17

Graphic Submodular Function Minimization: A Graphic Approach and Applications

Myriam Preissmann and András Sebő

Summary. In this paper we study particular submodular functions that we call "graphic". A graphic submodular function is defined on the edge set E of a graph $G = (V, E)$ and is equal to the sum of the rank-function of G and of a linear function on E. Several polynomial algorithms are known that can be used to minimize graphic submodular functions and some were adapted to an equivalent problem called "Optimal Attack" by Cunningham. We collect eight different algorithms for this problem, including a recent one (initially developed for solving a problem for physics): it consists of $|V| - 1$ steps, where the i-th step requires the solution of a network flow problem on a subgraph (with slight modifications) induced by at most i vertices of the given graph ($i = 2, \dots, |V|$). This is a fully combinatorial algorithm for this problem: contrary to its predecessors, neither the algorithm nor its proof of validity use directly linear programming or keep any kind of dual solution. The approach is direct and conceptually simple, with the same worst case asymptotic complexity as the previous ones. Motivated by applications, we also show how this combinatorial approach to graphic submodular function minimization provides efficient solution methods for several problems of combinatorial optimization and physics.

17.1 Introduction

For basic graph theoretic notions, terminology and notation we refer to Lovász (1979). Given a finite set E, a function $f : 2^E \longrightarrow \mathbb{R}$ is called *submodular*, if for any two subsets $A, B \subseteq E$:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B). \qquad \text{(SUB)}$$

In honor of Bernhard Korte's 70-th birthday and in memory of the pioneering work and important achievements that have been reached concerning submodular functions in the Institut für Disckete Mathematik, Ökonometrie und Operations Research in the '80-s and '90-s, due to the generous visitor's program and high quality research and training of the institute led by Professor Korte, the results of which provide the background of the present work.

In this paper f will be called *modular* if the equality holds and $f(\emptyset) = 0$, that is, $f(A) = \sum_{a \in A} f(a)$. (We use here and in the following the notation a instead of $\{a\}$ when no confusion is possible.)

The submodular function f will be called *graphic* if E is the set of edges of a graph $G = (V, E)$ and moreover $f = r + w$, where $f, r, w : 2^E \longrightarrow \mathbb{R}$, $r = r_G$ is the rank-function of G, that is, for $X \subseteq E$, $r(X) := n - c_G(X)$, where $c_G(X)$ is the number of connected components of $G = (V, X)$ and w is any modular function. We will use the notation $n := |V|$ throughout.

It is well-known and easy to check that r is a submodular function, and since w is modular, f is also submodular. *Graphic submodular function minimization* is the following problem:

GRAPHIC submodular FUNCTION MINIMIZATION (GSM)
Input: graph $G = (V, E)$, $w : E \longrightarrow \mathbb{R}$
Output: $X \subseteq E$ that minimizes $r + w$.

As will be shown in the next sections, Graphic Submodular Function Minimization is useful for several applications such as computing the strength of a graph, solving the optimal reinforcement problem or approximating the partition function in the Potts model when the number of states tends to infinity. Our attention to Graphic Submodular Function Minimization was initiated by this last problem suggested by physicists. In an article that appeared in Journal of Physics A (Anglès d'Auriac et al. 2002) we already explain how it could be solved efficiently, allowing the solution of the problem for several hundred thousands of variables instead of one or two dozens provided by previous approaches. We give here the full details and mathematical background of this direct algorithm for graphic submodular function minimization. We also give a detailed analysis of the problem by comparing several (eight different) possible solution methods, their interconnections and their applications to physics or to some other problems. Notice that the previously known specific algorithms for Graphic Submodular Function Minimization were studied in the (equivalent) formulation of "Optimal Attack" (see Sect. 17.3.2).

For convenience we will most of the time consider the following equivalent problem, that we will call *Optimal Cooperation*:

$$\text{Maximize} \left\{ f_{G,w}(A) = c_G(A) + \sum_{e \in A} w(e) : A \subseteq E(G) \right\}. \qquad \text{(OC)}$$

This problem could indeed arise from the following "academic" situation. In a research institute, bilateral cooperations between researchers (vertices) are encouraged by the state: the *weight of an edge represents the reward for cooperation between two researchers. Furthermore there is a unit support for each research project (set of people connected by cooperations, no cooperation is possible outside a project).* So for instance there is a loss of support when two components unite unless the benefits from cooperations between the two sum up to at least 1. The goal of the researchers is to realize the cooperations in a way which maximize the total amount of money received from the state. Cooperate optimally!

Subtracting n and replacing w by $-w$ in the expression of $f_{G,w}$ we get exactly the function of (GSM) multiplied by -1, so (OC) is equivalent to (GSM) and maximizing $f_{G,w}$ $(= n - (r + (-w))$ is equivalent to minimizing the graphic submodular function $r + (-w)$. From this it is clear that $f = f_{G,w}$ is *supermodular*, that is,

$$f(A) + f(B) \leq f(A \cup B) + f(A \cap B). \qquad \text{(SUPER)}$$

When no confusion is possible we will simply use f or f_G for $f_{G,w}$, c for c_G.

When considering (OC) we can (and we will always) suppose $0 < w(e) < 1$ for all $e \in E(G)$, because if $w(e) \leq 0$, then *deleting* the edge, if $w(e) \geq 1$, then *contracting* it (identifying its two endpoints and keeping the other weights) we get an equivalent problem. We will call such w a *weight function* on the edges.

An important observation: if F is a set of edges, and X is the vertex set of a connected component of $G(F)$, then adding to F edges *induced* by X, that is with both endpoints in X, increases the value of f. Thus in an optimal solution $F^* \subseteq E$ of (OC) each connected component of F^* contains all the edges of G it induces.

Given a graph $G = (V, E)$ with edge-weighting w we will say that the *value* of a partition $\mathcal{P}$ of V is

$$g_{G,w}(\mathcal{P}) = |\mathcal{P}| + \sum \{w(xy); xy \in E(G) : x \text{ and } y \text{ are in the same class of } \mathcal{P}\}.$$

Another important observation: if $\mathcal{P}$ is a partition of V containing a set W such that the subgraph $G(W)$ induced by W is not connected, then replacing in $\mathcal{P}$ the set W by the sets of vertices of the connected components of $G(W)$ increases the value of $g_{G,w}$. Thus a partition $\mathcal{P}^*$ maximizing $g_{G,w}$ is such that each of its sets induces a connected subgraph of G.

For any subset F of edges let $w(F) = \sum_{e \in F} w(e)$ and denote by $\mathcal{P}_{G,F}$ (or simply $\mathcal{P}_F$ when no confusion is possible) the partition of $V(G)$ determined by the components of $G(V, F)$. For any set $\mathcal{S}$ of disjoint subsets of V we denote by $E(\mathcal{S})$ the subset of edges of E with both extremities in the same set of $\mathcal{S}$ and by $\delta(\mathcal{S})$ the set of edges of E joining vertices belonging to two different sets in $\mathcal{S}$. So for a partition $\mathcal{P}$ of V, $E(\mathcal{P})$ and $\delta(\mathcal{P})$ partition the edge-set E and we can write the preceding equation as:

$$g_{G,w}(\mathcal{P}) = |\mathcal{P}| + w(E(\mathcal{P})).$$

From the preceding observations it is clear that there is a one to one correspondence between subsets of edges maximum for $f_{G,w}$ and partitions of the vertices maximum for $g_{G,w}$. As will be shown in the following, it is possible to compute an optimal subset of *edges* using a greedy algorithm on the *vertices*.

By now several polynomial algorithms (Grötschel et al. 1981, 1988; Schrijver 2000; Iwata et al. 2001; Iwata 2002) are known to minimize a submodular function. However, it does not seem to be evident how these general algorithms could exploit the particularity of graphic submodular function minimization and provide an efficient algorithm for this case. Fortunately, the specific properties of our problem make possible the use of a considerably simpler and quicker algorithm.

Another possibility is to use matroids or polymatroids and "primal–dual methods". An algorithm for finding an optimal solution of (OC) follows from Edmonds' matroid partition algorithm (Edmonds 1965), but a careless reduction to matroid partition is not sufficient for the polynomial time bound. Cunningham (1984) generalized and sharpened Edmonds' algorithm to matroid rank functions shifted by a linear function, and this provides the first algorithm not using the ellipsoid method for maximizing $f_{G,w}$ with polynomial running time, as early as 1984. Cunningham (1985) also worked out a specialized algorithm to the "Optimal Attack problem".

A crucial progress about the 'Optimal Attack' problem has been made later on by Barahona (1992) and Baïou et al. (2000), which made possible to decrease the complexity of the problem with an order of magnitude: the consideration of edges one by one is replaced by the consideration of vertices. Both Cunningham's and Barahona's approach implicitly use the extension of an intersecting submodular function to a fully submodular function called 'Dilworth truncation'. Dilworth truncation was clarified in general by Lovász (1977) and was algorithmically worked out by Frank and Tardos (1988). The present algorithm is enlightened by a knowledge of these results on the Dilworth truncation and a network flow method for computing an underlying fully submodular function based on Picard and Queyranne (1982) and Padberg and Wolsey (1983). Moreover, the here presented new ideas of combining these methods and reducing them into some simple graph-theoretic steps result in an algorithm that has the best asymptotic worst case complexity, the same as (Baïou et al. 2000). It is probably also the simplest, both conceptually and in the use of computational resources of the implementation. The sophisticated ingredients were necessary for finding the right solution, but we did not need to make explicit use of them. It consists of a greedy algorithm with n steps on the *vertices* of the graph where the steps are max-flow-min-cut computations on graphs from 3 to at most $n+2$ vertices and very elementary manipulations on graphs. This can be seen as a greedy algorithm on a polymatroid indexed by the *vertices* of the graph where the rank oracle can be computed with network flows, and the whole procedure can be discussed in fully elementary graphic terms. The results of Barahona (1992) and Baïou et al. (2000) have already a vertex-centered framework but in a technically more complicated primal-dual context.

Let us finish this introduction by summarizing various polynomial algorithms for graphic submodular function minimization. We enumerate eight such algorithms, without mentioning possible smaller variations of each:

1–3. The three methods handling general submodular functions: one using the ellipsoid method and two recent combinatorial algorithms (Grötschel et al. 1981, 1988; Schrijver 2000; Iwata et al. 2001; Iwata 2002).

4. Cunningham's testing membership algorithm (Cunningham 1984) that sharpens Edmonds' matroid partition algorithm, to minimize the sum of a matroid function and of a linear function in strongly polynomial time.

5. Grötschel, Lovász and Schrijver's variant of Khachian's linear programming algorithm (Grötschel et al. 1981) proving the equivalence of optimization and

separation, where we plug in the subroutine PQPW for separation. (PQPW is explained in Sect. 17.2.2.)

6. Cunningham's 'Optimal Attack' algorithm (Cunningham 1985). This algorithm needs to solve $|E|$ maximum flow problems on a graph with $|V| + 2$ vertices.
7. Barahona's 'vertex-sensitive' algorithm (Barahona 1992) needs to solve at most $|V|$ maximum flow problems on a graph with $|V| + 2$ vertices.
   Another version of this algorithm proposed by Baïou et al. (2000) consists in solving $|V|$ maximum flow problems but on graphs which have, at iteration i, exactly $i + 2$ vertices.
8. The algorithm of Sect. 17.2.3 solving $|V|$ maximum flow problems on graphs which have, at iteration i, at most $i + 2$ vertices (but in average much less).

The last two methods implicitly use Frank and Tardos's algorithm for 'truncating' an intersecting submodular function (Frank and Tardos 1988) with a variant of PQPW (cf. Sect. 17.2.2).

In Sect. 17.2 we explain our approach and solution of the problem and restate several "classical" ones; in Sect. 17.3 we consider a few sample applications.

17.2 Solution

In this section we solve graphic submodular function minimization. In the first two subsections we provide the two main ingredients: the inductive step of extending a solution from the smaller graph that arises after the deletion of a vertex, and the known graph-theory procedure needed to execute this extension. The last two subsections make clear the complete algorithm and explain the origins of its ingredients.

17.2.1 Extension of a Solution

In this subsection we prove two theorems: the first is showing how a given graph G with weight function w and any set of edges maximizing (OC) for G minus a vertex, may be completed so that it maximizes (OC) for the graph G itself; the second shows how the optimizing set of edges changes if the weight of only one edge is increased.

The following lemma is an easy consequence of the (SUPER) property.

Lemma 2.1. *Let $G = (V, E)$ be a graph and $G' = (V, E')$ $(E' \subseteq E)$ an arbitrary subgraph of E. Let $F' \subseteq E(G')$ be an optimal solution of (OC) for G'.*

There exists an optimal solution of (OC) for G which contains all edges of F'.

Proof. We first remark that the restriction of f_G to the subsets of edges of G' is equal to $f_{G'}$, so we can simply omit the subscript.

Let F be any optimal solution of (OC) for G. By the (SUPER) property one has in G:

$$f(F \cup F') + f(F \cap F') \geq f(F) + f(F').$$

Since $F \cup F' \subseteq E(G)$, $F' \subseteq E(G')$, and F, F' maximize f in G and G' respectively, we have $f(F \cup F') \leq f(F)$, $f(F \cap F') \leq f(F')$, and therefore we have equality throughout. Now we see that $F \cup F'$ satisfies the conclusion. □

As a consequence of Lemma 2.1 we can obtain an optimal solution F^* for G by adding edges to F'. Each connected component in $G(F^*)$ will be either an element of $\mathcal{P}_{F'}$ or the union of at least two elements of $\mathcal{P}_{F'}$. The next lemma shows how the value of a partition is affected when a subset of connected components are put together.

For $U \subseteq V(G)$ we introduce the usual notation $\delta(U) := \delta(U, V \setminus U)$ for the set of edges with exactly one endpoint in U; in directed graphs $\delta(U)$ will denote the set of edges leaving U. For $\mathcal{W} \subseteq \mathcal{P}$ we will denote by $\delta(\mathcal{W})$ the set of edges of G joining vertices belonging to two different sets in $\mathcal{W}$.

Lemma 2.2. *Let $\mathcal{P}$ and $\mathcal{P}'$ be two partitions of V such that $\mathcal{P} = (\mathcal{P}' \setminus \mathcal{W}) \cup \{\bigcup X;\ X \in \mathcal{W}\}$ for some $\mathcal{W} \subseteq \mathcal{P}'$. Then*

$$g_G(\mathcal{P}) = g_G(\mathcal{P}') - \big(|\mathcal{W}| - 1 - w(\delta(\mathcal{W}))\big).$$

In particular, if $\mathcal{P}'$ is optimal then $|\mathcal{W}| - 1 - w(\delta(\mathcal{W})) \geq 0$, and if $\mathcal{P}$ is optimal, then $|\mathcal{W}| - 1 - w(\delta(\mathcal{W})) \leq 0$.

Proof. Replacing in $\mathcal{P}$ the sets of $\mathcal{W}$ by their union decreases the cardinality of $\mathcal{P}$ by $|\mathcal{W}| - 1$. On the other hand let A and A' be the sets of edges induced by the classes of $\mathcal{P}$ and $\mathcal{P}'$ respectively, $A = A' \cup \delta(\mathcal{W})$ and so the weight of the edges increases by $w(\delta(\mathcal{W}))$. The equality is then proved and the rest follows immediately. □

We will now use the above lemmas for proving the next two theorems.

Theorem 2.3. *Let $G = (V, E)$ be a graph with weight function w on the edges, let x be a vertex of G and let $G' = (V, E')$ be obtained from G by deleting all edges incident to x. The edges of G' keep their weight. Let $F' \subseteq E(G')$ be optimal for $f_{G'}$. For any $\mathcal{W}^* \subseteq \mathcal{P}_{F'}$ containing $\{x\}$ and minimizing $|\mathcal{W}| - 1 - w(\delta(\mathcal{W}))$ among all subsets $\mathcal{W}$ of $\mathcal{P}_{F'}$ such that $\{x\} \in \mathcal{W}$, the set $F^* = F' \cup \delta(\mathcal{W}^*)$ is optimal for f_G.*

We observe that, since x is an isolated vertex in G', the subset $\{x\}$ is an element of $\mathcal{P}_{F'}$ and F' is also optimal for $G \setminus x$.

Proof (of Theorem 2.3). By Lemma 2.1 we know that there exists F^* optimal for f_G which contains F'. Let $\mathcal{W} \subseteq \mathcal{P}_{F'}$ such that $W = \bigcup_{X_i \in \mathcal{W}} X_i$ is an element of $\mathcal{P}_{F^*}$. If $\mathcal{W}$ doesn't contain $\{x\}$ then, since F' is optimal for $f_{G'}$, we get by Lemma 2.2 that $|\mathcal{W}| - 1 - w(\delta(\mathcal{W})) \geq 0$ (this value is the same in G and G'). On the other hand F^* is optimal for f_G and therefore $|\mathcal{W}| - 1 - w(\delta(\mathcal{W})) = 0$, but then $F^* \setminus \delta(\mathcal{W})$ is also optimal for f_G. Hence there exists an optimal solution F^* for f_G containing F' and such that any element of $\mathcal{P}_{F^*}$ not containing x is already in $\mathcal{P}_{F'}$.

So any $\mathcal{W}^* \subseteq \mathcal{P}_{F'}$ containing $\{x\}$ and minimizing $|\mathcal{W}| - 1 - w(\delta(\mathcal{W}))$ among all subsets $\mathcal{W}$ of $\mathcal{P}_{F'}$ such that $\{x\} \in \mathcal{W}$ will provide an optimal solution $F' \cup \delta(\mathcal{W}^*)$. (Notice that this minimum is ≤ 0 since for $\mathcal{W} = \{\{x\}\}$ we get 0.) □

At this point we see that any way of finding an optimal $\mathcal{W}^*$ will provide a constructive algorithm for getting an optimal solution of (OC): we start with a solution for a small subgraph (for example a one vertex subgraph) and then add the vertices one by one, computing at each step an optimal solution. By Lemma 2.1 this can be done by extending the solution of the previous iteration.

Assume we are under the conditions of Lemma 2.1 and let $\{x\}, X_1, \ldots, X_k$ be the pairwise disjoint sets of $\mathcal{P}_{F'}$. We remark that the value $|\mathcal{W}| - 1 - w(\delta(\mathcal{W}))$ doesn't depend on the subgraphs induced by the subsets X_i in $\mathcal{W}$, so we may ignore these and work in a possibly smaller graph. More precisely: the result of *shrinking* $X_1, \ldots, X_k$ in G is the graph $\mathrm{shr}(G) = (V_{\mathrm{shr}}, E_{\mathrm{shr}})$ such that $V_{\mathrm{shr}} = \{x, x_1, \ldots, x_k\}$ where $x_1, \ldots, x_k$ are distinct new vertices, and the function $\mathrm{shr} : V \cup E \longrightarrow \{x, x_1, \ldots, x_k\} \cup E_{\mathrm{shr}}$ is defined by $\mathrm{shr}(v) := x_i$ if $v \in X_i$ and $\mathrm{shr}(x) := x$; the image of an edge is defined only for e with extremities a and b such that $\mathrm{shr}(a) \neq \mathrm{shr}(b)$ and $\mathrm{shr}(e) = \mathrm{shr}(a)\mathrm{shr}(b)$; edges keep their weight, that is $w_{\mathrm{shr}}(\mathrm{shr}(e)) = w(e)$; sets of vertices or of edges are replaced by the image sets. There is a one to one correspondence between the subsets $\mathcal{W}$ of $\mathcal{P}_{F'}$ containing $\{x\}$ and subsets W of vertices of $\mathrm{shr}(G)$ containing x and for any such $W = \mathrm{shr}(\mathcal{W})$ one has $|\mathcal{W}| - 1 - w(\delta(\mathcal{W})) = |W| - 1 - w_{\mathrm{shr}}(E_{\mathrm{shr}} \cap (W \times W))$. So $\mathcal{W}$ will be optimal if and only if $W = \mathrm{shr}(\mathcal{W})$ minimizes $|W| - 1 - w_{\mathrm{shr}}(E_{\mathrm{shr}} \cap (W \times W))$.

The following theorem follows also from Lemmas 2.1 and 2.2. It is useful for the *augmentation problem* that will be considered in Sect. 17.3.

Theorem 2.4. *Let $G = (V, E)$ be a graph with weight function w on the edges, let $e = xy$ be an edge of G and let w' such that $w'(e) > w(e)$ and $w'(f) = w(f)$ for any edge $f \neq e$. Let $F \subseteq E(G)$ be optimal for $f_{G,w}$. If $e \in F$ then F is also optimal for $f_{G,w'}$. If $e \notin F$ then for any $\mathcal{W}^*$ minimizing $|\mathcal{W}| - 1 - w'(\delta(\mathcal{W}))$ among all $\mathcal{W} \subseteq \mathcal{P}_F$ such that $x \in \bigcup_{X_i \in \mathcal{W}} X_i$, the set $F^* = F \cup \delta(\mathcal{W}^*)$ is optimal for $f_{G,w'}$.*

Proof. Clearly we cannot expect the optimal value to increase more than $w'(e) - w(e)$, so the case when $e \in F$ is trivial. Assume now that $e \notin F$. It is then obvious that F is optimal in $G' = G \setminus e$. So, by Lemma 2.1 we know that there exists an optimal solution F^* of $f_{G,w'}$ which contains F and we can then conclude exactly as in the proof of the previous theorem. □

17.2.2 Computing the Oracle

Given a graph H with weight function w on the edges and $W \subseteq V(H)$, we can define a vertex-function $b(W) := b_{H,w}(W) := |W| - 1 - w(E(W))$, where $E(W)$ denotes the set of edges of H with both extremities in W. (To avoid confusion, some letters (like b) denote functions on the vertices, and some other letters (like w) denote functions on the edges.)

It is clear from Theorem 2.3 that a solution to the following problem solves (OC): given (H, w) and a vertex $x \in V(H)$, find a subset W^* of $V(H)$ containing x such that $b(W^*) = \min(b(W); W \subseteq V(H), x \in W)$. Luckily, precisely this variant of the problem is directly solved as a key subroutine in Picard and Queyranne (1982) or Padberg and Wolsey (1983), using a network flow model that will be described below and that will be referred to as *PQPW* in the sequel.

We first give (or remind) some definitions and well-known facts. Given a directed graph and a subset X of its vertices, $\delta(X)$ denotes the set of arcs leaving X, and is called a *cut*; if $s \in X$, $t \notin X$ it is an (s,t)-cut; $\delta(X, Y)$ denotes the set of arcs oriented from X to Y. A function c of nonnegative value on the arcs of a directed graph is called a *capacity* function. If F is a set of arcs then $c(F) := \sum_{e \in F} c(e)$. A *network* is a directed graph with capacity function. A first efficient algorithm for finding the minimum capacity of an (s,t)-cut is by Ford and Fulkerson (1956) (see Schrijver 2003 for the complexity)—very efficient versions are known by now, see (Ahuja et al. 1993).

We state now the algorithm.

PQPW(H, w, x),
INPUT: Arbitrary graph H, $w : E(H) \longrightarrow (0,1)$, where $(0,1) = \{t \in \mathbb{R} : 0 < t < 1\}$, $x \in V(H)$.
OUTPUT: $W^* \subseteq V(H)$ so that $x \in W^*$ and $b(W^*) := b_{H,w}(W^*) := |W^*| - 1 - w(E(W^*))$ is minimum under the condition $x \in W^*$.

The first three steps describe the construction of a network $(D, c) = N(H, w, x)$ that will be associated to H, w and x:

1. $V(D) := V(H) \cup \{s, t\}$ where s, t are distinct new vertices;
2. To each edge $uv \in E(H)$ we associate the arcs (u, v) and (v, u) of capacities $c((u,v)) = c((v,u)) = \frac{1}{2}w(uv)$. To each vertex $u \in V(H)$ we associate the arcs (s, u) and (u, t).
3. Define for all $u \in V(H)$:
$$p(u) := \sum_{v \in V(H),\, uv \in E(H)} c((u,v)) = \frac{1}{2} \sum_{v \in V(H),\, uv \in E(H)} w(uv),$$
and then $c((s,u)) := p(u)$, $c((u,t)) := 1$.
Note that $\sum_{u \in V(H)} p(u) = w(E(H))$.
4. Determine in the directed graph D with the capacity function c a set $S \subseteq V(D)$ so that $s, x \in S$, $t \notin S$, and $c(\delta(S))$ is minimum. Return the set $W^* := S \setminus \{s\}$.

END;

The problem that has to be solved in Step 4 is a minimum cut problem in a graph with given nonnegative capacities: $s, x \in W^*$ can be enforced by contracting sx or putting an ∞ capacity on it.

Theorem 2.5. *The output of PQPW(H, w, x) is $W^* \subseteq V(H)$ containing x and minimizing $b(W^*)$ under this condition. This optimal value can be computed by one minimum cut computation on D.*

The proof of this theorem is a direct consequence of the following lemma (using the notations of the algorithm):

Lemma 2.6. *The value $c(\delta(\{s\} \cup W)) - b_{H,w}(W)$ $(W \subseteq V(H))$ is a constant independent of W.*

Proof. Let $W \subseteq V(H)$. Then $\delta(\{s\} \cup W)$ is an (s,t)-cut of $N(H, w, x)$ and we prove:

$$c(\delta(\{s\} \cup W)) = |W| - w(E(W)) + K = b(W) + K + 1, \qquad \text{(CUT)}$$

where $K := c(\delta(s)) = w(E)$.

Indeed, let us see how the capacity of the cuts changes if we start with the set $\{s\}$ inducing an (s, t)-cut of capacity $c(\delta(s)) = K$ and then 'add' to it the vertices of W one by one:

When we add v to the side of s, the edge sv of capacity $p(v)$ disappears, and the edge vt of capacity 1 appears, whence the change corresponding to such edges is $1 - p(v)$, and

$$\sum_{v \in W} 1 - p(v) = |W| - w(E(W)) - \frac{1}{2} \sum_{xy \in E(H), x \in W, y \notin W} w(xy).$$

On the other hand the contribution of the arcs between W and $V(H) \setminus W$ is clear, at the beginning it is zero, and at the end it is:

$$c(W, V(H) \setminus W) = \frac{1}{2} \sum_{xy \in E(H), x \in W, y \notin W} w(xy).$$

The change comparing to K is provided by the sum of the two contributions, which is $|W| - w(E(W))$. □

Notice that Lemma 2.6 is verified for any network obtained from the one in the algorithm by changing the capacities of the arcs from the source s and to the sink t in such a way that $c(u, t) - c(s, u) = 1 - p(u)$ for every vertex u.

Notice also that PQPW can be easily *generalized to minimize any modular shift of b, that is, to minimize $b + m$ where m is an arbitrary modular function on the vertex set.* The special case $m(W) := |W|$ $(W \subseteq V)$ worked out above is just a particular choice of a modular function. One only has to set $c(u, t) := m(u)$ instead of 1 in Step 3. The minimization of $b + m$ solved by PQPW for arbitrary m is exactly the 'oracle' that Frank and Tardos' truncation algorithm needs for 'truncating b', see (Frank and Tardos 1988), p. 526, remark. Our algorithm is a concatenation of the truncation algorithm combined with PQPW, with an elementary graphic solution of the truncation algorithm using new graphic ideas.

17.2.3 The Algorithm

In this section we state the algorithm solving the GSM problem, still in the OC form.

At each iteration a subset $U \subseteq V$ and a partition $\mathcal{P}$ of U will be at hand. In Step 0 we give trivial initial values; in Step 1, we choose an arbitrary vertex u to be added to U and through the following steps we *compute a subset $\mathcal{W}$ of $\mathcal{P} \cup \{u\}$ providing a new partition of $U \cup \{u\}$.*

OC algorithm

INPUT: A graph $G = (V, E)$, and a weight function $w\colon E \longrightarrow (0, 1)$.

OUTPUT: A partition $\mathcal{P}^*$ optimizing (OC): the set A^* of edges induced by the sets in $\mathcal{P}^*$ maximizes $\{f_{G,w}(A) := c_G(A) + \sum_{e\in A} w(e) : A \subseteq E\}$.

```
0. $U := \emptyset, \mathcal{P} := \emptyset$.
Do $n$ times consecutively Steps 1 to 5, and then define $\mathcal{P}^* = \mathcal{P}$:
    1. Choose a vertex $u \in V \setminus U$.
    2. Define $H$ and $w_H$ as the result of shrinking the classes of $\mathcal{P}$ in $G(U \cup \{u\})$ with
       weight function $w$ restricted to the edges of $G(U \cup \{u\})$.
    3. Call PQPW$(H, w_H, u)$. Suppose it outputs the set $W^* = \{u, x_1, \dots, x_k\} \subseteq$
       $V(H)$ where each $x_i$ is a vertex of $H$ corresponding to a set $X_i \in \mathcal{P}$.
    4. Define $R := \{u\} \cup X_1 \cup \dots \cup X_k$
    5. Redefine $U$ and $\mathcal{P}$: $U := U \cup \{u\}, \mathcal{P} := (\mathcal{P} \setminus \{X_1, \dots, X_k\}) \cup \{R\}$;
END.
```

Remarks.

- Note that W^* can be equal to $\{u\}$.
- The graphs H can also be constructed iteratively with only one shrinking in each iteration, by adding u and then shrinking the set W^*.
- The choice for u is completely free; this freedom could be used for making the computations simple.
- Since $f_{G,w}$ is a supermodular function (on the edges) we know that there exists a unique minimal optimal solution and a unique maximal optimal solution. Choosing in our $PQPW$ algorithm a minimal (respectively maximal) minimum (s,t)-cut we will obtain a minimal (respectively maximal) optimal solution. These minimum (s,t)-cuts are easy to obtain by starting from s (respectively t).

Theorem 2.7. *The output $\mathcal{P}^*$ is an optimal partition for (G, w).*

Proof. At step 0, $\mathcal{P} := \emptyset$ is an optimal solution for the subgraph of G induced by $U := \emptyset$. Assume now that $\mathcal{P}$ is an optimal solution for the subgraph of G induced by U, and show that after applying steps 1 to 5 the new partition is optimal for the subgraph of G induced by $U \cup \{u\}$. It is clear that $\mathcal{P}' = \mathcal{P} \cup \{\{u\}\}$ stays optimal when adding the vertex u but no edges. From the preceding chapter we know that $W^* = \{u, x_1, \dots, x_k\}$ is a subset of vertices of H containing u minimizing b_{H,w_H}. But H and w_H are obtained by shrinking the classes of $\mathcal{P}'$ in $(G(U \cup \{u\}), w)$, hence $\mathcal{W} = \{u, X_1, \dots, X_k\}$ is a subset of $\mathcal{P}'$ minimizing $|\mathcal{W}| - 1 - w(\delta(\mathcal{W}))$ and by Theorem 2.3 the partition $(\mathcal{P} \setminus \{X_1, \dots, X_k\}) \cup \{R\}$ is then optimal. □

Notice that the algorithm consists merely of $n - 1$ network flow computations and $n - 1$ shrinking. Our approach looks lucky in the sense that the network flow model is called for a small number of vertices: *in the beginning because the number of considered vertices is small, and at the end because hopefully many vertices are identified.* Moreover we don't need computing 'primal solutions' which are present in an essential way in the previous algorithms of the same complexity and so the capacities of the networks are not affected by the change.

17.2.4 Further Explanations

As already noticed, we can rewrite (OC) as:

$$\text{Maximize}\left\{ g(\mathcal{P}) = |\mathcal{P}| + \sum_{i=1}^{|\mathcal{P}|} w(E(V_i)) : \mathcal{P} = \{V_1, \dots, V_{|\mathcal{P}|}\} \text{ is a partition of } V \right\}.$$

Now maximizing g is equivalent to minimizing $n - g(\mathcal{P})$, moreover, noticing that $n - |\mathcal{P}| = \sum_{i=1}^{|\mathcal{P}|}(|V_i| - 1)$ we finally get the following formulation of our problem:

$$\text{Minimize}\left\{ \sum_{i=1}^{|\mathcal{P}|}(|V_i| - 1 - w(E(V_i))) : \mathcal{P} = \{V_1, \dots, V_{|\mathcal{P}|}\} \text{ is a partition of } V \right\}. \tag{DUAL}$$

Recall $b(U) = b_{G,w}(U) = |U| - 1 - w(E(U))$; (DUAL) is in fact 'an integral solution to the linear programming dual' of

$$\text{Maximize} \sum_{v \in V} x_v \quad \text{under the constraints}$$

$$\sum_{v \in U} x_v \le b(U) \quad \text{for every } U \subseteq V,\ U \neq \emptyset.$$

If $x \in \mathbb{R}^V$ satisfies these inequalities we will say that it is *feasible*. Note that x is then nonpositive! (The term $|U|$ could be deleted here, this causes a modular shift which influences (DUAL) only with the additive constant n; however, it is exactly the modular shift $b(U)$ that plays a role in the formulation (FOREST) considering subsets of edges, explained below.)

The algorithm we developed determines an optimal dual solution to this linear program. Another way of looking at the problem, that provides some insight and another way to solve it (even if a less efficient one) is through Edmonds' forest polytope:

Let $G = (V, E)$ be a graph, $w \in (0, 1)^E$, and for $F \subseteq E$ define $\chi_F(e)$ be 1 if $e \in F$ and 0 otherwise. According to a basic theorem of Edmonds (1970) the forest polytope defined as:

$$\begin{aligned}&\operatorname{conv}(\chi_F : F \text{ is a forest})\\ &\quad = \{x \in \mathbb{R}^E : x(E(U)) \le |U| - c_G(U),\ x \ge 0,\ (U \subseteq V(G))\},\end{aligned} \tag{FOREST}$$

is a particular polymatroid. Hence it has integer vertices (the forests of G), and the defining system of inequalities has an integer dual solution for any integer objective function.

Note the following:

(i) If $c_G(U) > 1$ then the corresponding inequality can be straightforwardly written as the sum of $c_G(U)$ others with $c_G(\cdot) = 1$, so *the only essential inequalities in* (FOREST) *are those with* $c_G(U) = 1$.

(ii) Edmonds (1970, 1971) proved (see also Schrijver 2003) that *a polymatroid intersected with upper bound constraints is still a polymatroid, and therefore it corresponds to a TDI system.*

(iii) The submodular function h, associated to the polymatroid defined by the constraints of (FOREST) plus the additional upper bound constraints $x \leq w$, is such that $h(\{e\}) = w(e)$ for all $e \in E$. Let us consider the objective function $\max \sum_{e\in E} x_e$ on this polymatroid. For each $e \in E$, there is a dual variable y_e corresponding to the upper bound constraint $x_e \leq w(e)$ and, for each U such that $c_G(U) = 1$ there is a dual variable y_U corresponding to the (FOREST) constraint on U. By (ii) there exists an optimal dual solution with $0 - 1$ coordinates and it is not hard to check that its value is equal to the minimum of (DUAL) + the sum of the upper bounds of all edges.

(iv) The new right hand sides are easy to express: $h(A) = \max\{\sum_{e\in A} x_e : x$ is in the forest polytope (FOREST), $x \leq w\}$, $(A \subseteq E)$. Similarly to (iii), the dual linear program provides a formula for computing h from b, and this problem is clearly the same again as GSM ("restricted to A"), so the greedy algorithm does not seem to be easily executed unless a good way of computing h recursively occurs.

Such a way exists, but first the "graphic" property has to enter the game. Since the constraints are associated to edge-sets induced by vertex-sets U, we can directly associate the new right hand sides to these vertex-sets, and this is what we did when we defined $b(U)$. The function b however is negative on the empty set, so the corresponding polymatroid would be empty. If we set $b(\emptyset) := 0$ then the new function is submodular only on intersecting pairs. Frank and Tardos (1988) compute (the rank oracle of) a new fully submodular function (which must be 0 on the $\emptyset$) that defines the same polyhedron, with an appropriate implementation of the greedy algorithm. In our case this corresponds to gradually growing U and computing "$h(E(U))$".

Algorithms for graphic submodular function minimization (Cunningham's, Barahona's and ours) can be all considered to be combinations of this greedy algorithm with a network flow model. We hope that by realizing this, and working out the details in a graphic way, the present paper provides an elementary, efficient and self-contained presentation of such an algorithm.

(v) In order to decide whether a given vector $w \in \mathbb{R}^E$ belongs to the forest polytope or not it is sufficient to know whether the minimum of $b(U) = |U| - 1 - w(E(U))$ is nonnegative for all U, $\emptyset \neq U \subseteq V$ or not. This problem is solved by PQPW. Using the solution as a separation subroutine for Grötschel, Lovász and Schrijver's equivalence of separation and optimization (Grötschel et al. 1981), and using that a dual optimum can also be computed in polynomial time see (Grötschel et al. 1988), this provides another way (mentioned in '5.' in Sect. 17.1) for solving our problem.

In short, our problem (DUAL) can be viewed as follows: one can separate from Edmonds' polytope efficiently in a combinatorial way (with PQPW) but it is not trivial to optimize on it combinatorially. However, this task can also be achieved combinatorially in a much simpler and more efficient way than general submodular function minimization, and this is exactly graphic submodular function minimization (GSM).

The algorithm has been implemented by our physicist coauthors (Anglès d'Auriac et al. 2002; Anglès d'Auriac 2004). Using the push-relabel algorithm (Goldberg and Tarjan 1988) for computing minimum cuts into PQPW, they were able to deal with grids of size 512×512 (Anglès d'Auriac et al. 2002), that is, more than 250 000 vertices.

17.3 Applications

17.3.1 Potts' Model

In this section we recall the Potts model (for a review of the huge amount of work devoted to it see Wu 1982) and show why the order of magnitude – or under a restrictive condition the approximate value – of the partition function is determined by the minimum value of the submodular function GSM if the number of 'states' tends to infinity. The partition function is an important quantity that encodes the statistical properties of a system in thermodynamic equilibrium.

A *lattice* of spins is given at each *site* of which lives a variable. Each variable $\sigma_1, \dots, \sigma_n$ ($n \in \mathbb{N}$) can take values in a given set $\mathbb{Z}_q := \{0, \dots, q-1\}$, $q \in \mathbb{N}$. Pairs of neighboring sites on the lattice are called *bonds*. We denote the number of sites by n, and the number of bonds by m.

Each configuration $\sigma = (\sigma_1, \dots, \sigma_n)$ has an energy

$$E(\sigma) = \sum_{ij} K_{ij} \delta_{\sigma_i \sigma_j}, \tag{1}$$

where $\sigma_i \in \mathbb{Z}_q$, the sum runs over all bonds ij, $K_{ij} \in \mathbb{R}_+$ is a given non-negative weight of bond ij, and δ_{ab} is the Kronecker symbol (1 if $a = b$ and 0 otherwise). (A lattice is a graph, sites are vertices, bonds are edges.) The aim is to compute or approximate the partition function

$$Z(K_{ij} : ij \text{ is a bond}) := \sum_{\sigma} \exp(E(\sigma)) = \sum_{\sigma} \prod_{ij} e^{K_{ij} \delta_{\sigma_i \sigma_j}}, \tag{2}$$

where the summation runs over all assignments of values $\sigma := (\sigma_1, \dots, \sigma_n) \in \mathbb{Z}_q^n$, and the product over all bonds ij.

We follow (Juhász et al. 2001). Note that $e^{K\delta} = 1 + (e^K - 1)\delta$ for $\delta \in \{0, 1\}$; introduce $\nu_{ij} = e^{K_{ij}} - 1$ and expand the product of sums:

$$\prod_{ij} e^{K_{ij} \delta_{\sigma_i \sigma_j}} = \prod_{ij} (1 + (e^{K_{ij}} - 1)\delta_{\sigma_i \sigma_j}) = \sum_{L} \prod_{ij \in L} \nu_{ij} \delta_{\sigma_i \sigma_j},$$

where the summation runs over all subsets of bonds L. Substituting this to (2) we get:

$$Z(K_{ij} : ij \text{ is a bond}) = \sum_\sigma \sum_L \prod_{ij \in L} \nu_{ij}\delta_{\sigma_i\sigma_j} = \sum_L \sum_\sigma \prod_{ij \in L} \nu_{ij}\delta_{\sigma_i\sigma_j}$$
$$= \sum_L q^{c(L)} \prod_{ij \in L} \nu_{ij}, \tag{3}$$

where the sum runs over all subsets L of bonds, and where $c(L)$ is the number of connected components of L on the set of all sites, counting also the isolated sites among the components. (By convention if L is empty then $\prod_{ij \in L} \nu_{ij} = 1$.) We got the last equality by counting the number of different $\sigma = (\sigma_1, \ldots, \sigma_n)$ for which the product is nonzero; since it is nonzero if and only if $\sigma(i) = \sigma(j)$ for every $ij \in L$, that is, if and only if σ is constant on every connected component of L, all possible σ can be given by choosing an element of Z_q for every connected component of L independently. Therefore we have exactly $q^{c(L)}$ such σ.

Note that this sum has 2^m terms, and that in (3) q does not need to have an integer value. (This provides a way of defining the Potts model for non-integer values.) Clearly, $K_{ij} \geq 0$ is equivalent to $\nu_{ij} \geq 0$, and we can introduce a new set of variables α_{ij} with

$$\nu_{ij} = q^{\alpha_{ij}},$$

and the partition function becomes

$$Z = \sum_L q^{c(L) + \sum_{ij \in L} \alpha_{ij}}. \tag{4}$$

Finally one introduces the function

$$f(L) = c(L) + \sum_{ij \in L} \alpha_{ij} \tag{5}$$

so that

$$Z = \sum_L q^{f(L)}. \tag{6}$$

As pointed out in Juhász et al. (2001), while q tends to infinity the sum in (6) is dominated by the maximum value of $q^{f(L)}$ and the order of magnitude of the partition function depends on the maximum of the function f that we have determined. (The asymptotic value of the maximum depends also on the number of optimal solution; nevertheless we have this optimum in the important case when the optimum is unique.)

17.3.2 Optimal Cooperation, Optimal Augmentation, Strength and Reinforcement

Cunningham (1985) studied Graphic Submodular Function Minimization in the context of Optimal Attack and defense of a network. Each edge has strength $s(e)$ which is a measure of the effort required by an attacker to destroy it. On another hand there is a benefit b for each new "disconnection". The goal of the attacker is to destroy a subset of edges A minimizing $s(A) - (c_G(E \setminus A) - 1)b$. This "Optimal Attack" problem is equivalent to Optimal Cooperation but with a 'complementary' viewpoint.

In the following we go further towards other applications mentioned in Cunningham (1985), and again, we provide a fully graphic interpretation for the sake of simplicity and practical efficiency. We will follow the language of "Optimal Cooperation".

The Augmentation Problem

Let us adopt the viewpoint of an employer ready to pay to stimulate the cooperation between researchers, by increasing the benefits of some pairwise cooperations (for instance with primes or rewards), for keeping the whole working group together. (The recent policy of the authors' employer, the CNRS, encourages indeed the creation of big institutes.) A cost function $k : E \longrightarrow \mathbb{R}_+$ expresses the cost of a unit increase in sponsoring a cooperation.

We say that (G, w) (or w) is *strong* if for the weight function w, $\{V\}$ is an optimal partition (for OC), and we can now state the problem more formally.

Augmentation Problem
Input: A graph $G = (V, E)$ with weight and upper bound functions $w, u : E \longrightarrow \mathbb{R}_+$, $w \le u \le 1$, cost function $k : E \longrightarrow \mathbb{R}_+$.
Output: A new weight function $w' : E \longrightarrow \mathbb{R}_+$, $w \le w' \le u$, such that (G, w') is strong, and $\sum_{e \in E} k(e)(w'(e) - w(e))$ is minimum; or a certificate of infeasibility.

If $u := 1$ everywhere, that is, if there are no upper bounds, the augmentation problem is called *unconstrained* otherwise *constrained*.

In some sense we already know a solution for this problem: $\{V\}$ is an optimal solution for (G, w') if and only if for the all 1 objective function, (FOREST) completed with the inequalities $x(e) \le w'(e)$ does not have a dual solution of smaller value than $|V| - 1$.

This means exactly that, $y_e := 0$ for all $e \in E$, $y_U := 0$ for all $U \subseteq V, U \ne V$, and $y_V := 1$ is an optimal dual solution. Clearly, this holds if and only if there exists a primal solution of the same value $|V| - 1$, that is, if and only if the polymatroid (FOREST) has a (noninteger) basis $w'' \le w'$:

For (G, w') *the set* $\{V(G)\}$ *is an optimal solution of* (DUAL) *if and only if* (FOREST) *has a basis* $w'' \le w'$.

We will say then that w' *contains a basis*. Weight functions w' which contain a basis form a "contrapolymatroid", that is, a set of vectors closed under adding nonnegative vectors, and such that the (coordinatewise) minimal elements satisfy the polymatroid basis axioms (Schrijver 2003). Equivalently, a contrapolymatroid is the set of nonnegative vectors satisfying the inequality system $x(U) \ge g(U)$ for all $U \subseteq E$, where g is a given supermodular function on 2^E.

These remarks lead to two possibilities for solving the augmentation problem. Both require to replace each edge e by two parallel edges e' and e'': e' has cost 0, weight 0 and upper bound $w(e)$, e'' has cost $k(e)$, weight 0 and upper bound $u(e) - w(e)$. We consider the polymatroid (FOREST) with the upper bound constraints associated to this new graph.

The first possibility is to start from the all 0 vector and to increase the variables one by one according to the polymatroid greedy rule (in increasing order of the costs). If at the end, the sum of the variables is equal to $|V| - 1$ then we increase the weight of each edge e by the value of the variable associated to e''; otherwise the problem is not feasible. Notice that this leads to an algorithm similar to the one of Cunningham (1985), but there is no need to introduce a new polymatroid like in Cunningham (1985).

The second possibility is to use that vectors containing a basis form a contrapolymatroid. If the vector with coordinates $x_{e'} = w(e)$, $x_{e''} = u(e) - w(e)$ for each e in E, is in this contrapolymatroid then decrease the coordinates greedily (in decreasing order of positive cost) according to the contrapolymatroid greedy algorithm (Schrijver 2003); else the problem is not feasible.

We rather work out a completely graphic version that extends naturally the approach of this paper.

Graphic Augmentation Algorithm

Input and **Output**: see above.

0. Give the initial value $w' := w$. Compute (using the OC algorithm) a maximal partition $\mathcal{P}$ such that $g_{G,w'}(\mathcal{P})$ is maximum. (That is: $\mathcal{P}$ is optimal but any partition obtained from $\mathcal{P}$ by merging elements of $\mathcal{P}$ is not. There exists a unique such partition, see the Remarks after OC algorithm)
   If $\mathcal{P} = \{V\}$, then STOP and output w', otherwise all edges induced by the classes of $\mathcal{P}$ are marked *fixed*, shrink all the classes of $\mathcal{P}$.
   H is assigned to be the arising graph, $E(H) \subseteq E(G)$ (contraction-minor).
1. Let $e = xy$ be an edge of H that is not fixed (in particular its endpoints are in different classes of $\mathcal{P}$) such that
   $$k(e) = \min\{k(f) : f \in E(G),\ f \text{ is not fixed}\}.$$
   The restriction to H of the current w' will be denoted by $w'_{|H}$.
2. Set $w'(e) := u(e)$, run PQPW$(H, w'_{|H}, x)$ to get the maximal output $S_e \subseteq V(H)$. Let $b := -b_{H,w'_{|H}}(S_e)$. Fix e, and
   - In case $S_e \neq \{x\}$:
     Set $w'(e) := u(e) - b$. If $S_e = V(H)$ then STOP and output w'; else fix e and all other edges induced by S_e, shrink S_e in H, redefine H as the resulting graph and GOTO 1.
   - In case $S_e = \{x\}$:
     If furthermore every edge is fixed STOP and return the corresponding partition of G as certificate of infeasibility, otherwise keep H and w' and GOTO 1.

END

The optimality of the algorithm (stated in the theorem below) relies on the following:

Lemma 3.1. *All along the algorithm, at the beginning of Step 1, the trivial partition $\{\{v\} : v \in V(H)\}$ is optimal for the current w' in H, and the corresponding partition of G is also optimal for (G, w'). Furthermore all these partitions have the same value than the first one.*

Proof. Indeed, just after Step 0 this is certainly true. We must show that this is true at the end of Step 2 where w' and H are redefined when considering the edge e. By Theorem 2.4 we know that after shrinking S_e in H, the vertices of the new H correspond to an optimal partition for (G, w') for w' obtained from the previous weight function by simply increasing the weight of e to its maximum possible value $u(e)$. In the case where $S_e \neq \{x\}$ this may be true even for a smaller weight of e. The value b corresponds to the gain of increasing the weight of e. A gain of 0 is enough for us, and we decrease the weight of e to the minimum which makes it enter into the partition. So, the new partition has the same value than the previous one. In the other case there is nothing to prove since the partition stays the same. So the Lemma is proved. □

Theorem 3.2. *If the algorithm stops with infeasibility, then all edges outside the partition $\mathcal{P}^*$ corresponding to the output have $w'(e) = u(e)$. Otherwise, the algorithm stops with $w \leq w' \leq u$ where w' is strong, and $k^\top(w' - w)$ is smallest among any strong w' satisfying these bounds.*

Proof. As pointed out above, the validity of the greedy algorithm follows from the fact that (FOREST) with upper bounds w' is a polymatroid. It can also be checked directly in terms of the graph.

If, at the end of the algorithm, H has only one vertex, then according to Lemma 3.1, w' is strong. If not, then $w'(e) = u(e)$ for every edge not induced by a class of the partition $\mathcal{P}^*$ when the algorithm stops, and so this maximal partition has a bigger value than $\{V\}$ for any $w' \leq u$. □

In case of the unconstrained Augmentation Problem we get an algorithm which has the same complexity as at most n minimum cut computations on networks of decreasing sizes. Indeed, in that case, each time a new partition $\mathcal{P}'$ is computed at Step 2, its cardinality is strictly smaller than that of the previous partition $\mathcal{P}$. Note that the minimum cost spanning tree problem is a special case of the unconstrained Augmentation Problem.

Extrapolating the success of the implementation of the OC algorithm in Anglès d'Auriac et al. (2002) (solution for several hundred thousand variables) this version of the reinforcement problem should also be efficient. However the advantages that make our algorithm run quickly (the small size of the current graphs) cannot be converted into a worst-case bound. The asymptotic worst case bound for the running time of the unconstrained case is the same as that of Barahona (2006).

Unfortunately, in the case of constrained case, we may have to perform much more minimum cut computations, m in the worst case, since we increase the weights of edges one by one. For this problem, Barahona (2006) performs the computation of a sequence of at most m minimum cuts with a time-complexity of only n times the complexity of one push-relabel algorithm, by trading work for memory space: this algorithm uses a "parametric flow" technique (Gallo et al. 1989), which involves keeping in memory n copies of the original graph. In our algorithm the number of vertices of the different networks we use will decrease from $n+2$ to 4, except if the problem is not feasible, but this last case is easy to test by running the OC algorithm separately with u as weight function. (If the maximal optimal partition is not $\{V\}$ then there exists no feasible solution.)

The reason for which we cannot use the parametric flow technique is precisely the shrinking of subsets of vertices. However in case the weights have a small common denominator q, typical in the application in physics, we may use a similar technique using "augmenting paths" instead of a "push-relabel" algorithm for computing the maximum flows and as in Barahona (2006) the complexity of the algorithm can be improved at the cost of a higher space complexity: during the Augmentation Algorithm we maintain at most n networks $N(H, w', x)$ of PQPW type obtained from the initial one by shrinking subsets of vertices and increasing the weights of some of the edges. We keep a feasible flow in each of these networks and an (s,t)-cut of weight $n = |V|$ induced by $V \setminus t$. Then the value of the maximum flow will never exceed n and so the algorithm will perform at most nq augmentations in each network and we get a complexity of $O(n^2mq)$. The way we construct the networks is as follows. At Step 2, we increase the capacities accordingly to $w'(xy) := u(xy)$. Obviously the flow we had before is still compatible. In case S_e does not contain y we just need to increase the capacities similarly in all current networks. In the other case we discard all networks $N(H, w', v)$ such that $v \in S_e \setminus y$. In $N(H, w', y)$ we simply shrink S_e into y, and keep the same flow. For all other networks $N(H, w', v)$ $v \notin S_e$ we do the following. We shrink S_e in y. We set $c(sy) := |S_e| - 1 + \sum_{ab \in E, a \in S_e, b \notin S_e} w'(ab)/2$. Since S_e was not a set of the previous maximal optimal partition we have $|S_e| - 1 > \sum_{ab \in E, a, b \in S_e} w'(ab)/2$ and so the flow obtained by setting $f(sy) := \sum_{ab \in E, a \in S_e} f(s,a)$ is a feasible flow. Notice that the new networks are not exactly as defined in PQPW but the condition $c(a,t) - c(s,a) = 1 - p(a)$ is fulfilled for every vertex a, which is sufficient. Another improvement of the m minimum cut computations algorithm of Cunningham (1985) is due to Gabow (1998) who proposed an algorithm computing a minimum-cost base of a graphic polymatroid in time $O(n^2 m \log(n^2/m))$.

Strength

Cunningham (1985) defined the strength and the reinforcement of a graph—we translate these into the "Optimal Cooperation language" and show how they can be computed using graphic submodular function minimization and (constrained) augmentation.

Given an undirected graph $G = (V, E)$ where each edge e has a nonnegative strength $s(e)$, the *strength* of (G, s) is defined as

$$\sigma(G, s) := \max\{\lambda : (G, s/\lambda) \text{ is strong}\}.$$

(It is easy to see that the maximum exists.)

Cunningham showed that the strength of a graph can be computed by solving at most n Optimal Cooperation problems, that is, after executing n GSM algorithms.

Indeed, let σ be the strength of the graph $G = (V, E)$ with strength function s on the edges. By definition of σ we have $f_{G,s/\sigma}(E) \geq f_{G,s/\sigma}(\emptyset)$ and so $\sigma \leq s(E)/(n-1)$. So if E is optimal for the weights $\frac{(n-1)}{s(E)}s$ then $\sigma = s(E)/n-1$ and we stop. Else $\sigma < s(E)/n-1$, that is, dividing the weights by $s(E)/n-1$ some edges are too light to enter into an optimal solution and the maximal optimal solution A is strictly included in E. By Theorem 2.4 we know that while the weights are augmented, the edges of A will stay in an optimal solution, so we can simply shrink all sets of $\mathcal{P}(A)$ in order to get a new graph G' with strength function $s_{|G'}$ of the same strength as G, s and we repeat this procedure until E becomes optimal. Notice that $|V(G')| < |V(G)|$ since else E would be optimal; and so we will stop after solving at most n Optimal Cooperation problems. This algorithm performs virtually the same actions as the one of Cunningham except that once again the sizes of the problems to be solved will be decreasing. Similarly to "parametric flow" for computing the strength (Gusfield 1991; Cheng and Cunningham 1994), we can keep at most n networks with increasing capacities and feasible flows, but in that case a small common denominator of the capacities is unlikely. However we can avoid this problem by solving the generalized Optimal Cooperation problem where the benefit of a connected component is a constant B which may be different from 1, that is

$$\text{Maximize}\left\{f_{G,w,B}(A) = Bc_G(A) + \sum_{e\in A} w(e),\ A \subseteq E(G)\right\}.$$

By dividing the weights by B we are lead to the usual Optimal Cooperation problem, but we can also solve it directly applying PQPW to a modular shift of b (as in Cheng and Cunningham 1994). In that case we obtain an algorithm of complexity $O(n^2 mBq)$. Gabow (1998) proposed an algorithm to compute the strength of a graph in time $O(n^2 m \log(n^2/m))$ and space only $O(m)$ (instead of $O(nm)$ for the other algorithms of the same complexity).

Let us finally check that our definition of the strength is the same as Cunningham's:

By definition (G, s) is of strength at least σ if and only $1 + s(E)/\sigma \geq c(A) + s(A)/\sigma$ for all $A \subseteq E$, that is, if and only if $\sigma s(E \setminus A) \geq \sigma(c(A)-1)$. it follows that

$$\sigma(G, s) = \min\left\{\frac{s(E-A)}{c(A)-1} : A \subseteq E, c(A) > 1\right\}.$$

Reinforcement

We consider now the minimum cost reinforcement problem:

Reinforcement
Input: A graph G with strength function $s : E \longrightarrow \mathbb{R}_+$, cost function $k : E \longrightarrow \mathbb{R}$, $k \geq 0$ and upper bound function $l \geq s$ on the edges, and $\sigma_0 > 0$.

Output: A new strength function $s' : E \longrightarrow \mathbb{N}$, $s \leq s' \leq l$, such that $\sigma(G, s') \geq \sigma_0$ and $\sum_{e \in E} k(e)(s'(e) - s(e))$ is minimum; or a certificate of infeasibility.

Clearly, according to the definition, the reinforcement problem (G, s, l, k, σ_0) can be solved by finding an optimal solution w' of the Augmentation Problem on $(G, w = \frac{s}{\sigma_0}, u = \frac{l}{\sigma_0}, k)$. Indeed, then $s' = w'\sigma_0$ is an optimal reinforcement. If there is no solution for the Constrained Augmentation Problem, then no reinforcement exists.

17.4 Conclusion

This is a thoroughly revisited version of the mathematical part of an article that appeared on this problem in Journal of Physics A. It is completed here with full details of the mathematical part of the paper, by more mathematical background, and also new results concerned by the minimization of a graphic submodular function like the problems of augmentation, strength or reinforcement.

Acknowledgements

The algorithms have been worked out for the application in physics (Anglès d'Auriac et al. 2002) with an active participation of our physicist colleagues. We thank them for helping us helping them! We also thank the referees for their thorough reading and helpful comments.

References

Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms and Applications. Prentice Hall, Englewood Cliffs (1993)

Anglès d'Auriac, J.-C., Iglói, F., Preissmann, M., Sebő, A.: Optimal cooperation and submodularity for computing Potts'partition functions with a large number of states. J. Phys. A, Math. Gen. **35**, 6973–6983 (2002)

Anglès d'Auriac, J.-C.: Computing the Potts free energy and submodular functions. In: Hartmann, A.K., Rieger, H. (eds.) New Optimization Algorithms in Physics, pp. 101–117. Wiley, New York (2004)

Baïou, M., Barahona, F., Mahjoub, A.R.: Separation of partition inequalities. Math. Oper. Res. **25**(2), 243–254 (2000)

Barahona, F.: Separating from the dominant of the spanning tree polytope. Oper. Res. Lett. **12**, 201–203 (1992)

Barahona, F.: Network reinforcement. Math. Program. **105**, 181–200 (2006)

Cheng, E., Cunningham, W.H.: A faster algorithm for computing the strength of a network. Inf. Process. Lett. **49**, 209–212 (1994)

Cunningham, W.H.: Testing membership in matroid polyhedra. J. Comb. Theory, Ser. B **36**, 161–188 (1984)

Cunningham, W.H.: Optimal attack and reinforcement of a network. J. Assoc. Comput. Math. **32**(3), 549–561 (1985)

Edmonds, J.: Minimum partition of a matroid into independent sets. J. Res. Natl. Bur. Stand. Sect. B **69**, 73–77 (1965)

Edmonds, J.: Submodular Functions, Matroids, and certain polyhedra. In: Guy, R., Hanani, H., Sauer, N., Schönheim, J. (eds.) Combinatorial Structures and Their Applications, pp. 69–87. Gordon and Breach, New York (1970)

Edmonds, J.: Matroids and the greedy algorithm. Math. Program. **1**, 127–136 (1971)

Ford, L.R., Fulkerson, D.R.: Maximum flow through a network. Can. J. Math. **8**, 399–404 (1956)

Frank, A., Tardos, É: Generalized polymatroids and submodular flows. Math. Program. **42**, 489–563 (1988)

Fujishige, S.: Submodular Functions and Optimization. Annals of Discrete Mathematics, vol. 47. North-Holland, Amsterdam (1991)

Gabow, H.N.: Algorithms for Graphic Polymatroids and Parametric $\bar{s}$-Sets. J. Algorithms **26**, 48–86 (1998)

Gallo, G., Grigoriadis, M.D., Tarjan, R.: A fast parametric maximum flow algorithm and applications. SIAM J. Comput. **18**(1), 30–55 (1989)

Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flow problem. J. Assoc. Comput. Math. **35**(4), 921–940 (1988)

Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method, and its consequences in Combinatorial Optimization. Combinatorica **1**, 169–197 (1981)

Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Berlin (1988)

Gusfield, D.: Computing the strength of a graph. SIAM J. Comput. **20**(4), 639–654 (1991)

Iwata, S.: A fully combinatorial algorithm for submodular function minimization. J. Comb. Theory, Ser. B **84**, 203–212 (2002)

Iwata, S., Fleischer, L., Fujishige, S.: A combinatorial strongly polynomial algorithm for minimizing submodular functions. J. Assoc. Comput. Math. **48**, 761–777 (2001)

Juhász, R., Rieger, H., Iglói, F.: The random-bond Potts model in the large-q limit. Phys. Rev. E **64**, 56122 (2001)

Lovász, L.: Flats in matroids and geometric graphs. In: Cameron, P.J. (ed.) Combinatorial Surveys, Proceedings of the 6th British Combinatorial Conference, pp. 45–86. Academic Press, London (1977)

Lovász, L.: Combinatorial Problems and Exercises. North-Holland/Akadémiai Kiadó, Amsterdam (1979)

Padberg, M.W., Wolsey, L.A.: Trees and cuts. Ann. Discrete Math. **17**, 511–517 (1983)

Picard, J.-C., Queyranne, M.: Selected applications of minimum cuts in networks. Int. Syst. Oper. Res. **20**, 394–422 (1982)

Schrijver, A.: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. J. Comb. Theory, Ser. B **80**, 346–355 (2000)

Schrijver, A.: Combinatorial Optimization. Springer, Berlin (2003)

Wu, F.Y.: The Potts model. Rev. Mod. Phys. **54**, 235–268 (1982)