A new algorithm for the intersection of a line with the independent set polytope of a matroid

Alexandre Skoda

A new algorithm for the intersection of a line with the independent set polytope of a matroid.

Alexandre Skoda
Equipe Combinatoire et Optimisation. CNRS et Université Paris 6
France Telecom R&D Sophia Antipolis [1]

Abstract

We present a new algorithm for the problem of determining the intersection of a half-line $\Delta_u = \{x \mid x = \lambda u, \lambda \geq 0, u \in \mathbb{R}^n_+\}$ with the independent set polytope of a matroid. We show it can also be used to compute the strength of a graph and the corresponding partition using successive contractions. The algorithm is based on the maximization of successive linear forms on the boundary of the polytope. We prove it is a polynomial algorithm in probability with average number of iterations in $O(n^5)$. Finally, numerical tests reveal that it should only require $O(n^2)$ iterations in practice.

Keywords: Algorithm, graph, strength of a graph, submodular function, matroid.

1 Introduction

Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid defined on a finite set $E = \{e_1, e_2, \ldots, e_n\}$, with the collection of independent sets $\mathcal{I}$. Let r be the rank function of $\mathcal{M}$. J. Edmonds [6] showed that the independent set polytope $P(\mathcal{M})$ of $\mathcal{M}$ is fully determined by a family of linear inequalities:

$$P(\mathcal{M}) = \{x \in \mathbb{R}^n_+ \mid x(S) \leq r(S) \text{ for all } S \subseteq E\}.$$

where $x(S) = \sum_{i \in S} x_i$.

When $\mathcal{M}$ is a graphic matroid, E becomes the family of edges of a graph G and $\mathcal{I}$ is the set of forests of this graph. In this case, we denote $P(G)$ the independent set polytope of $\mathcal{M}$.

For a given $u \in \mathbb{R}^n_+$, the intersection of the half-line $\Delta_u = \{x \mid x = \lambda u, \lambda \geq 0\}$ with the polytope $P(\mathcal{M})$ is a closed interval $[0, \lambda_{max}]u$. The upper bound λ_{max} is the solution of the following linear program :

$$(PL) : \max\{\lambda \mid x \in P(\mathcal{M});\ x = \lambda u\}$$

[1] This research has been supported by a grant from "France Telecom R&D".

We give a new algorithm solving (PL). Many important problems in combinatorial optimization can be reduced to the program (PL). For instance, the computation of the strength of a graph and the minimization of a submodular function. Several efficient algorithms have already been found to solve (PL) ([4, 5], [8]). Most of them use an auxiliary digraph. Besides the augmentation at each iteration can be relatively small but sufficient to guarantee their polynomiality. The new algorithm we propose here is based on the maximization of appropriate linear forms on the polytope $P(\mathcal{M})$. The augmentation at each iteration is chosen as large as possible. In this way, this new algorithm could be faster than the previous ones. Nevertheless the analysis of the complexity of this algorithm seems to be difficult and at first we prove the algorithm is polynomial in probability. We now describe this new algorithm. At each step of the algorithm, a point $\lambda u \in \Delta_u$ is given as a convex combination of n linearly independent vertices $(X_1, X_2, \ldots, X_n)$ of $P(\mathcal{M})$. We start with the n elements of E, *i.e.* $X_j = e_j$ for all $j \in [1, n]$. At each step, we modify at least one of the X_j by "pushing" λu along the line Δ_u to the boundary of $P(\mathcal{M})$. Of course, the algorithm stops when λu meets the boundary. In a preceding paper [7], we chose to progress "carefully" at each step. We had to consider all vertices we can obtain from the X_j's by addition or exchange of an element of E according to the structure of the matroid. In order to decide which vertex we had to modify, we needed to construct an auxiliary digraph and to solve problems of arborescences of shortest paths in this graph. We proved :

Theorem 1.1 *This algorithm, which works with additions and exchanges of elements of E to solve the linear program (PL), ends after at most n^5 iterations. Its running time is $O(n^8 + \gamma n^7)$ where γ is the time for an oracle call.*

In this paper, we make the opposite choice to progress as much as possible at each step. We look for a new vertex X_{n+1} which is as far as possible above the affine hyperplane generated by the X_j's. It is equivalent to maximize on the boundary of the polytope $P(\mathcal{M})$ the linear form defining the equation of the hyperplane. We replace one of the X_j's by the new vertex X_{n+1} and we iterate. It seems more difficult to control the complexity of this new algorithm because, even if at each step we progress more quickly, nevertheless we don't necessarily progress in the right direction of the line Δ_u but in the direction which is orthogonal to the hyperplane. Of course, these two directions are in average close one from the other and we'll prove here:

Theorem 1.2 *This new algorithm which maximizes appropriate linear forms on the boundary of the polytope to solve the linear program (PL), is strongly polynomial in probability. It ends after an average number of iterations less or equal to $4n^5$. Its average running time is less or equal to $O(n^7 + \gamma n^6)$.*

These two algorithms are useful to compute the strength of a graph or of a matroid. Let $G = (V, E)$ be a connected graph. We suppose, for each edge e of E, a strength $u(e) > 0$ is given, measuring the cost we have to pay to delete the edge e from the graph. For $S \subseteq E$, we set $u(S) = \sum_{e \in S} u(e)$ and we denote $k(S)$ the number of new connected components which arise when we cancel the set S of edges. We define the strength $\sigma(G, u)$ of the graph G by :

$$\sigma(G, u) := \min_S \left(\frac{u(S)}{k(S)};\ S \subseteq E,\ k(S) > 0 \right).$$

The strength of a graph has been proposed by D. Gusfield [10] as a measure of network invulnerability. W.H. Cunningham [5] has proposed an algorithm for computing the strength and finding a minimizing set S. D. Gusfield [11], H. N. Gabow [9], Cheng and Cunningham [3] and F. Barahona [1, 2] have proposed other algorithms improving the complexity of W.H. Cunningham's strongly polynomial algorithm.
The rank function is submodular, therefore there exists a unique maximal set $S \subseteq E$ such that $x_{max} = \lambda_{max} u$ verifies the constraint $x(S) = r(S)$ of the polytope $P(G)$. We prove, in section 2, the following results :
1) If $S = E$, $\sigma(G, u) = \frac{u(E)}{k(E)}$.
2) If $S \subset E$, we build the contracted graph G' by replacing the vertices of G incident to at least one edge of S by a single vertex and deleting S. Let u' be the restriction of the vector u to $E \setminus S$. Then we prove :

$$\sigma(G, u) = \sigma(G', u').$$

As each contraction removes at least one edge from the graph G, we need at most n iterations of the algorithm solving (PL) to compute the strength of G. We more generally define in section 2, the strength of a matroid and we state for a matroid the results we have claimed for graphs.

In [5], W.H. Cunningham computes the strength of a graph by another method without any use of (PL). Nevertheless, the schemes of the two methods are similar. W.H. Cunningham solves at most n linear programs. For each of them he has to find, for any given x not in the polytope $P(G)$, the most violated inequality, that is to find $S \subseteq E$ maximizing $x(S) - r(S)$. Our linear program (PL) is equivalent to minimize $\frac{r(S)}{x(S)}$ and therefore it is quite different. But W.H. Cunningham has proved in [5], one can solve (PL) by at most n iterations of the "most violated inequality" problem. We give here a direct solution of (PL) and we hope the interest of this solution is also justified by the new methods we use.

At each step of the algorithm, we consider the affine hyperplane H generated by the n vertices $X_1, X_2, \ldots, X_n$ of $P(G)$ (or $P(\mathcal{M})$) and the equation h of H ($h(X_j) = 1$, $1 \leq j \leq n$). As in W.H. Cunningham's article [4],

we build a directed graph $\tilde{G}$, which classifies all possible additions and exchanges of edges. We don't use this digraph to choose a new vertex X_{n+1} by addition or exchange of edges as in [7] but to prove that the vertex X_{n+1} which maximizes the linear form h on $P(\mathcal{M})$ is far enough above the hyperplane H generated by $X_1, X_2, \ldots, X_n$.

The theorem 4.1 in section 4.1 is the decisive result : we prove we progress at least of the quantity $\frac{1}{n^2}$ to the boundary of $P(G)$ at each iteration in the direction which is orthogonal to the hyperplane H. A convenient probabilistic model using Bernoulli's scheme allows us to prove in section 4.2 that the algorithm stops in average after at most $4n^5$ iterations. At the end of the paper, section 4.4 gives a review of numerical tests. According to the results of these tests, one can hope the algorithm ends after $O(n^2)$ iterations in the deterministic case. The results of sections 2 and 3 can be extended to polymatroids. In section 4, the proof of theorem 4.1 doesn't work for polymatroids because the augmentation associated with an addition or an exchange can be very small (for a matroid it is equal to 1) and so our probabilistic approach is restricted to matroids.

2 Strength of a matroid

Similar results arise in papers of S. Fujishige [8] and J. Fonlupt and A. Skoda [7] but for the sake of completeness we give direct proofs of the results we need in the general case of a matroid (or even a polymatroid). Let $r : P(E) \to \mathbb{Z}^+$ (resp. $r : P(E) \to \mathbb{R}^+$) be a submodular function, that is, for all A and $B \subseteq E$, we have : $r(A \cup B) + r(A \cap B) \leq r(A) + r(B)$. We suppose $r(\emptyset) = 0$ and r is nondecreasing ($A \subseteq B \Rightarrow r(A) \leq r(B)$). Then r is a rank function (resp. r is a polymatroid function). The polytope associated with r is defined by :

$$P_r = \{x \in \mathbb{R}_+^E; \forall A \subseteq E, x(A) \leq r(A)\}.$$

When r is a polymatroid function the polytope P_r will be called a polymatroid.

We consider a matroid $\mathcal{M}$ with basis E and rank function r (or more generally in this section a polymatroid P_r with underlying set E and polymatroid function r). Let $u \in \mathbb{R}^n$ be a positive vector. We define the *strength* σ *of the matroid* $\mathcal{M}$ by:

$$\sigma(\mathcal{M}, u) := \min_{\emptyset \neq B \subseteq E} \frac{u(B)}{r(E) - r(\overline{B})}. \tag{1}$$

where $\overline{B} = E \setminus B$. For a given $\lambda \in \mathbb{R}_+$, let H_λ be the hypercube defined by:

$$H_\lambda := \{x \mid x \in \mathbb{R}^n, 0 \leq x(e) \leq \lambda u(e) \text{ for all } e \in E\}. \tag{2}$$

We consider the linear program (P_λ), parametrized by λ:

$$z(\lambda) := \max \left\{ \sum_{e \in E} x(e) \mid x \in P(\mathcal{M}) \cap H_\lambda \right\}. \tag{3}$$

(respectively $x \in P_r \cap H_\lambda$). $z(\lambda)$ is a nondecreasing function of λ. We have $z(0) = 0$ and there exists $\lambda^{'} > 0$ such that $\forall \lambda \geq \lambda^{'}$, $z(\lambda) = r(E)$. The following result is a direct consequence of the theorem of intersection of two polymatroids.

Proposition 2.1 *For each fixed λ, we have the equality :*

$$z(\lambda) = \min_{A \subseteq E} [\lambda u(A) + r(\overline{A})]. \tag{4}$$

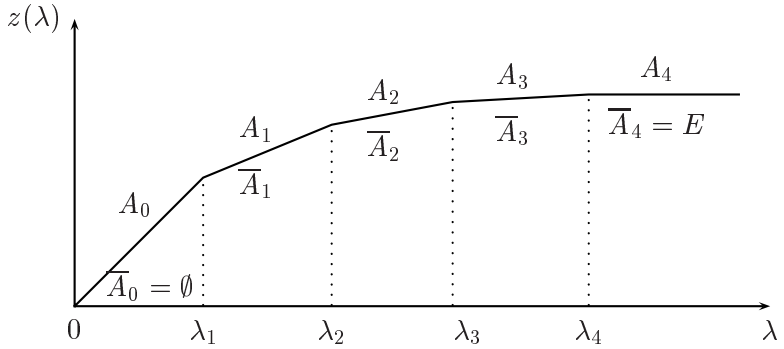
In [12] we give a direct proof for a matroid (which is also valid for a polymatroid because the proof only uses the submodularity of r). A similar result appears in W.H. Cunningham's article [5].

As consequence of linear parametrized programming, we can prove the function $z(\lambda)$ is nondecreasing, concave and piecewise linear. We prove more precisely:

Proposition 2.2 *Let $\lambda_0 = 0 < \lambda_1 < \lambda_2 < \ldots < \lambda_k$ be the values of λ for which the function $\lambda \to z(\lambda)$ is not differentiable (that is the breakpoints of the curve $z(\lambda)$).*

- (i) *For every open interval $]\lambda_i, \lambda_{i+1}[$ there exists a unique subset A_i of E such that $z(\lambda) = \lambda u(A_i) + r(\overline{A}_i)$.*
- (ii) *If $\lambda = \lambda_i$, there exist two distinct subsets A_{i-1} and A_i such that $z(\lambda) = \lambda u(A_{i-1}) + r(\overline{A}_{i-1})$ and $z(\lambda) = \lambda u(A_i) + r(\overline{A}_i)$.*
- (iii) *$\emptyset = \overline{A}_0 \subset \overline{A}_1 \subset \overline{A}_2 \subset \ldots \subset \overline{A}_k = E$ and $|\overline{A}_i| \geq |\overline{A}_{i-1}| + 1$.*

As $|A_i| \leq n$, (iii) implies $k \leq n$. Therefore the curve $z(\lambda)$ has at most n breakpoints.



Proof: Let $\overline{\lambda}$ be a value of λ such that at least two distinct sets B_1 and B_2 achieve the minimum $z(\overline{\lambda})$. Therefore we have $z(\overline{\lambda}) = \overline{\lambda}u(B_1) + r(\overline{B}_1) = \overline{\lambda}u(B_2) + r(\overline{B}_2)$. Then, by submodularity of the rank function r:

$$\begin{aligned}\overline{\lambda}u(B_1 \cap B_2) + r(\overline{B}_1 \cup \overline{B}_2) + \overline{\lambda}u(B_1 \cup B_2) + r(\overline{B}_1 \cap \overline{B}_2) \\ \leq \overline{\lambda}u(B_1) + r(\overline{B}_1) + \overline{\lambda}u(B_2) + r(\overline{B}_2) = 2z(\overline{\lambda}).\end{aligned} \tag{5}$$

On the other hand, we have by definition of $z(\overline{\lambda})$:

$$z(\overline{\lambda}) \leq \overline{\lambda}u(B_1 \cap B_2) + r(\overline{B}_1 \cup \overline{B}_2) \quad \text{and} \quad z(\overline{\lambda}) \leq \overline{\lambda}u(B_1 \cup B_2) + r(\overline{B}_1 \cap \overline{B}_2). \tag{6}$$

We deduce from (5) and (6) :

$$z(\overline{\lambda}) = \overline{\lambda}u(B_1 \cap B_2) + r(\overline{B}_1 \cup \overline{B}_2) = \overline{\lambda}u(B_1 \cup B_2) + r(\overline{B}_1 \cap \overline{B}_2).$$

In other words, $B_1 \cap B_2$ and $B_1 \cup B_2$ are also minimizers of $z(\overline{\lambda})$. Thus we can define the smallest set $B_{\min}$ and the largest set $B_{\max}$ such that :

$$z(\overline{\lambda}) = \overline{\lambda}u(B_{\min}) + r(\overline{B}_{\min}) = \overline{\lambda}u(B_{\max}) + r(\overline{B}_{\max}).$$

In addition, for all subset B achieving the minimum $z(\overline{\lambda})$ and distinct from $B_{\min}$ and $B_{\max}$, we have :

$$u(B_{\min}) < u(B) < u(B_{\max}). \tag{7}$$

As $\lambda u(B) + r(\overline{B}) = \overline{\lambda}u(B) + r(\overline{B}) + (\lambda - \overline{\lambda})u(B)$, we have :

$$\lambda u(B) + r(\overline{B}) = z(\overline{\lambda}) + (\lambda - \overline{\lambda})u(B). \tag{8}$$

We obtain, from (7) and (8), for $\overline{\lambda} < \lambda$:

$$z(\overline{\lambda}) + (\lambda - \overline{\lambda})u(B_{\min}) < \lambda u(B) + r(\overline{B}) < z(\overline{\lambda}) + (\lambda - \overline{\lambda})u(B_{\max}).$$

As $B_{\min}$ and $B_{\max}$ minimize $z(\overline{\lambda})$, this is equivalent to:

$$\lambda u(B_{\min}) + r(\overline{B}_{\min}) < \lambda u(B) + r(\overline{B}) < \lambda u(B_{\max}) + r(\overline{B}_{\max}). \tag{9}$$

For $\lambda < \overline{\lambda}$ (*i.e.* $\lambda - \overline{\lambda} < 0$), we obtain, in a similar way, from (7) and (8) :

$$\lambda u(B_{\max}) + r(\overline{B}_{\max}) < \lambda u(B) + r(\overline{B}) < \lambda u(B_{\min}) + r(\overline{B}_{\min}). \tag{10}$$

The inequalities (9) and (10) show that $\overline{\lambda}$ is one of the values $\lambda_0, \lambda_1, \ldots, \lambda_k$ (it corresponds to an angular point in the graph of the function $z(\lambda)$). For example, let us suppose $\overline{\lambda} = \lambda_i$. Then we can choose $A_i = B_{\min}$ and $A_{i-1} = B_{\max}$. Thus $A_i \subset A_{i-1}$ and therefore $\overline{A}_{i-1} \subset \overline{A}_i$ and $|\overline{A}_i| \geq |\overline{A}_{i-1}| + 1$.

Proposition 2.3 *The last value of discontinuity λ_k determines the strength of the matroid $\mathcal{M}$:*

$$\sigma(\mathcal{M}, u) = \frac{1}{\lambda_k} = \frac{u(A_{k-1})}{r(E) - r(\overline{A}_{k-1})}. \tag{11}$$

The proof of proposition 2.3 immediately follows from proposition 2.2. More details can be found in [12].

Theorem 2.1 *The contracted matroid $\mathcal{M}/\overline{A}_1$ has the same strength as the matroid $\mathcal{M}$:*

$$\sigma(\mathcal{M}, u) = \sigma(\mathcal{M}/\overline{A}_1, u) = \frac{1}{\lambda_k}. \tag{12}$$

Proof: $z(\lambda)$ is solution of the linear program (P_λ) associated with $\mathcal{M}$ (resp. the polymatroid P_r) defined by:

$$(P_\lambda) \begin{cases} 0 \leq x(e) \leq \lambda u(e) & \forall e \in E, \\ x(A) \leq r(A) & \forall A \subseteq E, \\ \sum_{e \in E} x(e) = z(\lambda)(\max). \end{cases} \tag{13}$$

For $0 \leq \lambda \leq \lambda_1$, $H_\lambda \subset P(\mathcal{M})$ and the intersection of $P(\mathcal{M})$ with the half-line Δ_u generated by u is a line-segment $[0, \lambda_1 u]$. Thus λ_1 corresponds to the first rank constraint met by Δ_u. In other words, λ_1 is the solution of (PL) for the polytope $P(\mathcal{M})$. More precisely we have $x(\overline{A}_1) = \lambda_1 u(\overline{A}_1) = r(\overline{A}_1)$. Then, for all $\lambda \geq \lambda_1$, a solution x achieving the maximum of $z(\lambda)$ is such that $x(e) = \lambda_1 u(e)$ for $e \in \overline{A}_1$. Thus:

$$\sum_{e \in E} x(e) = \sum_{e \in A_1} x(e) + \sum_{e \in \overline{A}_1} x(e) = \sum_{e \in A_1} x(e) + \lambda_1 u(\overline{A}_1) = \sum_{e \in A_1} x(e) + r(\overline{A}_1). \tag{14}$$

On the other hand, from (13), we have $\forall A \subseteq A_1$, $x(A \cup \overline{A}_1) = x(A) + x(\overline{A}_1) \leq r(A \cup \overline{A}_1)$. Then we have :

$$x(A) \leq r(A \cup \overline{A}_1) - x(\overline{A}_1) = r(A \cup \overline{A}_1) - r(\overline{A}_1). \tag{15}$$

We consider the contracted matroid $\mathcal{M}/\overline{A}_1$ with set $E_1 = A_1$ and rank function $r_1(A) := r(A \cup \overline{A}_1) - r(\overline{A}_1)$ for $A \subseteq A_1$ (resp. the polymatroid P_{r_1} associated with r_1 and the set A_1). The linear program $(P_1(\lambda))$ associated with $\mathcal{M}/\overline{A}_1$ (resp. P_{r_1}) is defined by :

$$(P_1(\lambda)) \begin{cases} 0 \leq x(e) \leq \lambda u(e) & \forall e \in A_1, \\ x(A) \leq r_1(A) & \forall A \subseteq A_1, \\ \sum_{e \in A_1} x(e) = z_1(\lambda)(\max). \end{cases} \tag{16}$$

From (14) and (15), it follows (P_λ) is equivalent for $\lambda \geq \lambda_1$ to :

$$\begin{cases} x(e) = \lambda_1 u(e) & \forall e \in \overline{A}_1 \\ 0 \leq x(e) \leq \lambda u(e) & \forall e \in A_1 \\ x(A) \leq r_1(A) & \forall A \subseteq A_1 \\ z_1(\lambda)(\max) + r(\overline{A}_1) = z(\lambda)(\max) \end{cases}$$

Therefore $z(\lambda) = z_1(\lambda) + r(\overline{A}_1)$ for $\lambda \geq \lambda_1$. The breakpoints $\lambda_2, \ldots, \lambda_k$ are the same for the curves $z(\lambda)$ and $z_1(\lambda)$. Thus $\frac{1}{\lambda_k} = \sigma(\mathcal{M}/\overline{A}_1, u) = \sigma(\mathcal{M}, u)$.

In particular, λ_2 can be computed by solving (PL) for the polytope $P(\mathcal{M}/\overline{A}_1)$. In the same way, λ_{i+1} is solution of (PL) for the polytope $P(\mathcal{M}/(\overline{A}_1 \cup \cdots \cup \overline{A}_i))$. Therefore we have :

Theorem 2.2 *We can compute the solution $z(\lambda)$ of the linear program P_λ by solving the program (PL) at most n times, first for $P(\mathcal{M})$ and after for at most $n-1$ other polytopes associated with matroids, the underlying sets of which are strictly decreasing.*

3 Algorithm using maximization of linear forms

Let $r : P(E) \to \mathbb{R}$ be a polymatroid function. We recall the polymatroid P_r, associated with r, is the polytope defined by :

$$P_r = \{x \in \mathbb{R}_+^E; \forall A \subseteq E, x(A) \leq r(A)\}.$$

A vertex of P_r is an extreme point of P_r. For $u \in \mathbb{R}_+^n$, we recall the point $x_{\max}$, defined in the introduction, corresponds to the solution of the linear program :

$$(PL) \qquad \max \{\lambda \mid x \in \mathbb{R}_+^E;\ x \in P_r;\ x = \lambda u\}.$$

We denote $X_1, X_2, \ldots, X_l$, the vertices of P_r. We associate with each vector X_j a variable y_j for $1 \leq j \leq l$. P_r is the convex hull of the vectors X_j, $1 \leq j \leq l$. We can now write the linear program (PL) as follows :

$$(PL) = \begin{cases} \lambda_{max} = max\ \lambda \\ \forall j,\ 1 \leq j \leq l,\ \ y_j \geq 0 \\ \sum_{j=1}^{l} y_j X_j = \lambda u \\ \sum_{j=1}^{l} y_j = 1 \end{cases}$$

A subset of columns J is called a feasible basis if $|J| = n$, if the column vectors $X_j, j \in J$ are linearly independent and if there exists $\lambda > 0$ such that the linear system :

$$\begin{cases} \forall j \in J,\ y_j \geq 0, \\ \sum_{j \in J} y_j X_j = \lambda u \\ \sum_{j \in J} y_j = 1 \end{cases}$$

admits a unique solution. We call λ the value of the basis solution of the system.

Let X be a vertex of the polymatroid. We say the subset $A \subseteq E$ is X-tight if $X(A) = r(A)$. We have the following well known result :

Lemma 3.1 *If A and $B \subseteq E$ are X-tight, then $A \cap B$ and $A \cup B$ are X-tight.*

Proof: By submodularity of r, we have:

$$X(A \cup B) + X(A \cap B) \leq r(A \cup B) + r(A \cap B) \leq r(A) + r(B) = X(A) + X(B).$$

So equality must hold thoughout and $X(A \cup B) = r(A \cup B)$ and $X(A \cap B) = r(A \cap B)$.

3.1 Definition of an auxiliary digraph $G(J)$ and preliminary results

When we write $\lambda u = \sum_{j=1}^{n} y_j X_j$, we'll only consider the coefficients $y_j > 0$. We call support of the solution y the set $\operatorname{Supp} J = \{j \in J \mid y_j > 0\}$.
We'll associate with every feasible solution y with support J a directed graph denoted $G(J) = G(\tilde{E}, A)$. The set $\tilde{E}$ of vertices of the graph $G(J)$ is, on one hand, the set $E = \{e_1, e_2, \ldots, e_n\}$ of the polymatroid and on the other hand, an auxiliary element e_0 such that $\tilde{E} = E \cup \{e_0\}$. Let us now describe the set of oriented arcs of $G(J)$ (denoted A or $A(J)$). If $a \in E$, (e_0, a) is an arc if there exists $j \in \operatorname{Supp} J$ such that $X_j + a$ is independent. Then we say a is a source (associated with X_j). We denote S_0 the set of sources. If a and $b \in E$, (a, b) is an arc if there exists $j \in \operatorname{Supp} J$ such that $a \in X_j$, $b \notin X_j$, $X_j \cup \{b\}$ is dependent and a belongs to the unique circuit of $X_j \cup \{b\}$ which contains b. These arcs may be multiple. We denote $(e_0, a)_{X_j}$ or $(a, b)_{X_j}$ when we have to precise the vertex X_j associated with this arc.
If the vertex e_0 is not a root of the graph $G(J)$, we'll prove the current solution y is optimal and λ_{max} is the value of the solution y. More precisely, we have:

Proposition 3.1 *If e_0 is not a root of $G(J)$, the set :*

$$S := \{v \in E;\ \textit{no path of } G(J) \textit{ connects } e_0 \textit{ to } v\}$$

is X_j-tight for all $j \in Supp(J)$. That is, $\forall j \in Supp(J)$, $X_j(S) = f(S)$.

Proof: As e_0 is not a root of $G(J)$, S is non empty. If $v \in S$, then $\forall j \in Supp(J)$, $X_j + v$ is a dependent set, otherwise v would be a source. Then $(e_0, v)_{X_j}$ would be an arc of $G(J)$ in contradiction with the definition of S. Let us prove S is X_j-tight. For $v \in S$, let us consider the unique circuit C_v of $X_j + v$ which contains v. We claim C_v is a subset of S. Indeed, if $u \notin S$, e_0 is connected to u (by definition of S), but if moreover $u \in C_v$, then $(u, v)_{X_j}$ is an arc of $G(J)$ and therefore e_0 is connected to v, in contradiction with the assumption $v \in S$. C_v is X_j-tight since $X_j(C_v) = |C_v| - 1 = r(C_v)$. As C_v is a subset of S for $v \in S$, S is the union of the X_j-tight sets C_v. Therefore S is X_j-tight.

Proposition 3.2 *If the vertex e_0 is not a root of the graph $G(J)$, the current solution y is optimal and λ_{max} is the value of the solution y.*

Proof: Let us consider the set S defined in proposition 3.1. Therefore we have $\forall j \in Supp(J)$, $X_j(S) = f(S)$. As $\lambda u = \sum_{j=1}^{n} y_j X_j$ with $\sum_{j=1}^{n} y_j = 1$, we finally have for $x = \lambda u$:

$$x(S) = \lambda u(S) = \sum_{j=1}^{n} y_j X_j(S) = \left(\sum_{j=1}^{n} y_j \right) f(S) = f(S)$$

($y_j = 0$ if $j \notin Supp(J)$) and therefore the current solution is optimal.

Let us now suppose the vertex e_0 is a root of $G(J)$. We'll prove the following result:

Proposition 3.3 *If e_0 is a root of $G(J)$, we can find for all $e_j \in E$, a finite set A_j and two finite families of vertices of $P(\mathcal{M})$, $(X_\gamma)_{\gamma \in A_j}$ and $(X_\gamma^{'})_{\gamma \in A_j}$, such that e_j can be written:*

$$e_j = \sum_{\gamma \in A_j} (X_\gamma^{'} - X_\gamma)$$

and such that $X_\gamma \in \{X_1, X_2, \ldots, X_n\}$.

Proof: We consider an arborescence of shortest paths in $G(J)$ connecting e_0 to any element $e \in E$. More precisely, for each element $e \in E$, we consider the shortest paths connecting e_0 to e. We denote C the set of all these shortest paths when e varies in E. For each element of E, denoted e_j, we choose such a shortest path connecting e_0 to e_j. We denote C_0 this choice of shortest paths connecting e_0 to elements $e \in E$ which are the vertices of $G(J)$. For each fixed element e_j, we now consider the path C_j in C_0 which starts from a source e_1 and ends to e_j (for a fixed j we choose a specific numbering of the vertices of C_j independent of a given ordering of E). As the source e_1 is defined using an addition, there exist vertices X_1 and $X_1^{'}$ such that $X_1^{'} = X_1 + \{e_1\}$. Each arc $(e_{i-1}, e_i)_{X_i}$ (for $i \geq 2$) is associated with an exchange, so there exists X_i and $X_i^{'}$ such that $X_i^{'} = X_i + (e_i - e_{i-1})$. Then we have :

$$e_1 = X_1^{'} - X_1$$

and

$$e_i - e_{i-1} = X_i^{'} - X_i$$

for $2 \leq i \leq j$.
Summing all these j equalities, we obtain :

$$e_j = \sum_{i=1}^{j} (X_i^{'} - X_i). \tag{17}$$

We recall that C_j is the path going from the auxiliary element e_0 to the element e_j. We denote by A_j the set of arcs of the path C_j. We rewrite formula (17) as follows :

$$e_j = \sum_{\gamma \in A_j} (X_\gamma^{'} - X_\gamma) \tag{18}$$

where, if $\gamma = (s,t)_X$, we have $X_\gamma := X$, $X_\gamma^{'} := X + (t-s)$. If $\gamma = (e_0,t)_X$, that is t is a source, we have $X_\gamma := X$, $X_\gamma^{'} := X + \{t\}$.

4 Algorithm based on maximization of linear forms

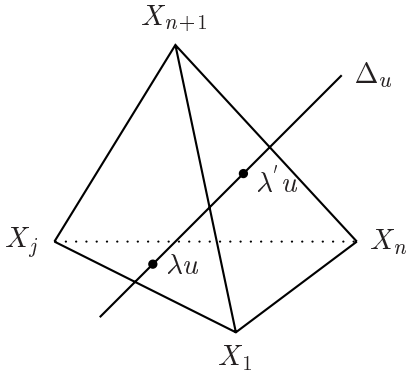
Let $\mathrm{Conv}(X_1, X_2, \ldots, X_n)$ denote the convex hull generated by $X_1, X_2, \ldots, X_n$. Let $S(P_r)$ be the set of vertices of the polytope P_r. Let J be a feasible basis of (PL). Suppose $J = \{1, 2, \ldots, n\}$. Let us consider the hyperplane H with equation $h(x) = 1$, generated by the vectors $X_j \in \mathbb{R}^n$, for $j \in J$. Let X_{n+1} be a vertex in $S(P_r))$ such that $h(X_{n+1}) = \max_{x \in S(P_r))} h(x)$.
-If $h(X_{n+1}) = 1$, the hyperplane corresponds to a facet of the polytope. The current solution y is optimal.
-If $h(X_{n+1}) > 1$, the vertex X_{n+1} lies strictly above the hyperplane. Let us consider the polytope $P := \mathrm{Conv}(X_1, X_2, \ldots, X_n, X_{n+1})$ and let H_j denote the hyperplane generated by $(X_1, \ldots, X_{j-1}, X_{j+1}, \ldots, X_n, X_{n+1})$ for $1 \leq j \leq n$. Let h_j be the affine equation $h_j(x) = 1$ of H_j. The intersection of the half-line Δ_u with the polytope P is a line segment $(\lambda u, \lambda^{'} u)$ with $\lambda < \lambda^{'}$. $\lambda^{'}$ is given by :

$$\lambda^{'} = \min_{i \text{ s.t. } h_i(u) < h(u)} \frac{1}{h_i(u)} \tag{19}$$

because $\lambda u = \frac{u}{h(u)}$ and $\frac{u}{h_i(u)}$ is the intersection of Δ_u with the hyperplane H_i.
There is an index $1 \leq j \leq n$ such that $\lambda^{'} u \in \mathrm{Conv}(X_1, \ldots, X_{j-1}, X_{j+1}, \ldots, X_n, X_{n+1})$. From (19) it follows j is given by :

$$h_j(u) = \max_{i \text{ s.t. } h_i(u) < h(u)} h_i(u)$$

Let $J^{'}$ denote the multi-index $\{1, 2, \ldots, j-1, j+1 \ldots, n, n+1\}$ and select the hyperplane H_j. Thus we have now a new feasible basis $J^{'}$ with $\lambda^{'} > \lambda$. The principle of our algorithm is to iterate this procedure as long as there exists a vertex strictly above the current hyperplane.



Let $< a, x >= \sum_{i=1}^{n} a_i x_i = 1$ be the equation of a selected hyperplane, then we have to solve the following problem : $\max < a, x >$ under the constraints $x(A) \leq r(A)$ for all $A \subseteq E$. Let us associate with each element $e_i \in E$ a weight equal to the coefficient a_i of the equation of the hyperplane. Then the researched vertex corresponds to an independent of maximal weight. It can be found with Edmond's greedy algorithm [6]. If $\max < a, x >= 1$, the obtained vertex is not strictly above the hyperplane. So the algorithm ends.

We give below a more formal description of the algorithm :
Algorithm
Step 1: $y = \lambda u = \sum_{i=1}^{n} \alpha_i X_i$ with $\sum_{i=1}^{n} \alpha_i = 1$. Compute the equation $h(x) = \sum_{i=1}^{n} a_i x_i = 1$ of the hyperplane H generated by $(X_1, \ldots, X_n)$. Find X_{n+1} solution of :

$$\begin{cases} \max\limits_{x \geq 0} < a, x > \\ \forall A \subseteq E, x(A) \leq r(A). \end{cases}$$

If $h(X_{n+1}) > 1$, go to step 2, otherwise go to step 3.
Step 2: Consider all the sets of the form $(X_1, \ldots, \hat{X}_i, \ldots, X_n, X_{n+1})$. For $i \neq n+1$, compute the linear equation h_i such that $h_i(X_k) = 1$ for $1 \leq k \leq n+1$ and $k \neq i$. Determine $j \neq n+1$ such that $h_j(u) = \max_{i \text{ s.t. } h_i(u) < h(u)} h_i(u)$. Change X_j to X_{n+1} and λ to $\lambda' = \frac{1}{h_j(u)}$. Go to step 1.
Step 3: End, the hyperplane H corresponds to a facet of the polytope.

We can start the algorithm with the vertices corresponding to the elements in E, thus $h(x) = \sum_{i=1}^{n} x_i$ and $\lambda = \frac{1}{\sum_{i=1}^{n} u_i}$. We now estimate the complexity of this algorithm. Let P' be the polytope $\text{Conv}(X_1, \ldots, X_{j-1}, X_{j+1}, \ldots, X_n, X_{n+1})$ we have constructed replacing X_j by X_{n+1}, that is $P' = \text{Conv}(X_1', \ldots, X_n')$ with $X_i' := X_i$ for $i \neq j$ and $X_j' := X_{n+1}$. We denote by h_i' the equation of the hyperplane generated by $X_1', \ldots, X_{i-1}', X_{i+1}', \ldots, X_n'$. P and P' have a common facet with equation $h_j(x) = h_j'(x) = 1$. Therefore (by linear

algebra) there exist real numbers t_i, $1 \leq i \leq n$, $i \neq j$ such that:

$$\begin{cases} h_i^{'} = t_i h_i + (1 - t_i) h_j \text{ for } i \neq j \\ h_j^{'} = h_j \end{cases} \tag{20}$$

The condition $h_i^{'}(X_{n+1}) = 1$ determines t_i:

$$t_i = \frac{h_j(X_{n+1}) - 1}{h_j(X_{n+1}) - h_i(X_{n+1})} \tag{21}$$

Hence it results from (20) and (21) we can compute $h_i^{'}$ in time $O(n)$ and all the $h_i^{'}$'s in time $O(n^2)$ when the h_i's and X_{n+1} are known. X_{n+1} which maximizes h_j on $P(\mathcal{M})$ is determined by the greedy algorithm in time γn. At each step, the running time is $O(n^2 + \gamma n)$. If N is the number of iterations in the algorithm, then the total running time is $O((n^2 + \gamma n)N)$. For a graphic matroid the running time is $O(n^2 N)$ ($\gamma = O(n)$ for a graph).

In the next section, we prove that, for a specific probabilistic model, this algorithm is a polynomial time algorithm.

4.1 Control of the increase in the algorithm

Let $g = \frac{1}{n} \sum_{i=1}^{n} X_i$ be the barycenter of the vertices $X_1, X_2, \ldots, X_n$ and let $\tilde{h}$ denote the homothetic transformation with center g and ratio k with $0 < k < 1$ and, for $1 \leq i \leq n$, let $X_i^{'}$ denote $\tilde{h}(X_i)$. $\text{Conv}(X_1^{'}, X_2^{'}, \ldots, X_n^{'})$ is the image of $\text{Conv}(X_1, X_2, \ldots, X_n)$ by the homothetic transformation $\tilde{h}$. Thus it is a subset of $\text{Conv}(X_1, X_2, \ldots, X_n)$. We have the following elementary result:

Proposition 4.1 *$x \in Conv(X_1^{'}, X_2^{'}, \ldots, X_n^{'})$ if and only if x can be written $x = \sum_{i=1}^{n} \alpha_i X_i$ with $\sum_{i=1}^{n} \alpha_i = 1$ (*i.e. *$x \in Conv(X_1, X_2, \ldots, X_n)$) and, for $1 \leq i \leq n$:*

$$\alpha_i \geq \frac{1 - k}{n}.$$

Particularly, if $k = 1 - \frac{1}{n}$, then $\alpha_i \geq \frac{1}{n^2}$.

In other words, the points in $\text{Conv}(X_1^{'}, \ldots, X_n^{'})$ lie at a distance $\geq \frac{1}{n^2}$ from the boundary of $\text{Conv}(X_1, \ldots, X_n)$.

Proof: The two convex combinations $x = \sum_{i=1}^{n} \alpha_i X_i = \sum_{i=1}^{n} \alpha_i^{'} X_i^{'}$ give the relation $\alpha_i = k\alpha_i^{'} + \frac{1-k}{n}$. $\alpha_i^{'} \geq 0$ is then equivalent to $\alpha_i \geq \frac{1-k}{n}$.

We recall $S(P_r)$ denote the set of vertices of the polytope P_r associated with the matroid.

Theorem 4.1 *At a given step of the algorithm, let h denote the equation of the affine hyperplane generated by the set of vertices $(X_1, X_2, \ldots, X_n)$*

of the polytope P_r and let X_{n+1} denote a vertex in $S(P_r)$ such that $M := \max_{x \in S(P_r)} h(x) = h(X_{n+1})$. Then :

$$M - 1 > \frac{1}{n|X_{n+1}|} \geq \frac{1}{n^2}.$$

Proof: We have $h(X_{n+1}) > 1$ (If $h(X_{n+1}) = 1$, the algorithm stops). We use the shortest paths associated with $G(J)$ in section 3.1. We have shown (cf. Proposition 3.3) that each element e_j of the set E of the matroid $\mathcal{M}$ can be expressed as a sum of the following form :

$$e_j = \sum_{\gamma \in A_j} (X'_\gamma - X_\gamma) \tag{22}$$

where A_j is the set of arcs of a shortest path C_j from e_0 to e_j in $G(J)$. X_γ is one of the vertices X_j's and X'_γ is a vertex of $P(\mathcal{M})$. The vertex X_{n+1} such that $h(X_{n+1}) = M$ can be expressed as :

$$X_{n+1} = \sum_{j=1}^{n} \beta_j e_j \tag{23}$$

where $\beta_j = 0$ or 1. By (22) and (23) and because $h(X_\gamma) = 1$, we have:

$$h(X_{n+1}) = \sum_{j=1}^{n} \left(\sum_{\gamma \in A_j} \beta_j (h(X'_\gamma) - 1) \right) = M. \tag{24}$$

As $M \geq h(X'_\gamma)$, (24) implies:

$$(M-1) \left(\sum_{j=1}^{n} \beta_j |A_j| \right) = \sum_{j=1}^{n} \left(\sum_{\gamma \in A_j} \beta_j (M-1) \right) \geq (M-1) + 1.$$

We can rewrite this last inequality :

$$(M-1) \left(\sum_{j=1}^{n} \beta_j |A_j| - 1 \right) \geq 1 \quad \text{or} \quad (M-1) \geq \frac{1}{\sum_{j=1}^{n} \beta_j |A_j| - 1}.$$

As a shortest path of $G(J)$ has at most $n-1$ arcs, we have $|A_j| \leq n$. $\sum_{j=1}^{n} \beta_j$ is the size of X_{n+1} denoted $|X_{n+1}|$. As $|X_{n+1}| \leq n$, we have finally :

$$M - 1 > \frac{1}{n|X_{n+1}|} \geq \frac{1}{n^2}.$$

This result does not always allow to control the increase $\lambda' - \lambda$ since $\lambda' u$ can be too much close to the face generated by $X_1, \ldots, X_n$ (cf the figure page 11). Actually, as it exists j such that $\lambda' u \in \text{Conv}(X_1, \ldots, \hat{X}_j, \ldots, X_n, X_{n+1})$, we have $\lambda' u = \sum_{i=1, i \neq j}^{n+1} \alpha'_i X_i$ with $\sum_{i=1, i \neq j}^{n+1} \alpha'_i = 1$, $\alpha'_i \geq 0$ and α'_{n+1} can be as small as possible. On the other hand, we can establish the following result :

Lemma 4.1 *If* $\lambda^{'}u = \sum_{i=1,i\neq j}^{n+1} \alpha_i^{'} X_i$, *then :*

$$\frac{\lambda^{'}}{\lambda} = 1 + \alpha_{n+1}^{'} \left[h(X_{n+1}) - 1\right] = 1 + \alpha_{n+1}^{'} \left(M - 1\right).$$

Particularly, if $\alpha_{n+1}^{'} \geq \frac{1}{n^2}$, *then* $\epsilon := \lambda^{'} - \lambda \geq \frac{1}{n^5}$.

Proof: Let $x = \lambda u = \sum_{i=1}^{n} \alpha_i X_i$ with $\sum_{i=1}^{n} \alpha_i = 1$ such that $h(x) = 1$ and $x^{'} = \lambda^{'} u = \sum_{i=1,i\neq j}^{n+1} \alpha_i^{'} X_i$ with $\sum_{i=1,i\neq j}^{n+1} \alpha_i = 1$. We then have :

$$h(x^{'}) = \sum_{i=1,i\neq j}^{n} \alpha_i^{'} h(X_i) + \alpha_{n+1}^{'} h(X_{n+1}) = \sum_{i=1,i\neq j}^{n} \alpha_i^{'} + \alpha_{n+1}^{'} h(X_{n+1})$$

as $h(X_i) = 1$ for $1 \leq i \leq n$. This implies (as $\sum_{i=1,i\neq j}^{n} \alpha_i^{'} + \alpha_{n+1}^{'} = 1$):

$$h(x^{'}) = \lambda^{'} h(u) = 1 + \alpha_{n+1}^{'} \left[h(X_{n+1}) - 1\right].$$

On the other hand $h(x) = \lambda h(u) = 1$. It follows :

$$\frac{\lambda^{'}}{\lambda} = 1 + \alpha_{n+1}^{'} \left[h(X_{n+1}) - 1\right]. \tag{25}$$

As we always have $\lambda' \geq \lambda$, the value of λ increases at each step. At the beginning of the algorithm, $\lambda = \frac{1}{u_1+u_2+\ldots+u_n} \geq \frac{1}{nu_n} \geq \frac{1}{n}$. Thus we always have $\lambda \geq \frac{1}{n}$. If we also have $\alpha_{n+1}^{'} \geq \frac{1}{n^2}$, we obtain $\frac{\lambda^{'}}{\lambda} \geq 1 + \frac{1}{n^4}$, then $\lambda^{'} - \lambda \geq \frac{\lambda}{n^4} \geq \frac{1}{n^5}$.

4.2 Probabilistic approach

If we always had $\alpha_{n+1}^{'} \geq \frac{1}{n^2}$, lemma 4.1 would imply the algorithm requires at most n^5 iterations. In this section we show it is effectively true in an appropriate probabilistic model. Let us consider a Bernoulli distribution such that, at each iteration, the increase $\lambda^{'} - \lambda$ is null or greater than $\frac{1}{n^5}$. More precisely, either $\alpha_{n+1}^{'} \geq \frac{1}{n^2}$ and $\lambda' u$ lies far from the boundary of the face of the polytope with probability p (we'll precise it afterwards) and the increase is at least $\frac{1}{n^5}$, or $\alpha_{n+1}^{'} < \frac{1}{n^2}$ with probability $q = 1 - p$ and the increase is null. Let x be such that :

$$x = \lambda u = \sum_{i=1}^{n} \alpha_i X_i$$

where $\alpha_i \geq 0$ for $1 \leq i \leq n$ and $\sum_{i=1}^{n} \alpha_i = 1$.
Let p denote the probability that x belongs to the set :

$$\mathrm{Conv}(X_1^{'}, \ldots, X_n^{'}) = \left\{ \sum_{i=1}^{n} \alpha_i X_i : \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \geq \frac{1-k}{n} \right\}.$$

A reasonable value for p is defined by :

$$p = \frac{\text{Volume}\{\sum_{i=1}^n \alpha_i X_i : \sum_{i=1}^n \alpha_i = 1, \alpha_i \geq \frac{1-k}{n}\}}{\text{Volume}\{\sum_{i=1}^n \alpha_i X_i : \sum_{i=1}^n \alpha_i = 1, \alpha_i \geq 0\}}.$$

where volumes are in the $(n-1)$-dimensional space. Then p is equal to the ratio of the volume of $\tilde{h}(\text{Conv}(X_1, X_2, \ldots, X_n))$ to the volume of $\text{Conv}(X_1, X_2, \ldots, X_n)$ where $\tilde{h}$ is the homothetic transformation with center g and ratio k :

$$p = \frac{\text{Volume}(\tilde{h}(\text{Conv}(X_1, X_2, \ldots, X_n)))}{\text{Volume}(\text{Conv}(X_1, X_2, \ldots, X_n))} = k^{n-1}.$$

At a given step of the algorithm, let us consider there are only two possibilities excluding each other. The first possibility is the event 1 such that :

$$\lambda u = \sum_{i=1}^n \alpha_i X_i, \sum_{i=1}^n \alpha_i = 1 \text{ and } \alpha_i \geq \frac{1-k}{n} \text{ for } 1 \leq i \leq n$$

with probability p. In this case, we'll say Δ_u lies far enough from the boundary of the facet $(X_1, X_2, \ldots, X_n)$. The second possibility is the event 0 such that :

$$\lambda u = \sum_{i=1}^n \alpha_i X_i, \sum_{i=1}^n \alpha_i = 1, \alpha_i \geq 0 \text{ and } \exists i \text{ such that } \alpha_i < \frac{1-k}{n}$$

with probability $q = 1 - p$.
Let N denote the number of iterations of the algorithm. Thus the probabilistic space we consider is $\{0,1\}^N$. We define an event w by a finite family $w = (w_1, w_2, \ldots, w_N)$ where $w_j = 0$ or 1. If $w_j = 1$ for l values of j, then the probability of w is :

$$P(\{w\}) = p^l q^{N-l}.$$

The random variable w_j (coordinate j of w), corresponds to the j-th polytope in the algorithm and is equal to 1 if $\{\lambda u\}$ lies far enough from the boundary of the face ($\alpha_i \geq \frac{1-k}{n}$ for $1 \leq i \leq n$) and is equal to 0 otherwise. We suppose the random variables w_j are independent. Now we consider the expected value of increase. At each step j, if $w_j = 1$ (that is Δ_u lies far enough from the edges of the selected facet of the j-th polytope) we progress of at least ϵ with probability p (we'll chose later $\epsilon = \frac{1}{n^5}$), otherwise we progress of 0 with probability $q = 1 - p$. Let $Y(w)$ denote the general increase during the event w (that is we have built N successive polytopes). We have :

$$Y(w) \geq l\epsilon$$

if during the event w, $w_j = 1$ for l steps and $w_j = 0$ for the $N - l$ other steps. The expected value of total increase after N steps is then :

$$E(Y) \geq \sum_w l\epsilon p^l q^{N-l} = \sum_{l=0}^N l\epsilon C_N^l p^l q^{N-l} = \epsilon \sum_{l=0}^N l C_N^l p^l q^{N-l}.$$

C_N^l is the number of events such that there has been exactly l real increases. We are in the classical Bernoulli scheme and it is easy to compute the expected value. Differentiating the binomial identity $\sum_{l=0}^{N} C_N^l x^l y^{N-l} = (x+y)^N$ with respect to x, we obtain, for $x = p$ and $y = q = 1-p$ the classical result :

$$E(Y) \geq \epsilon \sum_{l=0}^{N} l C_N^l p^l q^{N-l} = \epsilon N p (p+q)^{N-1} = Np\epsilon.$$

The algorithm will end after N steps in average when $E(Y) \geq 1$, in other words the average number N of steps of the algorithm verifies :

$$N \leq \frac{1}{p\epsilon}.$$

Let us consider for example $k = 1 - \frac{1}{n}$, then we have $p = (1-\frac{1}{n})^{n-1}$ and (according to lemma 4.1) $\epsilon \geq \frac{1}{n^5}$. As $(1-\frac{1}{n})^n$ is a nondecreasing function, we have $(1-\frac{1}{n})^n \geq (1-\frac{1}{2})^2 = \frac{1}{4}$. The average number of steps is then bounded by:

$$N \leq \frac{n^5}{(1-\frac{1}{n})^{n-1}} \leq 4n^5.$$

Theorem 4.2 *This algorithm based on maximization of linears forms on the matroid polytope will end on average after a number of steps less than or equal to $4n^5$. Its average running time is less or equal to $O(n^7 + \gamma n^6)$.*

When the most likely situation is not realized, Δ_u must be in a very specific position with regard to the boundary of the current convex set. So we may hope the algorithm is also polynomial in this case.

4.3 Case of a cycle and a tree

We have been able to entirely describe the progress of the algorithm in the cases of a tree and a cycle for which it requires $O(n)$ iterations and in the case of a tree and a cycle having a common vertex. In this last situation, the algorithm requires also $O(n)$ iterations when the edge of smallest weight belongs to the cycle. But in the reverse situation, it seems to require already $O(n^2)$ iterations and to present the difficulties of the general case. The study of these particular cases gives insight into a constructive method to describe all the steps of the algorithm in the case of a general graph. It consists in computing explicitly the equations of the linear forms $h_i(x) = 1$ of the different facets $(X_1, \ldots, \hat{X}_i, \ldots, X_n, X_{n+1})$ of the polytope P with vertices $(X_1, \ldots, X_n, X_{n+1})$ using Gaussian elimination to compute the inverse of a matrix. We show that, at each step, the vertex X_j we have to delete corresponds to the maximizer j in :

$$\boldsymbol{h_j(u) = \max_{i \text{ such that } h_i(X_i)<1} h_i(u).}$$

We have observed that these equations stay apparently quite simple, preserving a particular structure during the running time of this algorithm. At the moment, the number and the variety of all cases we have to consider prevent us from providing a complete classification of the equations h_i and so limit this method. All the proofs are in [12].

4.4 Numerical tests

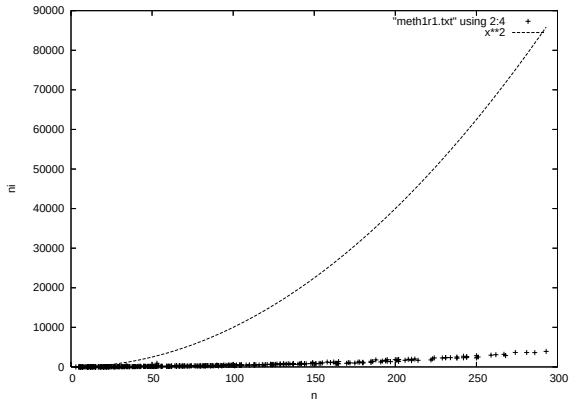
We have made some numerical tests with this algorithm in the case of a graphic matroid. The results indicate the algorithm ought to be polynomial with a number of iterations in $O(n^2)$. We have studied the number of iterations and the size of the tight set corresponding to the rank constraint obtained with the algorithm. We have also researched how these values are affected by the strengths of the edges and the graph density (the density of a graph $G(V,E)$ is equal to $\frac{2|E|}{|V||V-1|}$). We remind we can compute the strength of a graph by successive contractions of tight sets according to theorem 2.1. Therefore the sizes of the successive tight sets supplied by the algorithm have an influence on the speed of such a method. We have made five sequences of tests with different strengths. For each sequence, we have dealt with 2000 random graphs with expected number of edges varying between $|V|-1$ and $\frac{|V||V-1|}{2}$. In the following table, we give the average values obtained with three different sequences. $|S|$ represents the size of the tight set, ni the number of iterations.

strengths	density	$\|V\|$	$\|E\|$	$\|S\|$	$\frac{\|S\|}{\|E\|}$	ni	$ni \leq \|E\|^2$	$ni \leq \frac{\|E\|^2}{4}$
≤ 10	low	74	85	3	4%	1265	100%	94%
	intermediary	58	114	43	38%	5081	100%	38%
	high	12	62	59	96%	1073	100%	93%
≤ 100	low	75	87	3	4%	1852	100%	90%
	intermediary	58	114	43	38%	5441	100%	33%
	high	12	59	58	97%	1032	100%	89%
≤ 1000	low	70	81	3	4%	1591	100%	92%
	intermediary	57	113	44	39%	5399	100%	34%
	high	12	61	59	97%	1159	100%	82%

The number of iterations ni was sometimes very high but it always remained lower than $|E|^2$. In average, the number of iterations for low or high density graphs is inferior of almost five times the number of iterations required for intermediary density graphs. In average we contract 32% of the edges of the graph in the continuation of the algorithm. For low density graphs, we contract only 4% of the edges of the graph in average whereas, for high density graphs we contract more than 95% of the edges. Therefore the general algorithm for computing the strength will require few iterations of the

algorithm for a high density graph. On the other hand, it will probably require many iterations for a low density graph. Nevertheless, the computation time should stay acceptable since our results indicate the algorithm runs fast for low density graphs. When all the strength are equal to 1 the

Figure 1:



problem is in practice really easier to solve. In the graph of the figure 1, we see the number of iterations is quite lower than n^2 and algorithm 2 seems to be linear. It is also the case for low or high density graphs. More detailed comments can be found in [12].

5 Conclusion

We have been motivated in the study of the intersection of a half-line and the independent set polytope of a matroid by problems in the Telecommunication industry, suggested by Jérôme Galtier and Alexandre Laugier, where the strength of a network is a useful parameter. In this paper, we have proposed a new geometrical and very simple algorithm for this problem. We proved this algorithm requires in probability at most $4n^5$ iterations. We have established the progression is at least $\frac{1}{n^2}$ in the "normal" direction at each iteration of the algorithm. We would like to prove that this algorithm is polynomial and that, as it is suggested by our numerical tests, it has a good complexity. The polynomiality (in average) of this algorithm is essentially due to the particular structure of the considered polytopes. Thus one can hope to improve the complexity of this algorithm or to find alternative algorithms using the specificities of such polytopes. For instance, considering the importance of matchings in graph theory, it would be natural to study this algorithm for the matching polytope. The main result of this paper is a part of my thesis defended at Paris 6 University. I am grateful to my advisor Professor J. Fonlupt for valuable discussions and comments.

References

[1] F. Barahona. Separating from the dominant of the spanning tree polytope. *Operations Research Letters*, 12:201–204, 1992.

[2] F. Barahona. Network reinforcement. *Mathematical Programming*, 105(2–3):181–200, 2006.

[3] E. Cheng and W.H. Cunningham. A faster algorithm for computing the strength of a network. *Information Processing Letters*, 49:209–212, 1994.

[4] W.H. Cunningham. Testing membership in matroid polyhedra. *Journal of Combinatorial Theory, Series B*, 36:161–188, 1984.

[5] W.H. Cunningham. Optimal attack and reinforcement of a network. *J. ACM*, 32(3):549–561, 1985.

[6] J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1:127–136, 1971.

[7] J. Fonlupt and A. Skoda. Strongly polynomial algorithm for the intersection of a line with a polymatroid. *Book edited on the occasion of Bernhard Korte's 70th birthday (to be published)*, 2008.

[8] S. Fujishige. *Submodular functions and optimization*, volume 58 of *Annals of discrete mathematics*. Elsevier, 2005.

[9] H.N. Gabow. Algorithms for graphic polymatroids and parametric $\overline{s}$-sets. *J. Algorithms*, 26:48–86, 1998.

[10] D. Gusfield. Connectivity and edge-disjoint spanning trees. *Information Processing Letters*, 16:87–89, 1983.

[11] D. Gusfield. Computing the strength of a graph. *SIAM Journal on Computing*, 20(4):639–654, 1991.

[12] A. Skoda. *Force d'un graphe, multicoupes et fonctions sous-modulaires : aspects structurels et algorithmiques*. PhD thesis, Université Pierre et Marie Curie, Paris 6, 2007.