and $(-\Delta, f)$ is the optimal value. Consider the original parametric bounds max-flow problem. It is easy to see that there exists a vector $\boldsymbol{\lambda} \in \mathcal{Q}$ for which the induced network has a feasible flow if and only if $\Delta = 0$. Furthermore, if $\Delta = 0$ then $\boldsymbol{\lambda}$ is the solution of the original problem, and f is the value of the corresponding max flow.
Part (iii) is a special case of part (ii), where we consider only the first coordinate of the min-disc max-flow pair. There exists a $\boldsymbol{\lambda}$ which induces capacities allowing a feasible circulation if and only if $\Delta = 0$. ∎

Remark 4.15 Consider Problem 4.10 with parametric lower bounds on the flow. We show that if d is fixed d, then this problem can be solved in strongly polynomial time. The solution is based on the techniques we used to solve the parametric bounds max-flow problem. Define the "edge-discrepancy" $h(\boldsymbol{x})$ of a pseudoflow $\boldsymbol{x}$ as the sum, over all the edges for which the flow is smaller than the lower bound, of the difference between the lower bound and the flow. For an instance of Problem 4.12 with lower bounds, let $\mathcal{Q}$ be the intersection of all the halfspaces that guarantee that the bounds are nonnegative and that the upper bounds are not smaller than the respective lower bounds. Define the function $g : \mathcal{Q} \to R^3$ as follows. For $\boldsymbol{\lambda} \in \mathcal{Q}$, consider the induced network, and let $g(\boldsymbol{\lambda})$ be the maximum over all pseudoflows of $(-h(\boldsymbol{x}), -\Delta(\boldsymbol{x}), f(\boldsymbol{x}))$ relative to the lexicographic order. The domain of $g(\boldsymbol{\lambda})$ is $\mathcal{Q}$. Using arguments similar to the ones used for Problem 4.12, we can show that (i) the evaluation of g can be done by a piecewise affine algorithm (see Remark 4.11), and (ii) g is concave (see the proof of Proposition 4.13). Denote by $(-h, -\Delta, f)$ the maximum value of g. It is easy to verify that if $h = 0$, then the $\boldsymbol{\lambda}$ which maximizes g is the solution of the original instance of Problem 4.12 with lower bounds, and if $h \neq 0$, then the original instance has no solution.

4.3. Applications to Fixed Ratios and generalized flow

In this subsection we discuss the relation between the parametric extensions given in the previous subsection and the fixed ratios flow and generalized flow problems. We give strongly polynomial time reductions from these problems to Problem 4.12. It follows that the subclasses of the fixed ratios flow and generalized flow problems, which correspond to instances of Problem 4.12 with a fixed number of parameters, have strongly polynomial time algorithms.

Fixed Ratios Flow.

Problem 4.16 [Fixed Ratios Flow] Consider a network $G = (V, E)$ where ℓ, u, α, and S are as in Problem 1.2. Suppose $s, t \in V$ are two distinguished vertices. A *maximum fixed ratios flow* is an (s, t)-flow of maximum value, which satisfies the additional constraints of a fixed ratios circulation (see Problem 1.2).

Problem 4.17 [Flow with Fixed Forks] Consider a network $G = (V, E)$ where α and V are as in Problem 1.4. Suppose $s, t \in V$ are two distinguished vertices. A *maximum flow with fixed forks* is an (s, t)-flow of maximum value, which satisfies the additional constraints of a circulation with fixed forks (see Problem 1.4).

It is easy to see that Flow with Fixed Forks is a special case of the Fixed Ratios Flow problem.

Proposition 4.18 *There exists a linear-time reduction from instances of the fixed ratios flow problem to instances of the max-flow problem with parametric bounds such that (i) there is a trivial correspondence between the solutions of the two problems, and (ii) the number of parameters equals the number of equivalence classes relative to S.*

Proof: Given an instance of a fixed ratios flow problem, define the corresponding instance of the max-flow problem with parametric bounds as follows. We use the same network G and the same pair of distinguished vertices s, t. Let ℓ'_e, u'_e $(e \in E)$ be the affine forms which define the lower and upper bounds. If an edge e is not a member of any of the pairs in S, then we define $\ell'_e = \ell_e$, $u'_e = u_e$. Otherwise, we associate a parameter with each equivalence class of edges. Suppose that $\{e_{i_1}, \ldots, e_{i_k}\}$ is an equivalence class. Let λ be the parameter associated with this set. The flow values on these edges are related. Hence, we can find a vector $\boldsymbol{a} \in R^k_+$ as follows. For every feasible fixed ratios flow x_e $(e \in E)$, there exists a $\beta \in R_+$ such that $(x_{e_{i_1}}, \ldots, x_{e_{i_k}}) = \beta \boldsymbol{a}$. Define the parametric bounds $u'_{e_{i_j}} = \ell'_{e_{i_j}} = a_j \lambda$. Restrict the domain of the search $\mathcal{Q}$ to its intersection with the $O(m)$ halfspaces given by $\ell_{e_{i_j}} \leq a_j \lambda \leq u_{e_{i_j}}$. It follows from Remark 4.2 that Theorem 4.1 still holds when we intersect the domain with polynomially many halfspaces. To conclude the proof, note the following. A fixed ratio flow exists in G if and only if there exists a $\boldsymbol{\lambda} \in \mathcal{Q}$ relative to which the induced network has a feasible flow. Moreover, if $\boldsymbol{\lambda}$ is the solution of the parametric problem and $\boldsymbol{x}$ is the max-flow in the network induced by $\boldsymbol{\lambda}$, then $\boldsymbol{x}$ is a maximum fixed ratios flow in G. ∎

Corollary 4.19 *The problem of Fixed Ratios Flow can be solved in strongly polynomial time for any fixed number of equivalence classes relative to S.*

Corollary 4.20 *The problem of Flow with Fixed Forks can be solved in strongly polynomial time for any fixed number of forks.*

Generalized Flow.

The generalized flow problem [21, 16] is as follows. A network $G = (V, E)$ is given with lower and upper bounds ℓ_e, u_e ($e \in E$), respectively, on the flow, a distinguished vertex $s \in V$, and a vectors $\boldsymbol{\gamma} = (\gamma_e)_{e \in E}$ of *gain factors*. Find a pseudoflow $\boldsymbol{x}$ that maximizes

$$\sum_{i:(s,i) \in E} x_{si} - \sum_{i:(i,s) \in E} \gamma_{is} x_{is}$$

under the generalized flow conservation conditions

$$\sum_{i:(v,i) \in E} x_{vi} - \sum_{i:(i,v) \in E} \gamma_{iv} x_{iv} = 0 \quad (v \in V \setminus \{s\}) \; .$$

It is not known whether the generalized flow problem can be solved in strongly polynomial time. We show that if the number of edges e with $\gamma_e \neq 1$ is fixed, then the problem can be solved in strongly polynomial time. This result holds even when the vertices have supplies or demands and edges have lower bounds on the flow.

Proposition 4.21 *There exists a linear-time reduction from instances G of the generalized flow network problem with d edges with gains or losses to instances G' of the parametric min-disc max (s,t)-flow network problem with d parameters, such that if $\boldsymbol{\lambda}$ and $(-\Delta, f)$ constitute the solution of the parametric problem and $\boldsymbol{x}$ is the min-disc max (s,t)-pseudoflow relative to the capacities and demands induced by $\boldsymbol{\lambda}$, then: (i) $\Delta = 0$, (ii) f is the value of the max-flow in G, and (iii) given $\boldsymbol{\lambda}$ and $\boldsymbol{x}$, a max-flow in G can be constructed easily.*

Proof: Consider an instance of the generalized flow problem on a network $G = (V, E)$, with capacities $u_e \in R_+$ ($e \in E$), and two distinguished vertices $s, t \in V$. Let $e_i = (v_i, w_i) \in E$ ($i = 1, \ldots, d$) be the edges with gains or losses, and let γ_i be the gain factor associated with e_i. Define the corresponding instance of the parametric min-disc max-flow problem on a network $G' = (V, E')$, with s and t as the distinguished vertices, where the capacities u_e ($e \in E'$) are scalars, and the demands σ_v ($v \in V'$) are affine forms. Associate a parameter λ_i with the edge e_i ($i = 1, \ldots, d$). In the network G': (i) $E' = E \setminus \{e_1, \ldots, e_d\}$, (ii) the capacities are $u'_e = u_e$ ($e \in E'$), and (iii) the demands σ_v for $v \in V$ are $\sigma_v = \sum_{\{i|v_i=v\}} \lambda_{e_i} - \sum_{\{i|w_i=v\}} \gamma_i \lambda_{e_i}$. Replace $\mathcal{Q}$ by its intersection with the halfspaces $\lambda_i \leq u_{e_i}$. To prove correctness, consider any $\boldsymbol{\lambda} \in \mathcal{Q}$. Let $(-\Delta, f)$ be the value of the min-disc max-flow, and let x_e ($e \in E'$) be a feasible min-disc pseudoflow (i.e., $\Delta(\boldsymbol{x}) = \Delta$), relative to the capacities and demands induced by $\boldsymbol{\lambda}$. Let $\boldsymbol{x}'$ be a pseudoflow in G, where $x'_e = x_e$ ($e \in E'$), and $x'_{e_i} = \lambda_i$ ($i = 1, \ldots, d$). It is easy to see that $\Delta = 0$ if and only if $\boldsymbol{x}'$ is a generalized flow in G. ∎

Corollary 4.22 *The generalized flow problem, with a fixed number of edges with gain factors other than 1, can be solved in strongly polynomial time algorithm.*

Remark 4.23 Tardos [27] gave a strongly polynomial time algorithm for the min-cost circulation problem. Consider the parametric extension of the problem where the bounds on the flow on some edges are parameterized. Tardos' algorithm is piecewise affine in these bounds. Hence, the parametric extension of the min-cost flow problem, where the number of parameters is fixed, can be solved in strongly polynomial time. Interesting applications are strongly polynomial time algorithms for the min-cost generalizations of (i) the fixed ratios flow problem, where the number of equivalence classes of edges is fixed, and (ii) the generalized flow problem, where only a constant number of edges have gain factors other than 1.

References.

[1] G. M. Adel'son-Velskii, E. A. Dinic, and A. V. Karzanov. *Flow algorithms.* Science, Moscow, 1975. In russian.

[2] C. Berge and A. Ghouila-Houri. *Programming, games and transportation networks.* John Wiley & Sons, New York, 1965.

[3] P. J. Carstensen. *The complexity of some problems in parametric, linear, and combinatorial programming.* PhD thesis, Department of Mathematics, University of Michigan, Ann Arbor, Mich., 1983.

[4] E. Cohen. *Combinatorial Algorithms for Optimization Problems.* PhD thesis, Department of Computer Science, Stanford University, Stanford, Ca., 1991.

[5] E. Cohen and N. Megiddo. Strongly polynomial time and NC algorithms for detecting cycles in periodic graphs. *J. Assoc. Comput. Mach.* To appear.

[6] E. Cohen and N. Megiddo. Strongly polynomial and NC algorithms for detecting cycles in dynamic graphs. In *Proc. 21st Annual ACM Symposium on Theory of Computing*, pages 523–534. ACM, 1989.

[7] E. Cohen and N. Megiddo. Maximizing concave functions in fixed dimension. Technical Report RJ 7656 (71103), IBM Almaden Research Center, San Jose, CA 95120-6099, August 1990.

[8] E. A. Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Math. Dokl.*, 11:1277–1280, 1970.

[9] D. P. Dobkin, R. J. Lipton, and S. P. Reiss. Linear programming is Log-Space hard for P. *Information Processing Let.*, 8(2):96–97, 1978.

[10] D. P. Dobkin and S. P. Reiss. The complexity of linear programming. *Theoretical Computer Science*, 11:1–18, 1980.

[11] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. Assoc. Comput. Mach.*, 19:248–264, 1972.

[12] L. R. Ford Jr. and D. R. Fulkerson. *Flows in networks.* Princeton Univ. Press, Princeton, NJ, 1962.

[13] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.*, 18:30–55, 1989.

[14] A. Ghouila-Houri. Recherche du flot maximum dans certains réseaux lorsqu'on impose une condition de bouclage. In *Proc. of the 2nd Int. Conf. on Oper. Res., London*, page 156. American Mathematical Society, 1960.

[15] A. Ghouila-Houri. Une généralisation de l'algorithme de Ford-Fulkerson. *C. R. Acad. Sci., Paris*, 250:457, 1960.

[16] A. V. Goldberg, É. Tardos, and R. E. Tarjan. Network flow algorithms. Technical Report STAN-CS-89-1252, Stanford University, 1989.

[17] P. Gordan. Über die auflösung linearer gleichungen mit reelen coefficienten. *Mathematische Annalen*, 6:23–28, 1873.

[18] D. Gusfield. Parametric combinatorial computing and a problem of program module distribution. *J. Assoc. Comput. Mach.*, 30:551–563, 1983.

[19] A. J. Hoffman. A generalization of max-flow min-cut. *Math. Prog.*, 6:352–359, 1974.

[20] A. Itai. Two-commodity flow. *J. Assoc. Comput. Mach.*, 25(4):596–611, 1978.

[21] E. L. Lawler. *Combinatorial optimization: networks and matroids.* Holt, Reinhart, and Winston, New York, 1976.

[22] N. Megiddo. A good algorithm for lexicographically optimal flows in multi-terminal networks. *Bulletin of the AMS*, 83:407–409, 1977.

[23] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. Assoc. Comput. Mach.*, 30:337–341, 1983.

[24] N. Megiddo. Towards a genuinely polynomial algorithm for linear programming. *SIAM J. Comput.*, 12:347–353, 1983.

[25] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. Assoc. Comput. Mach.*, 31:114–127, 1984.

[26] C. H. Norton, S. A. Plotkin, and É. Tardos. Using separation algorithms in fixed dimension. In *Proc. 1st ACM-SIAM Symposium on Discrete Algorithms*, pages 377–387. ACM-SIAM, 1990.

[27] É. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, 1985.

[28] É. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Oper. Res.*, 34:250–256, 1986.