N^{o}64

OPTIMA

Mathematical Programming Society Newsletter

SEPTEMBER 2000

Recent Progress in Submodular Function Minimization

Lisa Fleischer*

August 2, 2000

Last summer, two independent papers [31, 41] gave a positive answer to a question posed almost two decades ago: is there a polynomial time, combinatorial algorithm to minimize a submodular function?

This article is an attempt to relate the history and importance of the problem, the difficulties that arose in confronting it, the motivation for the solutions proposed, some consequences of the existence of these new algorithms, and some further challenges.

*Graduate School of Industrial Administration, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213. This article was written while the author was a faculty member at the Department of Industrial Engineering and Operations Research, Columbia University, New York. Additional support provided by NSF through grant EIA-9973858.

In the next issue of OPTIMA: *The Atlanta Symposium.*

64

Recent Progress in Submodular Function Minimization

1. Introduction

A submodular function f is a set function defined on all subsets of a discrete set V (denoted 2^V) that satisfies the following inequality for every pair of subsets $A, B \subseteq V$:

$$f(A)+f(B) \geq f(A \cup B)+f(A \cap B). \qquad (1)$$

Lovász showed that every submodular function has a natural extension to the nonnegative reals that is convex [35]. As with convexity, submodularity is a property of many naturally occurring functions in engineering and economics, and it is preserved under many natural transformations. Submodular functions arise as cut functions in a graph (see Figure 1), as matroid rank functions, and have applications in areas including game theory, information theory, and graph theory. An in-depth treatment of submodular functions and applications can be found in surveys of Lovász [35], Fujishige [20], Topkis [45], and Frank and Tardos [17].

Examples of Submodular Functions.

The cut function in a graph: Given an undirected graph G on vertex set V, define Δ on a subset of vertices $A \subseteq V$ by $\Delta(A)$:=number of edges with exactly one endpoint in A. Then Δ is a submodular function on 2^V. Similarly, if G is a directed graph, then the function $\vec{\Delta}$ defined by $\vec{\Delta}(A)$:= number of arcs leaving A, is a submodular function.

The $\{s; t\}$-cut function in a graph: Given graph G on vertex set $V \cup \{s, t\}$, define Δ_t^s on a subset $A \subseteq V$ by $\Delta_t^s(A)$:= the number of edges with exactly one endpoint in $A \cup \{s\}$. Then Δ_t^s is a submodular function on 2^V. The appropriately defined directed version, $\vec{\Delta}_t^s$, is also submodular.

The rank function of a matroid: A *matroid* is defined by a ground set M and set of independent subsets $\mathcal{I}$ that satisfy $\emptyset \in \mathcal{I}$, $I \subseteq J \in \mathcal{I}$ implies $I \in \mathcal{I}$, and $I, J \in \mathcal{I}$, $|I| < |J|$ implies $I \cup \{e\} \in \mathcal{I}$ for some $e \in J \backslash I$. Then the *rank function r*, defined on a subset A to be the size of the largest independent set contained in A, is a submodular function.

Figure 1.

Given submodular function f on V, a *minimizer* of f is a subset $A \subseteq V$ such that $f(A) \leq f(B)$ for all other subsets $B \subseteq V$. *Submodular function minimization* (*SFM*) is the problem of finding a a minimizer of f. As with convex functions, it is possible to find a minimizer of a submodular function by starting from an arbitrary suboptimal point and following any sequence of feasible steps in improving directions. In addition, for a submodular function f with $f(\emptyset) = 0$, a minimizer of Lovász's convex extension of f restricted to the 0-1 cube corresponds to a minimizer of f of the same value [35].

The classic problem of finding a minimum $\{s, t\}$-cut in a graph is a special case of submodular function minimization. The minimum $\{s, t\}$-cut problem is efficiently solved via combinatorial algorithms invoking the strong duality relation of minimum cut equals maximum flow. For the minimum cut problem, all efficient algorithms use the structure of the graph. Imagine now having only an oracle that, given a set of vertices, returns the value of the associated cut. How could you use this oracle to find the minimum cut? While this problem is easier than general submodular function minimization because it is possible to recover the graph [4], it gives the essence of the problem: given an oracle that returns the value on any subset of V, find the minimizing subset.

Another special case of submodular function minimization arises in the field of matroids: find the maximum cardinality independent set of two matroids defined on the same ground set. This is the *matroid intersection problem* of which maximum bipartite matching is a special case. Let $M_1 = (\mathcal{I}_1, V)$ and $M_2 = (\mathcal{I}_2, V)$ be two matroids where $\mathcal{I}_1$ and $\mathcal{I}_2$ are the set of independent sets of M_1 and M_2 respectively. Denote the rank function of M_i by r_i. The matroid intersection theorem of Edmonds says that

$$\max\{|J| \mid J \in \mathcal{I}_1 \cap \mathcal{I}_2\} =$$

$$\min\{r_1(A) + r_2(V \backslash A) \mid A \subseteq V\} \qquad (2)$$

Since $r_1 + r'_2$ defined by $(r_1 + r'_2)(A) := r_1(A) + r_2(V \backslash A)$ is submodular, the cardinality of the maximum sized independent set can be determined by submodular function minimization.

While the minimum $\{s, t\}$-cut problem and matroid intersection problem both have combinatorial polynomial-time algorithms [15, 3], until recently, general submodular function minimization did not. In 1981, Grötschel, Lovász, and Schrijver established the polynomial time equivalence of separation and optimization for combinatorial problems via the ellipsoid method [26]. One of the major implications of this

The Greedy Algorithm and Matroids

The *greedy algorithm* for submodular functions is a natural extension of the greedy algorithm for matroids. Given a vector $c \in \mathbf{R}^V$, index the elements of V by decreasing c-value, so that $c(v_1) \geq c(v_2) \geq \cdots \geq c(v_n)$. This defines a total order $<$ of V by $v_i < v_{i+1}$. Iteratively set $x(v_i)$ to be the maximum possible amount, subject to having maximized the first $i-1$ components of x. Mathematically, define $<^{v_i} := \{v_1, \cdots, v_i\}$. Then the greedy algorithm sets $x(v_i) = f(<^{v_i}) - f(<^{v_{i-1}}) = f(<^{v_i}) - \sum_{j=1}^{i-1} x(v_j)$. In other words, greedy sets $x(<^{v_i}) = f(<^{v_i})$ for each set $<^{v_i}$. Any set with this property is called x-tight. The submodularity of f implies that the x determined in this manner is an element in P(f). Since these equations are linearly independent, it follows that the greedy algorithm finds an extreme point of P(f). This extreme point x maximizes c^Tx over all points in P(f). Each extreme point of P(f) is obtained by applying the greedy algorithm to some order $<$ of V [8, 42]. The greedy algorithm for matroids is the special case of this algorithm when f is the rank function of a matroid.

The submodular polyhedron P(f) is a natural extension of the *matroid polyhedron*, which is the convex hull of incidence vectors of independent sets of a matroid. (The incidence vector χ_A of a set A is defined by $\chi_A(v) = 1$ if $v \in A$ and $\chi_A(v) = 0$ if $v \notin A$.) While the matroid polyhedron is restricted to the nonnegative orthant and all vertices of the matroid polyhedron are $\{0,1\}$ vectors, the submodular polyhedron contains the negative orthant in its extreme cone, and has vertex coordinates that depend on the values taken by the submodular function. Figure 3 contains an example of a submodular polyhedron on a two element ground set.

Figure 2.

result was the first polynomial time algorithm for submodular function minimization. Their algorithm for SFM uses the fact that the greedy algorithm (described in Figure 2) maximizes c^Tx over all vectors x contained in the *submodular polyhedron* P(f) defined below. In this definition and elsewhere, a vector $x \in \mathbf{R}^V$ determines a function on 2^V by $x(A) := \sum_{v \in A} x(v)$ for $A \subseteq V$.

$$P(f) := \{x \in \mathbf{R}^V | x(A) \leq f(A), \forall A \subseteq V\}.$$

(Note that P(f) is nonempty only if $f(\emptyset) \geq 0$. Since the function obtained by adding the same constant to every value of a submodular function is submodular, we will assume without loss of generality that $f(\emptyset) = 0$ for the remainder of this article.)

The greedy algorithm establishes that the optimization problem for submodular polyhedra is polynomial time solvable [8, 42]. Thus, Grötschel, Lovász, and Schrijver's work implies the separation problem for P(f) is polynomial time solvable. The separation problem is to determine if $x \in$ P(f) for a given x and if not return a set A for which $f(A) - x(A) < 0$. If $0 \in$ P($f - \mu$) for $(f-\mu)(A) := f(A) - \mu$, then the minimum value of f is at least μ. With simple bounds on a minimizer, obtainable using the greedy algorithm and another max-min relation of Edmonds given later in (3)[1] binary search can be used to find the minimizing set A in polynomial time.

The ellipsoid algorithm is not combinatorial. It relies heavily on linear algebra, and does not use the combinatorial structure of the problem except in a very abstract way. It provides a proof of the polynomial time solvability of SFM, but does not give any new understanding of how and why. This leaves a natural open problem to develop combinatorial algorithms for SFM. Work toward this goal led to demonstrating that some special subclasses are solvable directly: Cunningham gives a pseudopolynomial time algorithm for submodular function minimization that generalizes his strongly polynomial time[2] algorithm for the special case of testing membership of a vector $x \in \mathbf{R}^V$ in a matroid polyhedron [3, 5]. Queyranne describes a strongly polynomial time algorithm for minimizing symmetric[3] submodular functions [39], building on the overall minimum cut algorithm of Nagamochi and Ibaraki [38]. Both of the new combinatorial, polynomial-time algorithms for general submodular function minimization [31, 41] build on Cunningham's work. Despite this common starting point, these new algorithms are very different from each other. These combinatorial algorithms not only yield deeper understanding of submodular functions, but also, due to their combinatorial nature, lend themselves more easily to modification to special cases or extension to more general problems.

The next section contains a review of Cunningham's algorithm and a high level summary of the new algorithms. Figure 5 provides an explanation of some of the general ideas applied to the special case of the minimum $\{s, t\}$-cut problem. In Section 3, the new ideas and algorithms are described in more detail. Section 4 discusses some extensions of these combinatorial algorithms. Section 5 concludes with some open problems.

2. Combinatorial Algorithms

In this section, we describe the basic ingredients used in combinatorial algorithms for submodular function minimization, and explain the difficulties encountered in using them to obtain a polynomial time algorithm. Figure 5 describes some of these ingredients in the context of the minimum $\{s, t\}$-cut problem.

Cunningham describes the first combinatorial, pseudopolynomial time algorithm for submodular function minimization [5]. His algorithm is motivated by the min-max relation given in (3) below. This relation is implied by a theorem of Edmonds [8] that generalizes the matroid intersection theorem (2). First, a little notation. The *base polytope* of a submodular function, denoted B(f), is the face of the submodular polyhedron P(f) that satisfies $x(V) = f(V)$. Note that the greedy algorithm described in Figure 2 always returns an element in B(f). An element of the base polytope is called a *base* and extreme points of B(f) are *extreme bases*. For a vector $x \in \mathbf{R}^V$, let $x^-(v) := \min\{x(v), 0\}$. Edmonds theorem implies:

$$\max\{x^-(V) | \in B(f)\} = \min\{f(A) | A \subseteq V\} \quad (3)$$

This equality is a form of strong duality. The weak duality direction is easy to see: $x^-(V) \leq x(A) \leq f(A)$ for all $A \subseteq V$ since at worst, A contains all elements of V that have negative x-value, and does not contain any element with positive x-value.

[1]The greedy algorithm produces a vector $x \in$ P(f) satisfying $x(V) = f(V)$. By (3), $x^-(V)$ gives a lower bound on a minimum value of f. This does not exceed the maximum absolute value of f by more than a linear factor. A simple upper bound may be taken as $f(\emptyset)$, $f(V)$, or any other function value.
[2]A function is *pseudopolynomial* if for an input Q it is a polynomial in length(Q) and max(Q), where length(Q) is the number of bits needed to describe Q and max(Q) is the magnitude of the largest number in Q. This differs from a polynomial function which needs to be a polynomial in length(Q) and can only depend logarithmically on max(Q). A function is *strongly polynomial* if for an input Q, it is polynomial in the number of items in Q and does not depend at all on the size of the numbers in Q.
[3]A *symmetric* set function f satisfies $f(A) = f(V \setminus A)$ for all $A \subseteq V$.

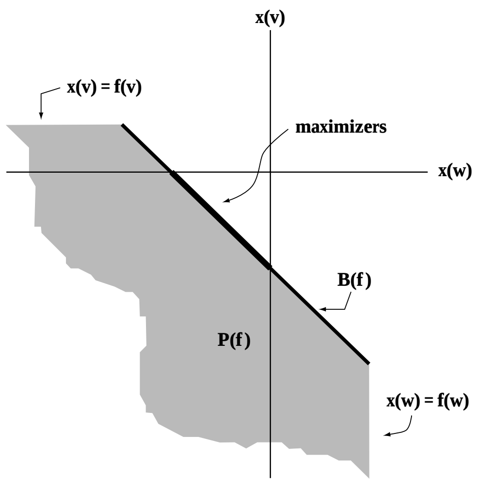


Figure 3. The submodular polyhedron and base polytope of a submodular function defined on a two-element ground set. The intersection of the base polytope with the negative orthant is the set of bases that maximize $x^-(\{v,w\})$.

Relation (3) can be used as the motivation for a combinatorial algorithm to find a minimizer of f: Starting with an arbitrary base $x \in \mathrm{B}(f)$ obtained using the greedy algorithm, try to "move towards" the base x^* that maximizes the left hand side of (3). An optimal base x^* can then be used to determine the minimizing set A. This is the idea underlying Cunningham's algorithm; we give details below.

The maximizer of the left hand side of (3) may not be an extreme point of $\mathrm{B}(f)$. An example of a base polytope with this property is given in Figure 3. This raises the following issue: Given the pair (x,A) as a candidate pair of optimal solutions for (3), how is it possible to verify their optimality, short of calling a subroutine for submodular function minimization? It is easy to check if $x^-(V) = f(A)$, but what proof is there that $x \in \mathrm{B}(f)$? To confirm that x is an extreme base, it is sufficient to provide a total order that generates x via the greedy algorithm. This can be done efficiently, e.g. [2]. For $x \in \mathrm{B}(f)$ not extreme, a proof of membership may be given by expressing x as a convex combination of affinely independent extreme bases y_i, $i \in I$: $x = \sum_{i\in I} \lambda_i y_i$. (Affine independence is used to ensure that I is not too large.) Cunningham introduced this idea first as a way to verify membership of x in a matroid polyhedron [3].

While it is possible to move from any base to any other base in $\mathrm{B}(f)$ by the addition of the appropriate vector $x \in \mathbf{R}^V$ with $x(V) = 0$, it is simpler to consider a restricted set of moves. The simplest move is to increase one component of x and decrease another component of x by the same amount. Mathematically, we can represent such a move from x using the incidence vector χ_v where $\chi_v(v) = 1$ and $\chi_v(w) = 0$ for all $w \neq v$. This move from x may be represented as $x' = x + \alpha(\chi_w - \chi_v)$. This move is called an *exchange operation* or simply an *exchange*.

Accessible Pairs

For an extreme base, it is possible to compute the exchange capacities of some pairs of elements efficiently. The following lemma illustrates this.

Lemma 2.2 *Suppose y is an extreme base of $B(f)$ generated by a total order $L = \{v_1, \cdots, v_{i-1}, v_i, v_{i+1}, \cdots, v_n\}$. Then*

$\alpha(y,v_i,v_{i+1}) = f(L(v_{i+1})\setminus \{v_i\}) - y(L(v_{i+1})\setminus \{v_i\})$

and $y' = y+\alpha(y,v_i,v_{i+1})(\chi_{v_{i+1}} - \chi_{v_i})$ is generated by $L' = \{v_1, \cdots, v_{i-1}, v_{i+1}, v_i, \cdots, v_n\}$.

There may be several different total orders that generate an extreme base y. For each of these orders, Lemma 2.2 applies. Bixby, Cunningham, and Topkis [2] describe an $O(n^2)$ algorithm to calculate the exchange capacities of all adjacent pairs (v_i,v_{i+1}) over all total orders generating extreme base y. It is this extended set of pairs that we call accessible pairs for y.

Figure 4.

To fully describe the set of allowable exchanges, it is necessary to specify how large α may be for a given triple $(x,v,w) \in \mathrm{B}(f) \times V \times V$. The constraint is that x' be a base, i.e. that it satisfies $f(A) - x'(A) \geq 0$ for all $A \subseteq V$. To ensure this, it is necessary to consider those sets A for which $x'(A) > x(A)$. These are precisely the sets that contain w and don't contain v. The amount of allowable exchange is determined by that set A which minimizes $f(A)-x(A)$. Thus the *exchange capacity* of (x,v,w), denoted $\alpha(x,v,w)$, is

$$\alpha(x,v,w) := \min\{f(A) - x(A) | w \in A \subseteq V\setminus\{v\}\}. \quad (4)$$

Exchange operations that increase a negative component of x and decrease a positive component of x improve $x^-(V)$. The following theorem, which is implicit in the proof of correctness of Cunningham's algorithm for SFM, implies it is possible to reach a maximizer by performing improving exchange operations only.

Theorem 2.1 *A base $x \in \mathrm{B}(f)$ maximizes $x^-(V)$ over all bases in $\mathrm{B}(f)$ if and only if for all elements v with $x(v) > 0$ and all elements w with $x(w) < 0$, the exchange capacity $\alpha(x,v,w) = 0$.*

Theorem 2.1 implies a deceptively simple algorithm for submodular function minimization: Define $S_x^+ := \{v|x(v) > 0\}$ and $S_x^- := \{v|x(v) < 0\}$. Next, find a maximizer of the left side of (3) by repeatedly finding a pair of elements in $S_x^+ \times S_x^-$ and performing the appropriate exchange operation. When the conditions of Theorem 2.1 are satisfied, then find a minimizer of f by taking the complement of the set of elements W that have strictly positive exchange capacity with an element in $\{w \in V\,|x^*(w) <0\}$. (It is not hard to show that $V\setminus W$ is x^*-tight. Thus $x^{*-}(V) = x^*(V\setminus W) = f(V\setminus W)$.)

The problem with this algorithm is that performing exchange operations relies on computing exchange capacities for x. Our expression for exchange capacity for $x \in \mathrm{B}(f)$ requires finding a minimizer of the function $f-x$ defined by $(f-x)(A) := f(A)-x(A)$. This is again submodular since the function x satisfies (1) at equality. In general, there is nothing special about the submodular function $f-x$ that would enable us to compute it faster than general submodular function minimization. In fact, even in the special case of finding a minimum s, t-cut, determining the exchange capacity requires a minimum cut computation. (See Figure 5.) Thus we have reduced our original problem to a problem that is just as hard.

Note that in the above discussion, we have ignored the representation of x as a convex combination of extreme bases, $x = \sum_{i\in I} \lambda_i y_i$, that we need to maintain as proof that $x \in \mathrm{B}(f)$. In order to maintain this, Cunningham restricts exchange operations to extreme bases y_i in this convex representation. Thus, performing an exchange operation of size $\alpha(y_i,v,w)$ on y_i changes $x(v)$ and $x(w)$ by only $\lambda_i\, \alpha(y_i,v,w)$.

The fact that computing exchange capacities in general is as difficult as SFM is an important obstacle that algorithms for submodular function minimization must overcome. In his pseudopolynomial time algorithm, Cunningham

An Example of Submodular Function Minimization: Minimum Cuts

Here, we demonstrate some of the concepts introduced in Section 2 by explaining what they mean for a familiar example of submodular function minimization: the minimum $\{s, t\}$-cut problem.

For the minimum $\{s, t\}$-cut problem, the submodular function we want to minimize is Δ_t^s. We start by normalizing to create $\tilde{\Delta}_t^s$ by subtracting $\Delta_t^s(\emptyset)$ from each value of Δ_t^s. Thus the minimum function value will be the value of the minimum cut minus $\Delta_t^s(\emptyset)$, which will be at most 0. A base of $B(\tilde{\Delta}_t^s)$ is determined orienting all edges incident to s away from s, all edges incident to t towards t, and assigning a fractional orientation to all other edges. A *fractional orientation* of an edge (v,w) is a replacement of the undirected edge (v,w) with two directed edges (v,w) and (w,v) along with nonnegative values u so that $u(v,w)+u(w,v)=1$. The corresponding base x is then defined setting $x(v)$ equal to the value of arcs leaving v minus the value of arcs entering v.

An extreme base of $B(\tilde{\Delta}_t^s)$ is obtained from order $(v_1, v_2, \cdots, v_n)$ by first orienting all edges incident to s away from s. The greedy algorithm then visits vertices in order, and iteratively orients all edges incident to v_i without previous orientation so that they leave v_i. (The greedy algorithm applied to order $(v_1, v_2, \cdots, v_n)$ sets $x(v_1) = \Delta_t^s(v_1)$ if there is no arc from s to v, and otherwise sets $x(v_1) = \Delta_t^s(v_1) - 2$.) In general, an orientation corresponds to an extreme base if and only if the resulting graph is acyclic: given an acyclic graph, any topological sort gives the order that generates the orientation and extreme base via the greedy algorithm.

An exchange operation for (x,v,w) corresponds to decreasing the value of arcs along a path or set of paths from v to w in the oriented graph G that generates x. This is done so that the net value of arcs leaving v decreases, net value of arcs leaving w increases, and net value of arcs leaving all other nodes stays the same. (*Net value* leaving v is total value of arcs leaving v minus total value of arcs entering v.) This could be interpreted as sending flow through the oriented graph G from v to w respecting upper bounds on capacities determined by u, and letting the new G be the residual graph of this flow. The exchange capacity of (v,w) is twice the value of the maximum flow from v to w in the oriented graph G with values u interpreted as upper bounds on capacities. (Twice the value because reversing the direction of (v,w) increases $x(w)$ by two, and decreases $x(v)$ by two.) So, even in this special case, the problem of computing exact exchange capacities is again a problem of the same form of the original: here, a minimum cut problem. However, finding a lower bound on the exchange capacity, using a simple augmenting path, is a much easier computation. In addition, if x is an extreme base, and v_i and v_{i+1} are adjacent in the order generating x, then $\alpha(x,v_i,v_{i+1})$ is either 0 if there is no edge between v_i and v_{i+1}, or 2 if there is such an edge.

The alternative characterization of a maximizer given in Theorem 2.1 for the minimum $\{s, t\}$-cut problem is the assertion that an optimal orientation does not contain a path from any v with positive value to any w with negative value. This corresponds to the well-known alternative characterization of a maximum flow. The union of s with the set of nodes reachable from S_x^+ in the final oriented network defines a minimum cut.

Figure 5.

[5] uses an idea first proposed in Bixby, Cunningham, and Topkis [2] to restrict exchange operations to a subset of directed pairs for which the exchange capacities can be computed efficiently. For $x = \sum_{i \in I} \lambda_i y_i$ with y_i an extreme base for all i, each y_i determines a set of special directed pairs with easily computed exchange capacities. (See Figure 4 for more explanation.) Call the union of these directed pairs over all $i \in I$ *accessible pairs*. Bixby et al. [2] define a graph with vertex set equal to V and a directed arc for each accessible pair (called an *accessible arc*). (For $\alpha(y_i,v,w)>0$ with (y_i,v,w) accessible, the arc is directed from v to w.) Cunningham proves that it is always possible to reach the optimal x^* by repeating the following operation a pseudopolynomial number of times: Find a path in the union of accessible arcs from S_x^+ to S_x^- and perform exchange operations for each arc along the path so that the x-value of an element in S_x^+ decreases, the x-value of an element in S_x^- increases, and all other x-values remain the same. Such a path is called an *augmenting path*.

This algorithm looks very similar to strongly polynomial augmenting path algorithms for the maximum flow problem [15], and to Cunningham's strongly polynomial algorithm for testing membership in matroid polyhedra [3]. However, it is not known to be even weakly polynomial for general submodular function minimization. For maximum flow, Edmonds and Karp [10] propose two heuristics used to select paths in a way that will lead to a polynomial time algorithm: the shortest path heuristic and the fattest path heuristic. Why don't these heuristics work here?

The shortest path heuristic for maximum flow selects the path with the least number of arcs. The proof of polynomiality of this heuristic follows from showing that the length of the shortest path never decreases, and after a polynomial number of iterations, provably increases. This heuristic does not work for Cunningham's algorithm because the length of the shortest path may actually decrease: The set of accessible pairs considered in finding the augmenting path is not the complete set of pairs with positive exchange capacity. After each exchange operation, the exchange capacities and the set of accessible pairs change.

The fat path heuristic selects the residual path with largest capacity. This works for maximum flow because the path with the largest residual capacity must carry at least a polynomial fraction of the difference between the maximum flow value and the current flow value, thus guaranteeing sufficiently large progress with each augmentation. For Cunningham's algorithm, there may be no residual path with large capacity [5]. The problems are two-fold: First, even if $f(A) - x(A)$ is large, there may not be an accessible arc with large exchange capacity leaving A. Second, the amount of exchange that is affected in x by performing exchanges on y_i for $i \in I$ depends not only on the exchange capacity, but also on the multiplier λ_i in the convex representation, which may be exponentially small. (It could depend inversely on a polynomial in $\max_{A \subseteq V} |f(A)|$.)

For network flow problems, since fat paths exist, an effective way to ensure large augmentations is to use scaling. However, it is not clear how to scale submodular functions. The perhaps natural idea of using $\left\lfloor \frac{f}{\delta} \right\rfloor$ defined by $\left\lfloor \frac{f}{\delta} \right\rfloor(A) := \left\lfloor \frac{f(A)}{\delta} \right\rfloor$ does not work because it is not submodular, and hence the structure of submodular polyhedra is lost. For example, if the function f is not submodular, then the greedy algorithm no longer optimizes over $P(f)$. (An easy way to see $\left\lfloor \frac{f}{\delta} \right\rfloor$ is not submodular is to consider the case of $\delta = 2$, $|A|$ and $|B|$ odd for $A,B \subseteq V$, $|A \cap B|$ and $|A \cup B|$ even, and $f(A) + f(B) = f(A \cap B) + f(A \cup B)$.)

New Approaches to SFM

The two new algorithms for submodular function minimization take two different approaches to extend Cunningham's algorithm. Iwata, Fleischer, and Fujishige [31] devise a scaling scheme and special augmenting paths to employ a "fat path" approach to SFM: They show that each of their augmentations is bounded by a polynomial in the difference between the current $x^-(V)$ value and optimal $x^{*-}(V)$ value. Schrijver [41] designs a different graph and augmentation step to employ a "short path" approach to SFM: he shows that the shortest path distances from S_x^+ in his graph are nondecreasing, and that at least one distance increases after a polynomial number of augmentations. Both of these approaches are described further in the next section.

3. New Combinatorial, Polynomial Time Algorithms

The two new polynomial time algorithms for submodular function minimization propose alternative solutions to the difficulty of establishing a polynomial time, augmenting path algorithm. Iwata, Fleischer, and Fujishige [31] use a scaling framework to relax the value of the submodular function by an additive function depending on the scaling parameter. This relaxation has the effect of relaxing exchange capacities by the same scaling parameter, thus allowing sufficiently large augmentations demonstrated in a "fat path" analysis. Schrijver [41] instead uses a "short path" analysis by allowing the use of all exchange operations for each y_i in the convex representation of the current base x. Thus he works with a graph that contains an *exchange arc* for every directed pair (v,w) with positive exchange capacity for some y_i, $i \in I$. To overcome the difficulty of computing exact exchange capacities, he shows how it is possible to compute lower bounds on exchange capacities that are sufficient for the algorithm to make demonstrable progress. In addition, he abandons the direct use of augmenting paths, instead focusing on removing selected exchange arcs in a way that ensures shortest path lengths are nondecreasing. His approach uses a layered network (á la Dinic [7]). Recently, Fleischer and Iwata [12] have shown that Schrijver's idea of removing selected arcs by successive exchange operations can be embedded in a push-relabel framework, thus yielding a faster and more adaptable algorithm. The push-relabel algorithm of Goldberg and Tarjan [24] provides a good analogy in maximum flows for the technique of removing selected arcs from the residual graph to ensure that shortest path lengths are nondecreasing. A more detailed description of both approaches follows.

3.1. A Scaling Algorithm

In this section, we describe the scaling algorithm for SFM introduced in [31]. Most of the new ideas used to obtain the polynomial time scaling algorithm for submodular function minimization appeared first in recent algorithms for submodular flow. To explain this connection, we start by defining submodular flow and describing some relations between submodular flow and submodular function minimization.

Submodular Flow

Like standard network flows, the submodular flow problem is defined on a graph $G=(V,E)$ with vertex set V and arc set E. A *flow* φ is a function defined on the arcs that obeys capacity constraints $\ell(v,w) \le \varphi(v,w) \le u(v,w)$ for all $(v,w) \in E$. Given a vector $b \in \mathbf{R}^V$ of supplies and demands satisfying $b(V) = 0$, a standard *feasible flow* is a flow φ that satisfies flow conservation constraints that say the net flow leaving v, written $\sum_w [\varphi(v,w) - \varphi(w,v)]$, equals $b(v)$ for each vertex $v \in V$.

A *feasible submodular flow* for submodular function f relaxes the flow conservation constraints in standard network flows and replaces them by inequalities that require that the net flow leaving any subset is at most the submodular function value of the subset. This can be described as follows. Define the *boundary* of the flow at A, denoted $\partial\varphi(A)$, as the net flow leaving A. Mathematically, $\partial\varphi(A) := \sum_{v\in A, w\notin A} [\varphi(v,w) - \varphi(w,v)]$. The *submodular constraints* can then be written as $\partial\varphi(A) \le f(A)$ for all $A \subseteq V$. It is not hard to show that $\partial\varphi(A) = \sum_{v\in A} \partial\varphi(v)$. Since $\partial\varphi(V) = 0$, we assume that $f(V) = 0$, and thus this is equivalent to requiring $\partial\varphi \in \mathrm{B}(f)$. The *submodular flow problem* asks for a feasible submodular flow of minimum cost, where cost is the dot product of c: $E \to \mathbf{R}$ with φ. Written as an exponential size linear program:

$$\textbf{(SF)}\quad \begin{array}{l} \min c^T\varphi \\ \ell \le \varphi \le u \\ \partial\varphi \in \mathrm{B}(f) \end{array} \qquad (5)$$

The minimum cost flow problem is a special case of submodular flow with f replaced by the set function defined by supply vector b. (Since $\partial\varphi(v) \le b(v)$ and $-\partial\varphi(v) = \partial\varphi(V \setminus v) \le b(V \setminus v) = -b(v)$, feasible φ satisfies flow conservation constraints.)

The submodular flow problem was introduced by Edmonds and Giles [9] as a framework that includes as special cases minimum cost flows, submodular function minimization, (poly)matroid intersection, and the problem of contracting a minimum cost subset of arcs to make a directed graph strongly connected. It also models other network design problems, such as finding the least cost subgraph containing k vertex disjoint paths in a graph from a root r to all other vertices $v \in V$ [17, 18]. Edmonds and Giles show that the set of extreme points of the submodular flow polytope are integral [9]. The submodular flow problem can be solved in polynomial time using the ellipsoid algorithm [27], and via combinatorial algorithms that depend on an oracle for submodular function minimization, e.g. [40, 34, 16, 6, 44, 22, 21, 14]. Thus, another application of SFM is in solving submodular flow problems. Why do all of these algorithms for submodular flow require an oracle for SFM?

With maximum flows, to move from one feasible flow to another, it suffices to send flow around a cycle, or set of cycles, in the residual graph of the flow. This maintains flow conservation at all nodes. With submodular flows, it is permissible to change the net flow in or out of any vertex, as long as the submodular constraints are satisfied. Thus, moving from one feasible flow to another may involve also sending flow along paths. Suppose one sends flow from s to t in the residual graph of φ. How much flow can be sent so that the new flow φ' is feasible? Sending flow along an s-to-t path increases the net flow leaving s and decreases the net flow leaving t. This increases the net flow leaving any set A containing s but not t. To stay feasible, the amount of flow sent along the path must be bounded by $f(A) - \partial\varphi(A)$ over all sets A that contain s but not t. This is precisely the exchange capacity $\alpha(\partial\varphi,t,s)$. The combinatorial algorithms for submodular flow add exchange arcs for every pair (t, s) with positive exchange capacity to the set of flow arcs, assign each arc a capacity equal to the corresponding exchange capacity, and solve the submodular flow problem in the resulting graph.

New Ideas for Submodular Functions and Flows

While submodular flow algorithms assume an oracle for SFM to compute exchange capacities, a capacity (or function) scaling algorithm for submodular flow would presumably deal with some of the problems Cunningham faced in trying to obtain a polynomial time algorithm for SFM. In particular, a capacity or function scaling algorithm would have to address the problem of finding augmenting paths with sufficiently large capacity. However, the existence of a polynomial time, function-scaling algorithm for submodular flow remained an open problem until only recently. We highlight below the two main ideas that led to the new scaling algorithm for SFM.

Approximate Optimality. In the first function-scaling algorithm for submodular flow, to adjust for the fact that $\left\lfloor \frac{f}{\delta} \right\rfloor$ is not submodular, Iwata [30] proposes using f^{δ} defined by $f^{\delta}(A) := \delta\lfloor f(A)/\delta \rfloor + \delta\,|A|\,|V-A|$. This extra term $\delta|A||V\text{-}A|$ is introduced to ensure submodularity. It is the cut function of the complete directed graph of capacity δ, and hence itself submodular. The submodular function f^{δ}, relying as it does on $\lfloor f(A)/\delta \rfloor$, which is not submodular, is difficult to deal with, and the resulting submodular flow algorithm has a high complexity [30].

A slight modification of this idea is to use the submodular function f_{δ} defined by $f_{\delta}(A) = f(A) + \delta|A||V\text{–}A|$. This function has the advantage that it is the sum of two submodular functions, which makes it easier to work with, as detailed below. This function is first used by Iwata, McCormick, and Shigeno [32] in a faster function-scaling algorithm for submodular flow.

The relaxation f_{δ} has a natural interpretation in the setting of network flows. As a counterpart to Goldberg and Tarjan's approximate optimality for conditions for minimum cost flows [25], Ervolina and McCormick [11] introduce a dual notion of approximate optimality for minimum cost flows. They define a relaxation that relaxes the capacity of each flow arc by the scaling parameter δ. If one includes also 0-capacity arcs, this is tantamount to adding a complete directed graph on V of capacity δ. For submodular networks, the "graph" is the set of possible exchange arcs, which is really the complete directed graph on V. Thus, using f_{δ}, interpreted as relaxing all exchange capacities by δ, is like solving the original problem with the added complete directed graph of capacity δ. This leads to an approximate optimality for submodular function minimization. Call this added graph $H(\delta)$, and the arcs in this added graph by *relaxation arcs*.

The algorithms that use this relaxed submodular function maintain a base $z \in B(f_{\delta})$ and seek to maximize $z^{-}(V)$ instead of $x^{-}(V)$ during a δ-scaling phase. The base z is represented as a base $x \in B(f)$ and a base $\partial\psi \in B(\Delta_{H(\delta)})$, where $\Delta_{H(\delta)}$ is the directed cut function of $H(\delta)$ defined on a subset A by the capacity of arcs leaving A minus arcs entering A. This representation of a base in $B(f_{\delta})$ explicitly uses the fact that f_{δ} is the sum of two submodular functions. The latter term $\partial\psi$ is equivalently expressed as the boundary of a flow ψ in the complete directed graph of capacity δ. A bound on $z^{-}(V)$ yields an approximate bound on $x^{-}(V)$: $x^{-}(V) \geq z^{-}(V) - \delta n^2/2$.

Fat Paths for SFM. How should $H(\delta)$ be used to obtain a polynomial time algorithm? With $H(\delta)$, it is now possible to augment on paths consisting of exchange arcs and relaxation arcs. However, once all paths of relaxation arcs are saturated, the same problem Cunningham faced still remains: there may be no fat path among the easily accessible exchange arcs.

The answer is suggested in a third function-scaling algorithm for submodular flow by Fleischer, Iwata, and McCormick [14]. This is an augmenting path algorithm that augments only along paths of relaxation arcs (and original flow arcs for the submodular flow problem). It avoids exchange arcs on an augmenting path by trading exchange capacity on an exchange arc for flow capacity on a parallel relaxation arc. Whenever an exchange arc (y_i,s,t) is encountered in the search for an augmenting path from v with $z(v) \geq \delta$ to w with $z(w) \leq -\delta$, an exchange for (y_i,s,t) is performed of value $\alpha \leq \delta$, and the flow on relaxation arc (s, t) is reduced by $\lambda_i\alpha$. This enforces that $z = x - \partial\psi$ remains unchanged. The flow reduction is an exchange operation for the triple $(\partial\psi,s,t) \in B(\Delta_{H(\delta)}) \times V \times V$. Thus this trading of exchange capacity for flow capacity is called a *double-exchange*. It can also be viewed as sending "flow" around a cycle consisting of an exchange arc, and a residual relaxation arc. A double-exchange removes an exchange arc whose behavior is hard to predict (in terms of being able to compute its capacity in the future), and replaces it with a parallel relaxation arc with a known, fixed capacity.

The idea for the double-exchange routine designed in [14] has roots in a distinctly different subroutine used in the submodular flow algorithm of Iwata, McCormick, Shigeno [32]. In both [14] and the SFM algorithm [31], a double-exchange is performed for individual arcs, and for $\alpha = \min\{\delta,\alpha(y_i,s,t)\}$, where δ is the scaling parameter. For SFM, an augmenting path of relaxation arcs with capacity δ is found after at most n^3 double-exchanges.

A δ-scaling phase ends when there are no augmenting paths of capacity δ. At this point, the set A of elements that can reach $\{v|\ z(v) \leq -\delta\}$ along paths of capacity at least δ satisfies $x^{-}(V) > f(A)\text{–}n^2\delta$ [31].[4] Thus, at the end of the $\delta = 1/n^2$ scaling phase, the algorithm finds a set A and a vector x with $x^{-}(V) > f(A)\text{–}1$. If f is integer, relation (3) implies that A is a minimizer.

The above ideas lead to a straightforward scaling algorithm for submodular function minimization that has a weakly polynomial run time, in that the run time depends on log ($\max_{A\subseteq V} |f(A)|$). It is possible to turn this into a strongly polynomial algorithm replacing this log term with an $n^2 \log n$ term. The main idea is to show that it is possible, after log n scaling phases to do one of the following two things:

i. Identify a new element that is contained in every minimizer of f.
ii. Identify a new pair of elements (v,w) such that w is contained in every minimizer containing v.

The strongly polynomial time algorithm maintains pairs satisfying (ii) as directed arcs in a directed acyclic graph. Whenever a cycle forms in the graph, the elements in the cycle may be contracted into a single element, since either they are all contained in a minimizer, or none of

[4]In [31], the graph used has the direction of each accessible arc reversed. This choice of direction is arbitrary but affects some details in the description of the algorithm. For example, with this alternate choice, augmentations are from v with $z(v) \leq -\delta$ to w with $z(w) \geq \delta$, and the corresponding statement about the set A applies in their setting to the set of elements reachable from $\{v|z(v) \leq -\delta\}$.

them are. Since this directed graph can have at most n^2 new arcs, after at most $n^2 \log n$ phases, a minimizer of f is found. The basic concept of fixing a new arc after $\log n$ scaling phases was first used by Tardos [43] in the design of the first strongly polynomial algorithm for minimum cost flow.

3.2. Schrijver's Approach

Schrijver [41] takes a completely different approach to confronting the difficulty of computing general exchange capacities. Instead of maneuvering with only a subset of the exchange arcs, Schrijver considers the full set of exchange arcs for each y_i, $i \in I$. Given a pair (s, t) with positive exchange capacity for $x = \sum_{i\in I} \lambda_i y_i$, Schrijver shows how it is possible to compute a lower bound $\eta \le \alpha(x,s,t)$ so that after performing the exchange $x' = x + \eta(\chi_t - \chi_s)$, the pair (s, t) does not have positive exchange capacity for any extreme base y'_i, $i \in I$, in the convex representation of x'. In other words, (s, t) is no longer in the union of exchange arcs for y'_i, $i \in I$. Applied to a pair (s, t) for which there originally was such an arc, this has the effect of increasing the distance between s and t in the graph of the union of exchange arcs of y'_i, $i \in I$. Schrijver shows that by applying this to specially chosen pairs (s, t), the distance of vertices from the set S^+_x never decreases, and after a polynomial number of iterations, the distance label of some vertex increases.

A Useful Subroutine. The key to Schrijver's algorithm is a subroutine that after at most n^2 calls, effectively removes an arc from the union of exchange arcs of extreme bases. To describe this subroutine, we will need a little notation. Let the relation $s \preceq_y t$ indicate that $\alpha(y,s,t) > 0$. (Note that $\preceq_y$ is transitive, since if $\alpha(y,s,t) = 0$, then there is a set A satisfying $s \in A \subseteq V\backslash\{t\}$ and $y(A) = f(A)$. For any $w \in V$, either $w \in A$ in which case $\alpha(y,w,t) = 0$, or $w \notin A$ in which case $\alpha(y,s,w) = 0$.) It is possible to compute the relation $\preceq_y$ for extreme base y efficiently, e.g. [2]. Define $[s,t]_y := \{v \in V | s \preceq_y v \preceq_y t\}$.

Given a pair (s, t) with positive exchange capacity for some extreme base in the convex representation of x, let y be the extreme base in this convex representation with maximum $|[s,t]_y|$, and let λ be its multiplier in the representation. Schrijver devises a subroutine that computes a lower bound $\mu \le \alpha(y,s,t)$, and extreme bases z_j, $j \in J$ with multipliers γ_j such that

1. $y + \mu(\chi_t - \chi_s) = \sum_{j\in J} \gamma_j z_j$,
2. $\sum_{j\in J} \gamma_j = 1$,
3. $|[s,t]_{z_j}| < |[s,t]_y|$ for all $j \in J$.

Let y' be the extreme base obtained by performing the full exchange operation: $y' = y + \alpha(y,s,t)(\chi_t - \chi_s)$. Let $x' = x + \lambda\mu(\chi_t - \chi_s)$ be the new base obtained by performing the lower bound exchange. The first two properties allow us to replace $\lambda(1-\mu)y + \lambda\mu\, y'$ in the convex representation of x' with $\lambda\sum_{j\in J} \gamma_j z_j$. The third property ensures that this substitution makes some progress: After one iteration either $\max_{i\in I} |[s,t]_{y_i}|$ decreases, or the number of bases in I achieving this maximum decreases. Call this subroutine Reduce-Interval(s, t).

If Gaussian elimination is used after each call to Reduce-Interval to reduce the number of bases in the convex representation of x', then $|I|$ remains at most n. Since there are at most n possible values of $|[s,t]_y|$, this implies that Reduce-Interval(s, t) is called at most n^2 times before $\alpha(y_i,s,t) = 0$ for all $i \in I$.

Short Paths for SFM. Schrijver finds a lexicographically shortest augmenting path from S^+_x to S^-_x in the union of exchange capacity arcs for y_i, $i \in I$ and applies Reduce-Interval to the last arc (s, t) in this path until either $x(t) = 0$ or $\alpha(y_i,s,t) = 0$, $\forall\, i \in I$. He shows that by such application, the distances of vertices from S^+_x never decrease, and that after a polynomial number of steps, the distance of some vertex increases. This algorithmic framework bears resemblance to the layered network approach for submodular flows described by Tardos, Tovey, and Trick [44], which in turn combines ideas from earlier augmenting path algorithms for feasible submodular flow [40, 34, 16] with the maximum flow framework of Dinic [7].

Instead of detailing the particulars of this approach, we describe a modification that is both faster and cleaner. To avoid computing augmenting paths in each iteration, the push-relabel framework, introduced for maximum flows by Goldberg and Tarjan [24], maintains distance labels in a lazy manner and, like Schrijver's algorithm, performs operations on only one arc at a time. Goldberg and Tarjan show that this improves on Dinic's maximum flow algorithm by a factor of n. This was adapted to the feasible submodular flow problem by Fujishige and Zhang [22] to obtain a similar improvement in run time. It can be adapted to SFM as well [12] and thus the overhead of computing shortest augmenting paths is avoided. We outline the main ideas of this push-relabel algorithm for SFM.

For maximum flows, the push-relabel algorithm maintains a flow φ that satisfies capacity constraints and a labeling d of the vertices that corresponds to a lower bound on the distance of a vertex from the sink. A valid label d satisfies $d(\text{source}) = n$, $d(\text{sink}) = 0$, and $d(s) \le d(t) + 1$ for any arc (s, t) with $\varphi(s,t)$ less than the capacity of (s, t). (In this case, arc (s, t) has *residual capacity* equal to the capacity of (s, t) minus $\varphi(s,t)$.) The algorithm relies on two operations: push and relabel. A push operation applies to (s, t) if the net flow leaving s (flow in minus flow out) is less than 0, arc (s, t) has residual capacity, and $d(s) = d(t) + 1$. Call *excess* at s the net flow entering s, and denote this by $e(s)$. Push applied to (s, t) increases flow on (s, t) by the minimum of $e(s)$ and the residual capacity of (s, t). A relabel operation applies to s if $e(s) > 0$, and no push operation applies to s. Relabel applied to s increases $d(s)$ by one. Goldberg and Tarjan [24] show that the push and relabel operations maintain valid labels. In addition, if a push is applied to the vertex with excess and the highest label, then they show that the algorithm terminates after at most n^3 pushes and n^2 relabels.

For submodular function minimization, a valid label d satisfies $d(t) = 0$ if $x(t) < 0$, and $d(s) \le d(t) + 1$ if $s \preceq_{y_i} t$ for some $i \in I$. The operation Push(s, t) repeatedly calls Reduce-Interval(s, t) until either $x(s) = 0$ or $\alpha(y_i,s,t) = 0$, $\forall i \in I$. Push(s, t) applies if $x(s) > 0$, $s \preceq_{y_i} t$ for some $i \in I$ and $d(s) = d(t) + 1$. In effect, Push is applied to the first arc of an augmenting path, instead of the last. (Schrijver's algorithm may be modified in this way also, yielding the same run time as his version that works on the last arc.) A relabel operation applies if $x(s) > 0$ and no Push operation applies to s. Relabel(s) increases $d(s)$ by one. The main complication in extending the push-relabel framework to submodular function minimization is to show that Push(s, t) maintains valid labels. All the other analyses in [24] then follows easily.

The problem that arises is that after applying an exchange operation to (s, t) to move to $y' = y + \alpha(\chi_t - \chi_s)$, a pair (v,w) that initially had 0 exchange capacity may now have positive capacity. This can happen if there is a y-tight set A that includes w and s and excludes v and t. After the exchange, this set is no longer tight, since $y'(A) = y(A) - \alpha < y(A) = f(A)$. If this was the only tight set separating v and w, then (v,w) has positive exchange capacity for y'. The following lemma, which appears in Schönsleben [40] and Lawler and Martel [34], describes under what circumstances this can happen.

Lemma 3.1 *If* $\alpha(y,v,w) = 0$ *and* $\alpha(y',v,w) > 0$ *where* $y' = y + \alpha(\chi_t - \chi_s)$ *for* $y', y \in \mathrm{B}(f)$, *then* $\alpha(y,s,w) > 0$ *and* $\alpha(y,v,t) > 0$.

One consequence of this lemma is that Push(s, t) maintains valid labels: if (v,w) satisfies $v \npreceq_{y_i} w$ for all $i \in I$ before Push(s, t) and $v \preceq_{y_i} w$ for some $j \in I$ after Push(s, t), then by the validity of labels before Push and Lemma 3.1 we have $d(v) \leq d(t) + 1 = d(s) \leq d(w) + 1$.

The push-relabel algorithm terminates with the optimal x^* when there are no augmenting paths from S_x^+ to S_x^-. By the validity of the distance label d, this occurs the first time there is a distance value p such that no vertex has this label, and all elements in S_x^+ have higher labels. The set of elements with label lower than p is a minimizer of f. By applying push or relabel always to the highest labeled vertex in S_x^+, the number of pushes and relabels that can occur before this condition is met is bounded in the same way as in the push-relabel algorithm for maximum flow.

4. Some Consequences of Combinatorial, Polynomial Time Algorithms for SFM

We now describe some examples that demonstrate that these new combinatorial algorithms lend themselves more easily to modification or generalization to other problems. Both sets of examples take advantage of the existing algorithmic research literature in network flows and submodular flows.

4.1. Advantages of a Push-Relabel Algorithm for SFM

The first extensions we describe use the push-relabel algorithm for submodular function minimization. This algorithm can be extended to efficiently solve a parametric submodular function minimization problem. This, in turn, leads to a combinatorial algorithm to compute a lexicographically optimal base [13].

The push-relabel algorithm for maximum flow [24] has been used in the design of efficient algorithms for many more general problems. Two prominent examples include the parametric flow problem and the overall minimum cut problem in a directed graph. In the first instance, Gallo, Grigoriadis, and Tarjan extend push-relabel to solve the parametric network flow problem [23]. This is defined on a network with arc capacities that are functions of a parameter θ: increasing in θ leaving the source, decreasing in θ entering the sink, and constant elsewhere. The *parametric network flow problem* is to compute a maximum flow for each of an increasing sequence of parameters $\theta_1 < \theta_2 < \cdots < \theta_n$ in the same asymptotic time as one push-relabel maximum flow computation. This has applications in scheduling [36]. In the second instance, Hao and Orlin [28] extend the push-relabel algorithm to compute an overall minimum cut in a directed graph in the same asymptotic time as one run of the push-relabel algorithm.

Both of these ideas can be generalized to submodular function minimization, with some care. In the first instance, let $f' \leftarrow f$ denote that for all sets A, B with $A \subset B$

$$f(B) - f(A) \geq f'(B) - f'(A).$$

The relation $f' \leftarrow f$ is called a *strong map*. Both the parametric flow problem considered in [23], and the problem of finding minimizers for a sequence of parameterized submodular functions $f + \theta w$ for a nonnegative vector $w \in \mathbf{R}^V$ and an increasing sequence of parameter values of θ, are examples of strong map sequences. Iwata, Murota, and Shigeno [33] give a generalization of the algorithm in [23] to polymatroid intersection for a strong map sequence. Extending the new push-relabel algorithm for submodular function minimization, it is possible find the minimizer of a sequence of submodular functions $f_1 \leftarrow f_2 \leftarrow \cdots \leftarrow f_n$ in the same asymptotic time as the push-relabel algorithm for SFM [13]. In turn, an efficient algorithm for strong map submodular function minimization implies a more efficient algorithm to find a lexicographic optimal base, a generalization of Megiddo's lexicographically optimal flow [37] introduced by Fujishige [19].

4.2. Improved Algorithms for Submodular Flow

In Section 3.1, we saw that ideas developed for submodular flow led to recent progress in algorithms for submodular function minimization. In this section we explain how these new algorithms for SFM lead to more direct algorithms for solving submodular flow. In particular, we briefly outline the first algorithms for submodular flow that do not rely on an oracle for computing exchange capacities. We start by highlighting similarities between network flows, SFM, and submodular flow.

Algorithmic Relation to Network Flows and SFM. Generic augmenting path algorithms for minimum cost flow with supply and demand vector b start with a flow φ and augment along paths of flow arcs with residual capacity from $\{v | b(v) > \partial\varphi(v)\}$ to $\{v | b(v) < \partial\varphi(v)\}$. The underlying idea behind the augmenting path algorithms we have described for submodular function minimization start with a base $x \in \mathrm{B}(f)$ and augment along paths of exchange capacity arcs from $\{v | x(v) > 0\}$ to $\{v | x(v) < 0\}$. A generic submodular flow algorithm may start with a flow φ and a base $x \in \mathrm{B}(f)$. By augmenting along paths of flow and exchange arcs from $\{v | \partial\varphi(v) < x(v)\}$ to $\{v | \partial\varphi(v) > x(v)\}$, it is desired to obtain base $x^* \in \mathrm{B}(f)$ and flow φ^* satisfying $\partial\varphi^* = x^*$.

Submodular Flow without an Exchange Oracle. The hard part of submodular flow is dealing with the exchange capacities efficiently. This is the problem solved by combinatorial, polynomial-time algorithms for submodular function minimization. Thus, by adding to the arc set in these SFM algorithms an additional set of arcs that correspond to the flow arcs of a submodular flow problem, it is possible to modify the SFM algorithms to solve the feasible submodular flow problem. Capacities of flow arcs are easy to compute, and thus they do not add extra complexity to these algorithms. This is the idea behind the feasible submodular flow algorithm detailed in [12]. It modifies the above mentioned push-relabel algorithm for SFM by adding flow arcs to the set of exchange arcs for y_i, $i \in I$, and generalizes the Push operation to allow sending flow on these flow arcs as well, thus solving the feasible submodular flow problem within the same asymptotic time bound as the push-relabel algorithm for SFM.

Given this result, it is perhaps natural to ask if feasible submodular flow equivalent to submodular function minimization. Checking if a submodular flow is feasible is a submodular function minimization problem. But it is not clear what the answer is if one asks to find a feasible flow. Even for the special case of minimum cuts, it is still not known if the exact (s, t)-minimum cut problem is any easier than maximum flow.

Minimum Cost Submodular Flow. While there seems to be a natural relation between feasible submodular flow and submodular function minimization, the submodular flow problem with costs would appear to be a more difficult problem. However, it is possible to extend the combinatorial algorithm for SFM in [31] to solve the minimum cost submodular flow problem [12]. One reason for the success of this extension is due to the fact that the SFM algorithm in [31] is based on ideas arising in algorithms for submodular flow, and in fact is highly similar to the submodular flow algorithm in [14]. Unlike the case for extending SFM algorithms to solve feasible submodular flow, it is not sufficient to simply add flow arcs to the set of arcs used in [31]. The problem is that the algorithm described in [31] is not particular in its choice of augmenting path. In augmenting path algorithms for standard minimum cost flows, it is necessary to select the *least cost* augmenting path. This is also necessary in submodular flows. Thus it must be shown that the augmenting path subroutine in [31] can be extended to find a least cost augmenting path. This is proven in [12]. The resulting algorithm finds an optimal dual solution, i.e. optimal node prices, for the submodular flow problem. The optimal flow can then be found with $n - 1$ additional submodular function minimizations.

5. Some Additional Questions

1. An example of a submodular function minimization problem that we do not know how to solve in polynomial time without recourse to a general submodular function minimization algorithm is checking the feasibility of transshipment problem over time [29]. The transshipment problem over time is flow problem with multiple sources (vertices with positive supply) and multiple sinks (vertices with negative supply; also called positive demand). Each arc (u,v) in G has a capacity, and a nonnegative transit time $\tau_{u,v}$ that determines how long it takes for flow leaving u to arrive at v. Thus, flow leaving u at time t arrives at v at time $t + \tau_{uv}$. Given a time bound T, the feasibility problem is to determine if there exists a flow that completes by time T, respects capacity constraints, and meets all demand. This can be solved by submodular function minimization. Is there a more efficient approach to this problem that uses a more specialized algorithm than general SFM? More broadly, do the algorithms in this paper yield new and interesting algorithms for special cases of SFM?
2. Despite the analogies drawn in this article, submodular flow algorithms are not simple extensions of minimum cost flow algorithms. While all of the existing submodular flow algorithms build on ideas used previously in network flows, it is often nontrivial to extend these network flow algorithms to submodular flows. For example, Goldberg and Tarjan devised a cost-scaling algorithm for minimum cost flow using the push-relabel framework [25], but there is still no cost scaling algorithm using push-relabel techniques for minimum cost submodular flow. Hence we do not know if the push-relabel algorithm for SFM can be extended to solve general submodular flow problems.
3. Submodular flow is a general problem class for which all extreme solutions are integral. It is not the most general such class. Edmonds and Giles [9] describe a framework that includes a more general class of problems with integral polytopes: TDI systems. Can these algorithms be extended to yield combinatorial algorithms for TDI systems, or other general problem classes?
4. There are other linear programs involving submodular functions for which the resulting polytopes are not integral. An interesting class model network design problems. A generic network design problem asks to add a minimum cost set of edges to an existing graph so that certain connectivity properties are met. Special cases include the Steiner tree problem and the minimum cost k-connected subgraph problem. If f is a set function describing the connectivity requirements between each subset and its complement, then the network design for a graph $G = (V,E)$ and cost vector $c \in \mathbf{R}^E$, can be modeled as an integer program with an exponential number of constraints variable vector $x \in \{0,1\}^{|E|}$ to indicate whether or not an edge is included in the minimum cost solution: min $c^T x$ subject to $\forall\ A \subseteq V$, $x(\Delta(A)) \geq f(A)$ and $x \in \{0,1\}^{|E|}$. Frequently f is supermodular (meaning that $-f$ is submodular), or weakly-supermodular (i.e. f satisfies $f(A) + f(B) \leq \max\{f(A \cup B) + f(A \cap B), f(A \backslash B) + f(B \backslash A)\}$). Some approximation algorithms to solve these often NP-hard problems round the optimal solution to the linear program relaxation of this formulation. Is there a combinatorial, polynomial time algorithm to solve this simple class of linear programs, for even special cases of f?

Acknowledgements

I am grateful to Satoru Iwata, Tom McCormick, and Éva Tardos for sharing their insights on the subject of submodular functions and flows. I am indebted to them and also to Garud Iyengar, Jay Sethuraman, and Kevin Wayne for taking the time to read and provide useful criticism on a draft of this article.

References

[1] *Proceedings of the 32th Annual ACM Symposium on Theory of Computing*, 2000.

[2] R.E. Bixby, W.H. Cunningham, and D.M. Topkis. Partial order of a polymatroid extreme point. *Math. Oper. Res.*, 10:367-378, 1985.

[3] W.H. Cunningham. Testing membership in matroid polyhedra. *J. Combinatorial Theory B*, 36:161-188, 1984.

[4] W.H. Cunningham. Minimum cuts, modular functions, and matroid polyhedra. *Networks*, 1985.

[5] W.H. Cunningham. On submodular function minimization. *Combinatorica*, 5:185-192, 1985.

[6] W.H. Cunningham and A. Frank. A primal-dual algorithm for submodular flows. *Math. Oper. Res.*, 10:251-262, 1985.

[7] E.A. Dinic. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Dokl.*, 1970.

[8] J. Edmonds. Submodular functions, matroids, and certain polyhedra. In R. Guy, H. Hanani, N. Sauer, and J. Schönheim, editors, *Combinatorial Structures and Their Applications*, pages 69-87. Gordon and Breach, 1970.

[9] J. Edmonds and R. Giles. A min-max relation for submodular functions on graphs. *Ann. Discrete Math.*, 1:185-204, 1977.

[10] J. Edmonds and R.M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19:248-264, 1972.

[11] T.R. Ervolina and S.T. McCormick. Two strongly polynomial cut cancelling algorithms for minimum cost network flow. *Disc. Appl. Math.*, 46:13-165, 1993.

[12] L. Fleischer and S. Iwata. Improved algorithms for submodular function minimization and submodular flow. In ACM [1], pages 107-116.

[13] L. Fleischer and S. Iwata. A push-relabel framework for submodular function minimization and applications. Working paper, 2000.

[14] L. Fleischer, S. Iwata, and S.T. McCormick. A faster capacity scaling algorithm for submodular flow. Technical Report 9947, C.O.R.E. Discussion Paper, Louvain-la-Neuve, Belgium, 1999.

[15] L.R. Ford and D.R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962.

[16] A. Frank. Finding feasible vectors of Edmonds-Giles polyhedra. *J. Combin. Theory*, 36:221-239, 1984.

[17] A. Frank and É. Tardos. Generalized polymatroids and submodular flows. *Math. Programming*, 42:489-563, 1988.

[18] A. Frank and É. Tardos. An application of submodular flows. *Linear algebra and its applications*, 114/115:329-348, 1989.

[19] S. Fujishige. Lexicographically optimal base of a polymatroid with respect to a weight vector. *Math. Oper. Res.*, 5:186-196, 1980.

[20] S. Fujishige. *Submodular Functions and Optimization*. North-Holland, 1991.

[21] S. Fujishige, H. Röck, and U. Zimmermann. A strongly polynomial algorithm for minimum cost submodular flow problems. *Math. Oper. Res.*, 14:60-69, 1989.

[22] S. Fujishige and X. Zhang. New algorithms for the intersection problem of submodular systems. *Japan J. Indust. Appl. Math.*, 9:369-382, 1992.

[23] G. Gallo, M.D. Grigoriadis, and R.E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18(1):30-55, 1989.

[24] A.V. Goldberg and R.E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35:921-940, 1988.

[25] A.V. Goldberg and R.E. Tarjan. Solving minimum cost flow problems by successive approximation. *Mathematics of Operations Research*, 15:430-466, 1990.

[26] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169-197, 1981.

[27] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1988.

[28] J. Hao and J.B. Orlin. A faster algorithm for finding the minimum cut in a directed graph. *J. Algorithms*, 17(3):424-446, 1994.

[29] Hoppe and É. Tardos. The quickest transshipment problem. *Math. Oper. Res.*, 25(1), February 2000. Extended abstract appeared in Proceedings of SODA 1995.

[30] S. Iwata. A capacity scaling algorithm for convex cost submodular flow. *Math. Programming*, 76:299-308, 1997.

[31] S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. In *ACM* [1], pages 97-106.

[32] S. Iwata, S. T. McCormick, and M. Shigeno. A strongly polynomial cut canceling algorithm for the submodular flow problem. In *7th International Integer Programming and Combinatorial Optimization Conference*, pages 259-272, 1999.

[33] S. Iwata, K. Murota, and M. Shigeno. A fast parametric submodular intersection algorithm for strong map sequences. *Math. Oper. Res.*, 22:803-813, 1997.

[34] E.L. Lawler and C.U. Martel. Computing maximal polymatroidal network flows. *Math. Oper. Res.*, 7:334-347, 1982.

[35] L. Lovász. Submodular functions and convexity. In A. Bachem, M. Grötschel, and B. Korte, editors, *Mathematical Programming - The State of the Art*, pages 235-257. Springer-Verlag, 1983.

[36] S.T. McCormick. Fast algorithms for parametric scheduling come from extensions to parametric maximum flow. *Operations Research*, 47(5):744-756, September-October 1999. Extended abstract appeared in Proceedings of STOC 96.

[37] N. Megiddo. Optimal flows in networks with multiple sources and sinks. *Mathematical Programming*, 7:97-107, 1974.

[38] H. Nagamochi and T. Ibaraki. Computing edge-connectivity of multigraphs and capacitated graphs. *SIAM J. Disc. Math.*, 5:54-66, 1992.

[39] M. Queyranne. Minimizing symmetric submodular functions. *Math. Programming*, 82:3-12, 1998. Extended abstract appeared in Proceedings of SODA 1995.

[40] P. Schönsleben. *Ganzzahlige Polymatroid-Intersektions-Algorithmen*. PhD thesis, ETH, Zurich, 1980.

[41] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. Preprint.

[42] L.S. Shapley. Cores of convex games. *International Journal of Game Theory*, 1:11-26, 1971.

[43] É. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5:247-255, 1985.

[44] É. Tardos, C.A. Tovey, and M.A. Trick. Layered augmenting path algorithms. *Math. Oper. Res.*, 11:362-370, 1986.

[45] D.M. Topkis. *Supermodularity and Complementarity*. Frontiers of Operations Research. Princeton University Press, Princeton, NJ, 1998.

Conference CALENDAR

The First Sino-Japan Optimization Meeting
October 26-28, 2000, Hong Kong
http://www.cs.uu.nl/events/ipco2001

INFORMS Fall 2000
November 3-7, 2000, San Antonio, Texas, USA
URL: http://ie.tamu.edu/informs2000/

IPCO VIII
June 13-15, 2001, Utrecht, The Netherlands
and DONET Summer school on Integer and Combinatorial Optimization
June 11-12, 2001, Utrecht, The Netherlands
http://www.cs.uu.nl/events/ipco2001

IPCO VIII
Utrecht, The Netherlands
June 13-15, 2001
and DONET Summer school on Integer and Combinatorial Optimization
Utrecht, The Netherlands
June 11-12, 2001

The eighth Integer Programming and Combinatorial Optimization (IPCO) conference will be held in Utrecht, The Netherlands, from June 13 to 15, 2001. The IPCO conference will be immediately preceded by a summer school on Integer and Combinatorial Optimization. The summer school will be given by Daniel Bienstock and Éva Tardos, who will each present four one-hour lectures. The school is organized under the auspices of DONET, a European Network for Discrete Optimization subsidized by the European Community. We hope that the combination of the two activities will make it easier for young researchers to participate in the joint event even if an IPCO submission is not made.
For more information, please see the home page of the IPCO conference at <http://www.cs.uu.nl/events/ipco2001> or send an e-mail to <ipco2001@cs.uu.nl>.
To be added to our mailing list, please fill out the pre-registration form at the web site or send your name and e-mail address to <ipco2001@cs.uu.nl>.
–Karen Aardal, The ICPO VIII Organization Committee

First Announcement

The First Sino-Japan Optimization Meeting

Hong Kong, October 26-28, 2000

Objectives The meeting aims to provide a forum for researchers, who are from Japan, Mainland China, Hong Kong, Taiwan, Singapore, other countries and regions, and working in the area of optimization, to gather together to report and exchange their latest works on optimization.

Topics Include Linear and Nonlinear Optimization, Continuous and Discrete Optimization, Deterministic and Stochastic Optimization, Smooth and Nonsmooth Optimization, Single- and Multi-Objective Optimization, Integer and Combinatorial Optimization, Convex and Nonconvex Optimization

Organization and Endorsement

The Sino-Japan Optimization Meeting (SJOM) is endorsed by the Mathematical Programming Society (MPS), the Research Association of Mathematical Programming (RAMP), Japan, and the Chinese Mathematical Programming Society. The First Sino-Japan Optimization Meeting (SJOM 2000) is organized by The City University of Hong Kong and The Hong Kong Polytechnic University. The organization committee of SJOM 2000 is in charge of the organization work of SJOM 2000, and will report to the steering committee of SJOM. The steering committee will assist the organization of SJOM 2000, and select the organizers and locations of the future SJOM meetings. It is expected that the steering committee will formally meet during SJOM 2000. The term of the co-chairs of the steering committee will end at the second SJOM meeting.

Organization Committee of SJOM 2000

Liqun Qi (<maqilq@polyu.edu.hk>, The Hong Kong Polytechnic University), Co-Chair; Jianzhong Zhang (<mazhang@cityu.edu.hk>, City University of Hong Kong), Co-Chair; Chuangyin Dang (<mecdang@cityu.edu.hk>, City University of Hong Kong); and Xiaoqi Yang (<mayangxq@polyu.edu.hk>, The Hong Kong Polytechnic University), Treasurer.

Steering Committee of SJOM

Xiaoqiang Cai (The Chinese University of Hong Kong); Xiaojun Chen (Shimane University); Satoru Fujishige (Osaka University); Masao Fukushima (Kyoto University), Co-Chair; Jiye Han (Chinese Academy of Sciences); Toshihide Ibaraki (Kyoto University); Masakazu Kojima (Tokyo Institute of Technology); Hiroshi Konno (Tokyo Institute of Technology); Shinji Mizuno (Tokyo Institute of Technology); Kazuo Murota (Kyoto University); Liqun Qi (The Hong Kong Polytechnic University); Jie Sun (The National University of Singapore); Kunio Tanabe (Institute of Statistics Mathematics); Tetsuzo Tanino (Osaka University); Kok Lay Teo (The Hong Kong Polytechnic University), Co-Chair; Soon-yi Wu (National Cheng Kung University); Wenci Yu (East China University of Science and Technology); Ya-xiang Yuan (Chinese Academy of Sciences); Jianzhong Zhang (City University of Hong Kong); Xiangsun Zhang (Chinese Academy of Sciences)

Guest Plenary Speakers

Rainer Burkard (Technische Universitat Graz, Austria), Gianni Di Pillo (Universita di Roma “La Sapienza,” Italy), Carl Timothy Kelley (North Carolina State University, USA), Jean-Philippe Vial (University of Geneva, Switzerland)

Special Arrangements

Meeting proceedings or special issues of some journals, tours and economical hotel accommodations will be indicated in the Second Announcement.

Further Information

E-Mail: Eva Yiu (maevayiu@polyu.edu.hk) or Peggy Chan (machan@cityu.edu.hk); web site: <http://www.polyu.edu.hk/~ama>; or contact Organization Committee Members (e-mail addresses shown above)

Call for Papers

COAP
Computational Optimization and Applications
Special Issue on Stochastic Programming

Computational Optimization and Applications (COAP) announces a call for papers in the area of Stochastic Programming with special emphasis on new computational methods, experimental results as well as applications of stochastic programming methodology. We use the term “stochastic programming” in a broad sense to cover stochastic optimization problems in which information is revealed sequentially over time. Thus, traditional dynamic programming as well as its more recent adaptations are also considered relevant to this special issue. As suggested by the focus of COAP, we are particularly interested in computational approaches and significant applications. Hence submitted papers should be motivated by these thrusts. As one might expect, a primary requirement for publication in this issue is novelty of the ideas expounded in the paper.

The special issue, which will be edited by Professors Suvrajeet Sen and Julia Higle, will be refereed in accordance with established procedures of COAP. *Please send five copies of the paper to:* Melissa Sullivan, Computational Optimization and Applications Editorial Office, Kluwer Academic Publishers, 101 Philip Drive, Norwell, MA 02061, USA. *Important:* In your cover letter, provide your e-mail address and state that the paper is for consideration in the special issue on Stochastic Programming. Papers for the special issue should be submitted by December 15, 2000.

Sixth International Symposium on Generalized Convexity/Monotonicity

Karlovassi, Samos, Greece
August 30 – September 3, 1999
preceded by Summer School on Generalized Convexity/Monotonicity, August 25-28,1999

The Symposium and its preceding Summer School were organized by the Working Group on Generalized Convexity (WGGC). About thirty students participated in the Summer School and over one hundred researchers attended the Symposium. The Summer School, directed by J.B.G. Frenk, introduced graduate students as well as researchers from other fields to major topics of Generalized Convexity and Generalized Monotonicity. The material was covered in twelve tutorials presented by J.P. Crouzeix, J.B.G. Frenk, N. Hadjisavvas, D.T. Luc, J.E. Martinez-Legaz, P.M. Pardalos, J.P. Penot and S. Schaible. Most "students" of the Summer School stayed on for the Symposium while several participants of the Symposium arrived early to attend tutorials of the Summer School. The response to the first WGGC summer school preceding a WGGC symposium was overwhelmingly positive. Thanks to the efforts of M. Sniedovich, IFORS representative, the three best students of the Summer School obtained an IFORS scholarship. During the following week more than fifty lectures were presented and followed by researchers from twenty-six countries. Topics covered included various kinds of generalized convex functions and generalized monotone maps, optimality conditions, duality, fractional programming, multiobjective programming, nonsmooth analysis, variational inequalities, equilibrium problems as well as topics less represented at previous symposia such as stochastic convexity and global optimization. Furthermore, as at the last symposium, several invited lectures introduced participants to neighboring fields of generalized convexity. This time set-valued optimization, multiplicative/fractional programming, global optimization and stochastic programming were the emphasis with tutorials by J. Jahn (Germany), H. Konno (Japan), P.M. Pardalos (USA) and A. Prekopa (USA), respectively.

The conference site, a small town on a Greek island, offered limitless opportunities for professional contacts among the participants. A rich scientific program was complemented by a social program with many highlights. The Symposium was hosted by the Department of Mathematics of the University of the Aegean, located on Samos, the birthplace of Pythagoras and Aristarchus. This truly scenic island with its interesting archeological sites gave the conference a special background. The participants left enriched and appreciative of the excellent organization by N. Hadjisavvas (Samos) who with his local and international team put together this memorable event in the history of WGGC. Further details are found on the web site of the Summer School and the Symposium at <http://www.samos.aegean.gr/math/gc6/>. A selection of refereed papers presented at the Symposium will appear in the Proceedings to be published by Springer Verlag in the series "Lecture Notes in Economics and Mathematical Systems" (eds. N. Hadjisavvas, J.E. Martinez-Legaz and J.P. Penot). For information on the proceedings of the previous five symposia in Vancouver (1980), Canton (1986), Pisa (1988), Pecs (1992) and Marseille-Lyminy (1996), see the web site of WGGC (http://genconv.ec.unipi.it/).
–Siegfried Schaible, Scientific Committee of WGGC (siegfried.schaible@ucr.edu)

STANFORD UNIVERSITY
Department of Management Science & Engineering

Applications are invited for a tenure-track faculty position in the field of optimization. The rank is open, and all branches of optimization are of interest, e.g., continuous, discrete, large scale, etc. In particular the department seeks a new faculty member who has an outstanding methodological foundation and strong applications interests. If appropriate, the successful candidate may choose to become involved in School of Engineering activities in the area of Computational and Mathematical Engineering. The department hopes to fill the position by September 1, 2001.

The Department of Management Science & Engineering (MS&E) is newly created by the merger of the Engineering-Economic Systems and Operations Research Department and the Industrial Engineering and Engineering Management Department. The department's mission is research and education associated with the development of the knowledge, tools, and methods required to make decisions, shape policies, configure organizational structures, design engineering systems, and solve operational problems arising in the information-intensive, technology-based economy. The department has special interest in theory and application in optimization and systems modeling, probability and stochastic systems, production operations and manufacturing, decision analysis and risk analysis, economics and finance, organizational behavior, and management and entrepreneurship. The department is also developing expertise in information science and technology and in technology, policy and strategy. A more complete description appears in the department Web site at http://www.stanford.edu/dept/msande/.

Applicants should send a resume (including research accomplishments, teaching experience, publications), transcript of (doctoral) graduate study, at least one published or unpublished research paper if available, and names and addresses of at least three references to: Professor Richard Cottle, Search Committee Chair, Department of Management Science & Engineering, Terman Engineering Center, Stanford University, Stanford, CA 94305-4026. They should also ask referees to send recommendation letters directly to Professor Cottle. Review of the applications will begin in January of 2001.

Stanford University is an equal opportunity employer and welcomes nominations of women and minority group members and applications from them.

FAP Web Announcement

We are happy to announce that the WWW server, FAP Web on Frequency Assignment, is now publicly available at <http://fap.zib.de/>. Our intention is to collect and supply information related to solving frequency assignment problems.

We have compiled a (certainly still incomplete) list of publications on frequency assignment methods and collected results on the CALMA, COST 259, and Philadelphia instances. We would like to keep track of the latest results obtained as well as to supply benchmark problems for download. This cannot be done without your support.

We would appreciate your assistance with helping to disseminate the information on FAP Web, offering suggestions to improve FAP Web, and keeping us updated. All information can be submitted with forms available at FAP Web (Submit section), or by sending an e-mail to <fap@zib.de>. You are also encouraged to fill in the form concerning your contact address if you'd like this address listed at the server. (This may increase the likelihood of receiving spam mail, however.)

Finally, we would like to take the opportunity to thank everybody who has (mostly indirectly) contributed so far, be it by sending (preprints of) papers, by supplying benchmark instances, or by providing assignments for some of the instances.
–Andreas Eisenblaetter (eisenblaetter@zib.de) and Arie Koster (koster@zib.de)

mind sharpener

We invite OPTIMA readers to submit solutions to the problems to Robert Bosch (bobb@cs.oberlin.edu). The most attractive solutions will be presented in a forthcoming issue.

Lights Out

by Robert A. Bosch

Dec. 14, 1999

In Tiger Electronics' handheld electronic solitaire game *Lights Out*, the player strives to turn out all 25 lights that make up a 5 x 5 grid of cells. On each turn, the player is allowed to click on any one cell. Clicking on a cell activates a switch that causes the states of the cell and its neighbors to change from on to off or from off to on. Corner cells are considered to have two neighbors, edge cells to have three, and interior cells to have four. Figure 1 demonstrates what happens when the player clicks on cells (1, 1) and (1, 2).

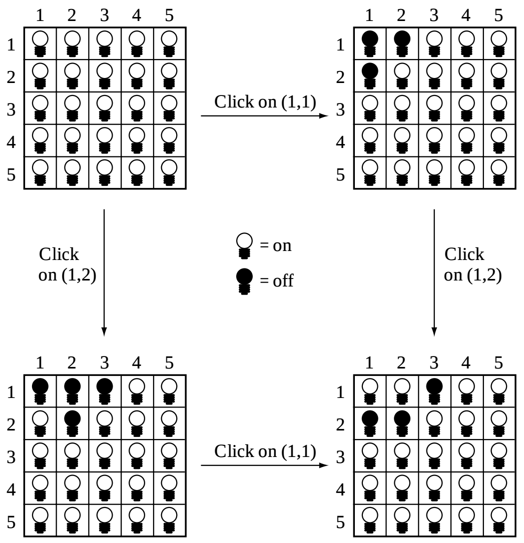


Figure 1.

Problems Interested readers may enjoy trying to solve the following problems:

1. Formulate an integer program for finding a way to turn out all the lights in as few turns as possible. *Hint 1:* The order in which the cells are clicked doesn't matter. *Hint 2:* A cell shouldn't be clicked more than once.
2. What if each cell has a three-way bulb? (Repeatedly clicking on a single three-way bulb changes its state from off to low, from low to medium, from medium to high, from high to off, and so on.) Which is easiest: (a) turning off all the lights when they're all on their high setting? (b) turning them off when they're all on medium? (c) turning them off when they're all on low?

Readers who have a Java-enabled browser can play the game online by going to Martin Chlond's "Integer Programming and Recreational Mathematics" web page (www.chlond.demon.co.uk/academic/puzzles.html) and following the link to the "Five by five puzzle." (In Chlond's version, all of the lights are initially off and the goal is to turn them all on.) To the author's knowledge, Chlond was the first person to use integer programming to find an optimal solution to a Lights Out-type game. Incidentally, the rest of Chlond's web page is well worth a visit too. It is a lovely collection of recreational mathematics problems that can be solved with integer programming.

Armies of Queens, Revisited

In the June installment of *OPTIMA Mind Sharpener*, we presented a variant of the well-known 8-queens problem. We stated that two armies of queens (black and white) *peaceably coexist* on a chessboard when they are placed on the board in such a way that no two queens from opposing armies can attack each other. We then asked readers to formulate an integer program to find the maximum size of two equal-sized peaceably coexisting armies.

Frank Plastria submitted the following very nice, very simple IP formulation of the problem:

$$
\begin{array}{lll}
\max & \displaystyle\sum_{i=1}^{8}\sum_{j=1}^{8} b_{i,j} & \\
\text{s.t.} & \displaystyle\sum_{i=1}^{8}\sum_{j=1}^{8} b_{i,j} = \sum_{i=1}^{8}\sum_{j=1}^{8} w_{i,j} & (1)\\
 & b_{i_1,j_1} + w_{i_2,j_2} \le 1 \qquad \text{for all } \big((i_1,j_1),(i_2,j_2)\big) \in M & (2)\\
 & b_{i,j}, w_{i,j} \in \{0,1\} \qquad \text{for all } 1 \le i, j \le 8. &
\end{array}
$$

Plastria's formulation has two binary variables for each square (i, j) of the board. One of them, b_i, j, indicates whether or not square (i, j) holds a black queen. The other, w_i, j, specifies whether or not square (i, j) contains a white queen. The constraint (1) keeps the armies the same size. The constraints (2) keep the armies at peace. The set M is the set of all ordered pairs of chessboard squares that share a row, column, or diagonal line of the chessboard. (Note that $((i_1,j_1), (i_2,j_2)) \in M$ if and only if $i_1 = i_2$, $j_1 = j_2$, or $|i_1 - i_2| = |j_1 - j_2|$.)

Due to a lack of adequate IP software, Plastria was unable to implement his formulation. He did prove, however, that the optimal value of the LP relaxation of his integer program is 32, and he did conjecture that the optimal value of his integer program is 9. (The best solution he was able to find has nine queens in each army and is displayed in Figure 2.)

Plastria's formulation is not a tight one. The author tested it with CPLEX (version 4.0.9) on a 200 MHz Pentium PC and obtained an optimal solution (with objective value equal to 9) in just over four hours. It turns out that there are many optimal solutions. The author hasn't yet classified all of them. His favorite one (for the moment) is displayed in Figure 3.

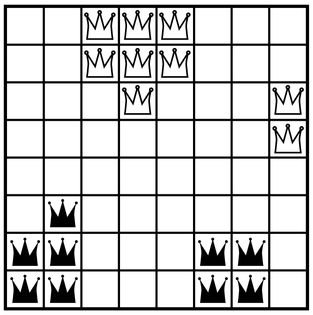


Figure 2.

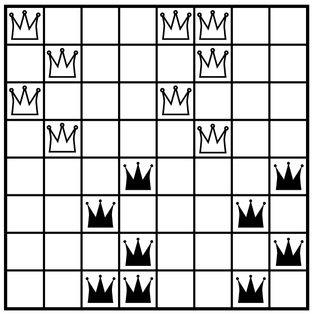
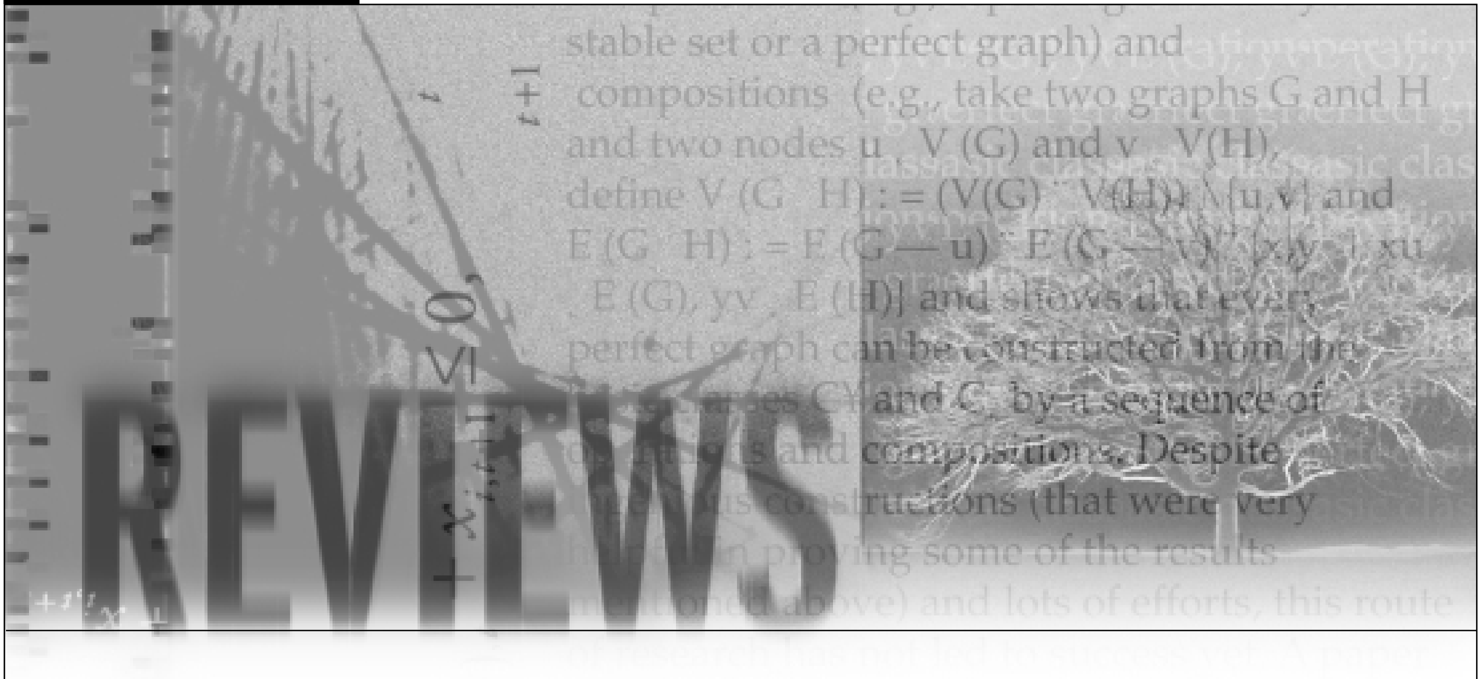


Figure 3.



Integer Programming

by Laurence A. Wolsey

Wiley 1998

ISBN 0-471-28366-5, GBP 37.50

Integer programming is a powerful, but finicky tool; while it provides a very broad framework for applications, care needs to be taken in the model building process in order to effectively use the current generation of solution methods. Laurence Wolsey is the world's master at squeezing the most out of the IP framework; watching him manipulate the constraints of an IP is like seeing a lecture at the Hogwarts School of Witchcraft and Wizardry. It is very fortunate that Wolsey's text is aimed squarely at teaching the rest of us the ins and outs of IP practice.

Wolsey's book is a true textbook. It is about one-third the size of the classic *Integer and Combinatorial Optimization* by G.L. Nemhauser and L.A. Wolsey, and it contains a factor of 148 times more exercises than the monumental work *Theory of Linear and Integer Programming* by A. Schrijver. I highly recommend Wolsey's book for adoption as a text in undergraduate and graduate courses. Indeed, in our department we created a new course to match Wolsey's text, and it is being very well received by the senior undergraduates, as well as by students in our Ph.D. program. The text, moreover, is also a nice vehicle for potential industrial users of integer programming to see how to best use IP software tools.

After two introductory chapters, a quick treatment of some basic combinatorial algorithms (shortest paths, minimum spanning trees, bipartite matching) is presented in chapters 3 and 4. This is followed by chapters on dynamic programming, complexity theory, and branch and bound. The heart of the book begins with the cutting-plane material presented in chapters 8 and 9. Wolsey gives an extensive treatment of general purpose cutting planes as well as facet-defining inequalities for special classes of integer and mixed-integer problems. Lagrangian relaxation and column generation methods are covered in chapters 10 and 11, and a short treatment of heuristic algorithms is given in chapter 12. The text concludes with an excellent summary chapter titled "From Theory to Solutions," that should be of particular interest to IP practitioners. This final chapter is presented as a series of questions and answers, guiding the reader through the modeling and solution process.

Wolsey's book is a pleasure to read, and the author has done a very good job at selecting the material to cover. *Integer Programming* will no doubt be the standard IP text for years to come.

–BILL COOK

Scheduling Algorithms

by Peter Brucker

Springer-Verlag, 1998

ISBN 3-540-64105-X, DM 138

This is a slightly extended (16 additional pages) and updated version of the book with the same title that appeared in the first edition in 1995. The structure and the titles of the chapters are the same as in the first edition. The book deals with deterministic problems in connection with machine or processor scheduling, and it concentrates on the presentation of polynomial algorithms for different classes of scheduling problems. Moreover, branch and bound algorithms as well as local search algorithms for the exact and heuristic solution of NP-hard scheduling problems are briefly discussed. The book consists basically of three parts.

The first part (Chapters 1 - 3) contains some elementary material dealing with the classification of scheduling problems, with some basic problems and algorithms that are relevant for the development of solution procedures in connection with scheduling problems (e.g. linear and integer programming, maximum flow problems, matching problems, arc coloring in bipartite graphs and dynamic programming) and with the computational complexity of scheduling problems.

Chapters 4 - 6 cover classical scheduling algorithms for solving single machine problems (Chapter 4), parallel machine problems (Chapter 5) and shop scheduling problems (open shop, flow shop, job shop and mixed shop; Chapter 6). This is the main part of the book and contains known scheduling algorithms developed during the past 40 years. In Chapter 6, the disjunctive graph model as a useful tool for constructing optimal schedules is introduced. The job shop problem with minimizing the makespan is considered in detail. In particular, a branch and bound algorithm based on the block approach and immediate selection is described, which was the first algorithm that was able to solve the famous 10 x 10 problem given by Muth and Thompson (1963), and the application of tabu search techniques for the heuristic solution of this problem is discussed.

The last part (Chapters 7 - 11) deals with some topics that have been considered recently, particularly in connection with flexible manufacturing, and which are not or only partially included in other recent books on scheduling. More specifically, single machine scheduling problems involving due dates (Chapter 7), single machine batching problems (Chapter 8), scheduling problems with changeover and transportation times (Chapter 9), problems with multi-purpose machines (Chapter 10) and problems with multiprocessor tasks (Chapter 11) are treated.

Most extensions in comparison with the first edition can be found in Chapters 8 and 11. In contrast to the first edition, where only batching problems with the length of a batch equal to the sum of the processing times of the jobs in the batch have been considered, the new edition includes also p-batching problems, where the length of a batch is equal to the maximum of the processing times of the jobs in the batch. Both the unbounded (the batch can contain arbitrarily many jobs) and the bounded model are considered. The chapter on multiprocessor tasks has been extended by some comments on preemptive multiprocessor task problems and on multi-mode multiprocessor task problems.

The chapters finish with a survey of complexity results, where both the maximal polynomially solvable problems and the minimal NP-hard problems are listed in the tables. These tables have been updated in comparison with the first edition of the book, and they now also contain a lot of results that have been recently obtained. The book is a very helpful tool for the development of concrete scheduling algorithms.

–FRANK WERNER, OTTO-VON-GUERICKE UNIVERSITÄT MAGDEBURG

Handbook of Computational Geometry

edited by J.R. Sack and J. Urrutia

Elsevier Science, North-Holland

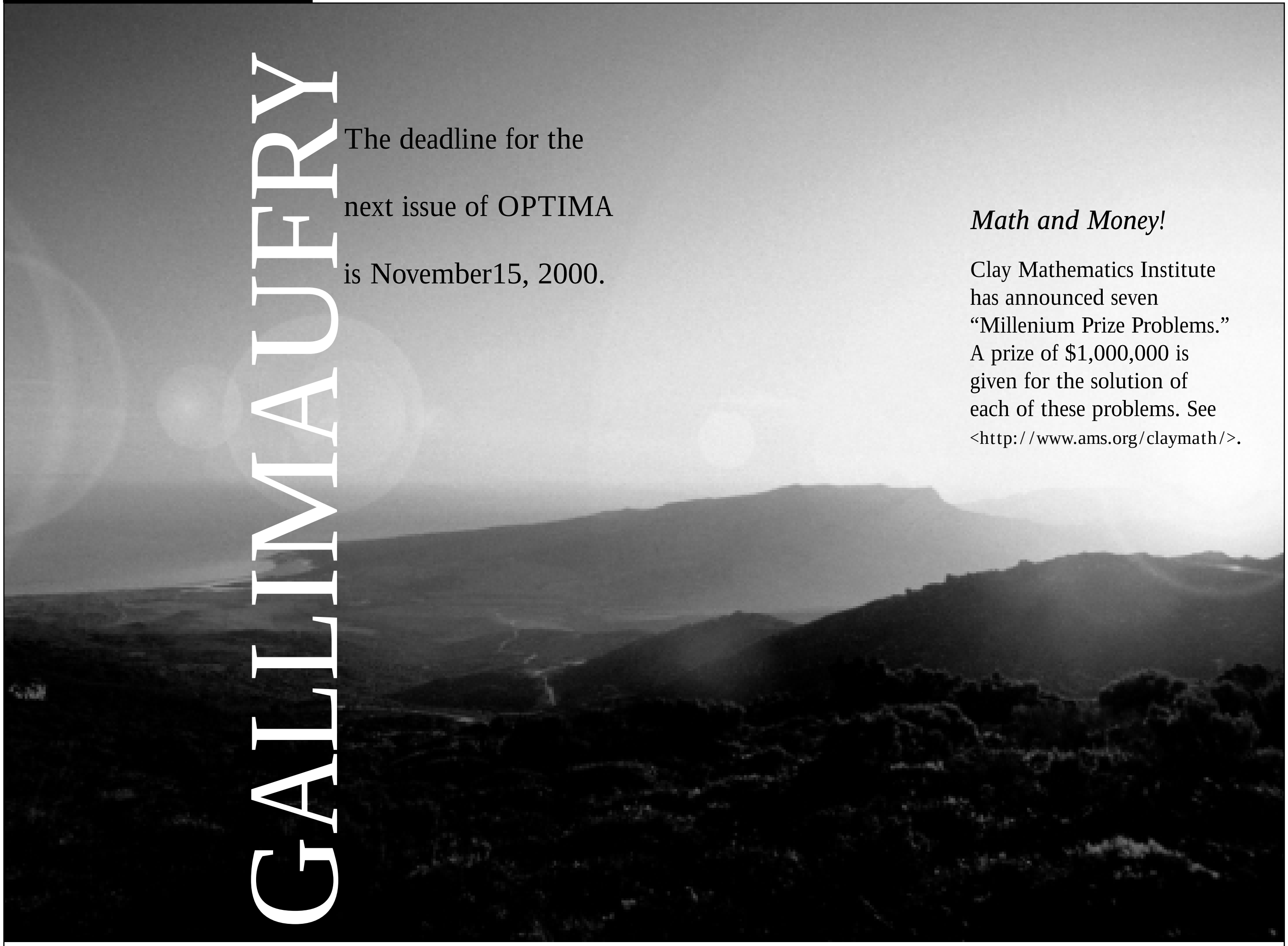
ISBN 0-444-82537-1, 1088 PAGES, USD 190.50

Compared with its important influence on today's Geometry and its strong impact on other fields inside and outside mathematics, a comprehensive and extensive source on *Computational Geometry* was long overdue. However, is there a chance to cover at least most of the variety of this field, ranging from Algorithms and Data Structure, Optimization and Euclidean Geometry to applications in Visualizations, Robotics, CAD, etc., in only one book? There is! And the proof is given by the *Handbook on Computational Geometry* edited by J. R. Sack and J. Urrutia.

This handbook contains 22 chapters, most of them devoted to basic problems and concepts in *Computational Geometry* like Davenport-Schinzel sequences, Arrangements, Voronoï Diagrams, Closest-Point Problems, Link Distance Problems, Graph-Drawing, Similarity of Geometric Shapes, Spanning Trees and Spanners, Visibility, Animation, Geometric Data Structures, Spatial Data Structures, Illumination Problems, Polygon Decomposition, Parallel Computational Geometry, Randomized Algorithms, Derandomization and Robustness in Geometric Computation. In addition, three chapters on Geographical Information Systems, Geometric Shortest Path and Network Optimization, and Mesh Generation show in an impressive manner the interaction of *Computational Geometry* with other fields of science.

My overall impression is that most of the chapters are written in a very comprehensive, detailed and stimulating way. Each chapter contains an extensive list of references, and there is no doubt that this book will be a very valuable resource for researchers. However I can also recommend this handbook to everyone who just wants to get an impression of the problems, tools and techniques used in *Computational Geometry*, or who is just curious about what is going on in this field. I am sure that this book will satisfy her/his curiosity and thirst for knowledge.

–MARTIN HENK, MAGDEBURG



GALLIMAUFRY

The deadline for the next issue of OPTIMA is November15, 2000.

Math and Money!

Clay Mathematics Institute has announced seven "Millenium Prize Problems." A prize of $1,000,000 is given for the solution of each of these problems. See <http://www.ams.org/claymath/>.

Application for Membership

I wish to enroll as a member of the Society.

My subscription is for my personal use and not for the benefit of any library or institution.

☐ I will pay my membership dues on receipt of your invoice.

☐ I wish to pay by credit card (Master/Euro or Visa).

CREDITCARD NO. ________ EXPIRY DATE ________

FAMILY NAME ________

MAILING ADDRESS ________

TELEPHONE NO. ________ TELEFAX NO. ________

EMAIL ________

SIGNATURE ________

Mail to:

Mathematical Programming Society
3600 University City Sciences Center
Philadelphia PA 19104-2688 USA

Cheques or money orders should be made payable to The Mathematical Programming Society, Inc. Dues for 1999, including subscription to the journal *Mathematical Programming*, are US $75.
Student applications: Dues are one-half the above rate. Have a faculty member verify your student status and send application with dues to above address.

Faculty verifying status ________

Institution ________