

Fast algorithms for even/odd minimum cuts and generalizations

András A. Benczúr^{1*} and Otilia Fülöp^{2**}
{benczur, otti}@cs.elte.hu

¹ Computer and Automation Institute, Hungarian Academy of Sciences, and
Department of Operations Research, Eötvös University, Budapest

² Institute of Mathematics, Technical University, Budapest

Abstract. We give algorithms for the directed minimum odd or even cut problem and certain generalizations. Our algorithms improve on the previous best ones of Goemans and Ramakrishnan by a factor of $O(n)$ (here n is the size of the ground vertex set). Our improvements apply among others to the minimum directed T-odd or T-even cut and to the directed minimum Steiner cut problems. The (slightly more general) result of Goemans and Ramakrishnan shows that a collection of *minimal* minimizers of a submodular function (i.e. minimum cuts) contains the odd minimizers. In contrast our algorithm selects an n -times smaller class of *not necessarily minimal* minimizers and out of these sets we construct the odd minimizer. If $M(n, m)$ denotes the time of a u - v minimum cut computation in a directed graph with n vertices and m edges, then we may find a directed minimum

- odd or T-odd cut with V (or T) even in $O(n^2m + n \cdot M(n, m))$ time;
- even or T-even cut in $O(n^3m + n^2 \cdot M(n, m))$ time.

The key of our construction is a so-called *parity uncrossing* step that, given an arbitrary set system with odd intersection, finds an odd set with value not more than the maximum of the initial system.

1 Introduction

The notion of minimum cuts in graphs and its generalization, the minimum of a submodular function, plays a key role in combinatorial optimization. See [15] for a survey of the application of minimum cuts and [13] for that of submodular functions.

In this paper we consider minimum cuts or minimizers of submodular functions by restricting minimization to either the even or the odd sets. More generally we will minimize in so-called triple families ([9] and [6]), set systems that satisfy certain properties of even or odd (for the definition see Section 1.3). The fact that the minimum odd cut or T-odd cut (requiring odd intersection with a prescribed set T) problems can be solved in polynomial time by maximum flow computations is first shown in [14]; the same for even cuts as well as its generalizations for matroids is shown in [2].

* Supported from grants OTKA T-30132 and T-29772; AKP 98-19; FKFP 0206/1997

URL: <http://www.cs.elte.hu/~benczur>

** Supported from OTKA grant T-29772

1.1 Minimum cut problems. The cut of the graph is a bipartition of its vertex set V into $C \subset V$ and its complement; the value of the cut $f(C)$ in a directed graph is the number or the total capacity of the edges leaving C . We consider the following optimization problems related to cuts in directed graphs:

- The minimum odd (even) cut problem asks for a cut $C \neq \emptyset, V$ such that $|C|$ is odd (even) with $f(C)$ minimum.
- The minimum T-odd (T-even) cut problem asks for a cut $C \neq \emptyset, V$ such that $|C \cap T|$ is odd (even) with $f(C)$ minimum.
- For a given integer p , we ask for the minimizer C of the cut value function f with $|C|$ (or for a given set T , the intersection $|C \cap T|$) not divisible by p .
- The generalized directed Steiner cut problem asks for a cut C with minimum value $f(C)$ in a family of cuts defined as follows. Given sets $T_1, \dots, T_k \subseteq V$, a cut C is a *generalized Steiner cut* if for some $i \leq k$ we have $\emptyset \neq C \cap T_i \neq T_i$. (For the notion and use of *undirected* Steiner cuts see [7].)

1.2 Submodular functions. A function f over all subsets of a ground set V is called *submodular* if all $X, Y \subseteq V$ satisfy $f(X) + f(Y) \geq f(X \cap Y) + f(X \cup Y)$.

An example of a submodular function is the cut value function f seen in the previous subsection. In this paper we generalize the cut problems of Section 1.1 to arbitrary submodular functions; for example we will consider the problem of finding the set $Y \subseteq V$ with $f(Y)$ minimum such that $|Y|$ is odd or even.

1.3 Triple families: a generalization of even/odd. Grötschel et al. [9–11] generalize the notion of an odd set and define a *triple family* as follows.

A family of subsets of ground set V forms a *triple family* if for all members X and Y whenever three of the four sets $X, Y, X \cap Y$ and $X \cup Y$ are not in the triple family, then so is the fourth.

Notice that each example in Section 1.1 asks for the minimum value cut in a certain triple family; thus all these problems form special cases of the algorithms described in this paper. This fact is immediate for cuts with cardinality odd, even, or not divisible by p . It is also easy to see that directed generalized Steiner cuts form a triple family. For the undirected version this follows by the results of [7] and [6]; we obtain the same for the directed version since the definition of a Steiner cut is symmetric.

To simplify the notation we will call members of the triple family *odd* while other sets *even*. Notice that this notion of even and odd is not related to the actual cardinality of a set C ; in fact when we optimize for even sets, an odd set is a set with even cardinality. Also note that the empty set may, in this sense, be odd as well.

In the discussion we will only use the next two key properties of triple families. Strictly speaking, the second property is more restrictive than the definition of Grötschel et al. [9–11] since they also consider triple families of lattices where we cannot take the complement of a set.

- (*) if X and Y are even (non-members) and $X \cap Y$ (or $X \cup Y$) is odd, then so is $X \cup Y$ (or $X \cap Y$).
- (**) if \emptyset is even and $X_1 \subset X_2$ with both X_1 and X_2 even, then $X_2 - X_1$ is even.

1.4 Previous results. The previous best result of Goemans and Ramakrishnan [6] finds the minimum f -value odd set over a ground set with $|V| = n$ by $O(n^2)$ calls to a submodular minimization oracle. Their algorithm also applies to a slight generalization of triple families called *parity families*, a notion defined in Section 4.4. The minimizer they find may both be the empty set as well as the entire ground set V . If one wants to exclude these possibilities (and \emptyset or V belongs to the triple family), then the algorithm must be repeated $O(n)$ times, thus yielding $O(n^3)$ oracle calls.

Goemans and Ramakrishnan [6] prove that a minimal odd minimizer of a submodular function f is either \emptyset or V or it is among the (inclusionwise) minimal minimizers X of function f with the property that X has $u \in X$ and $v \notin X$, where u and v takes all possible values from V . Hence for a minimum cut problem they must use a flow/cut algorithm that returns the *minimal* minimum cut, but for the price of this additional restriction they are able obtain a minimal minimizer.

As for earlier results, the first one related to even/odd minimization are those of Padberg and Rao [14] who prove that the odd cut and T-odd cut problems can be solved in polynomial time and those of Barahona and Conforti [2] who prove the same for even cuts. Grötschel et al. [9] improve their results both in that they only use $O(n^3)$ minimization oracle calls and in that their result applies to all triple families. The algorithm of Goemans and Ramakrishnan [6] mentioned above further improve both the efficiency and the scope of applicability of these even/odd mincut results.

1.5 New results and organization. We give algorithms that find the minimum of a submodular function f in a triple family over a ground set V with $|V| = n$. All these algorithms are faster than the Goemans–Ramakrishnan algorithm [6] by a factor of n . None of our algorithms requires *minimal* minimizers, just any minimizers of the function f . This might be an advantage for example for cuts in graphs in that arbitrary max-flow algorithms may be used. While we cannot get a speedup for the more general scenario of Goemans and Ramakrishnan [6] (parity families), we also sketch an algorithm (Section 4.4) that matches the asymptotic efficiency of their algorithm.

The main technique of our algorithms is a specific uncrossing procedure that we call *parity uncrossing*. All of our algorithms follow the generic framework of Section 2 by starting with a collection of (not necessarily odd) submodular minimizers and apply the parity uncrossing procedure of Section 2.1. As input, this procedure gets a collection of sets whose intersection is odd; in this collection it repeatedly picks sets and replaces them by either of the union or the intersection with smaller f -value. The procedure terminates when an odd set is found; these odd sets become the minimizer candidates.

We give four similar algorithms that minimize submodular functions over triple families depending on whether the empty set \emptyset or the ground set V are odd (belong to the family). We need such a distinction when minimizing submodular functions, since in general we want to avoid obtaining these sets as minimizers. For instance in case of even cardinality cuts in a graph $f(\emptyset) = f(V) = 0$ while $f(S) \geq 0$ for all other sets; the minimizer we get should be different from these two trivial ones.

Our first algorithm in Section 3 solves the simplest case when neither the empty set nor the ground set is in the family. We use $O(n)$ calls to a minimization oracle (max-flow computation) and another $O(n^2)$ to an evaluation oracle (one that tells the value of

$f(C)$ for a set C). This algorithm may for example find the minimum odd cardinality cut in $\tilde{O}(n^2m)$ time for $|V| = 2k$.

The next two algorithms drop the restriction that \emptyset and V must be even by keeping the asymptotic running time of the first one; however these algorithms may return both \emptyset and V as the odd minimizer of f . The second algorithm in Section 4.1 drops the restriction on the ground set V (the empty set still must be even); the third one in Section 4.2 drops both restrictions.

The fourth and strongest algorithm in Section 4.3 applies to all triple families and finds the odd minimizer of f different from \emptyset and V . The algorithm, just like that of Goemans and Ramakrishnan [6] for this case, makes $O(n)$ calls to the algorithm that may return \emptyset or V as minimizers. Thus we spend the time of $O(n^2)$ calls to a minimization oracle and another $O(n^3)$ to an evaluation oracle. This algorithm may for example find a minimum even cardinality cut in $\tilde{O}(n^3m)$ time.

2 The main algorithmic idea: Parity Uncrossing

Our generic submodular minimization algorithm over triple families (the *odd sets* as defined in Section 1.3) begins with a collection of certain minimizers of the submodular function f among all (not necessarily odd or even) subsets. Two earlier algorithms also follow this line: the algorithm of Goemans and Ramakrishnan [6] considers the $n(n-1)$ minimizers separating all possible pairs of the ground set $|V| = n$; and the algorithms for symmetric submodular functions such as in [5] arrange $n-1$ minimizers in a so-called Gomory–Hu tree [8] to select the odd minimizer.

We aim to achieve the same efficiency for minimization as in the second type of algorithms that use Gomory–Hu trees. Although Gomory–Hu trees do not exist in the asymmetric case (as pointed out by [3, 16]), we can still use the abstraction of *uncrossing* in the Gomory–Hu construction [8] to reduce the number of minimizations from $O(n^2)$ to $O(n)$ for odd minimization of asymmetric submodular functions.

Definition 1. Two sets X and Y *cross* if neither of $X \cap Y$, $X - Y$, $Y - X$ and $V - (X \cup Y)$ is empty. For two sets X and Y , *uncrossing* means that we replace one of X and Y with larger f -value by one of $X \cap Y$ and $X \cup Y$ with smaller f -value.

Note that submodularity $f(X \cap Y) + f(X \cup Y) \leq f(X) + f(Y)$ implies

$$\min\{f(X \cap Y), f(X \cup Y)\} \leq \max\{f(X), f(Y)\}$$

and thus uncrossing will not increase the minimum of $f(X)$ in a set system \mathcal{X} .

The key idea of our algorithm is *parity uncrossing*, a particular way of uncrossing that applies to asymmetric functions and can be related to the properties of triple families. Given that we start out with a set system \mathcal{X} , we replace elements X of \mathcal{X} one by one by odd sets (members of the triple family) Y with $f(Y) \leq f(X)$. All we require for this procedure is a subcollection $\mathcal{X}' \subseteq \mathcal{X}$ with $\bigcap\{X \in \mathcal{X}'\}$ odd. By using this odd set to guide uncrossing, we may ensure an odd set as the outcome of uncrossing.

Now we give Algorithm GENERIC-MINIMIZE-ODDSETS, the framework of minimization based on the parity uncrossing subroutine described in detail in the next subsection. In this algorithm we proceed by reducing the size of \mathcal{X} by using the odd sets

Y found by parity uncrossing. Given such a set Y , we will discard all $X \in \mathcal{X}$ with $f(X) \geq f(Y)$. In order to find Y , we require a subcollection of \mathcal{X} with odd intersection; the minimization algorithm terminates when there is none. Then our particular implementations will complete the task of odd minimization by using the candidate minimizer Y_0 and the remaining collection \mathcal{X} .

GENERIC-MINIMIZE-ODDSETS(\mathcal{X})

$\mathcal{Y} \leftarrow \emptyset$ \triangleright \mathcal{Y} contains the candidate odd sets that minimize f
while $\exists \mathcal{X}' \subset \mathcal{X}$ with $\bigcap \{X : X \in \mathcal{X}'\}$ odd **do**
 obtain an odd set Y with $f(Y) \leq \max\{f(X) : X \in \mathcal{X}'\}$
 $\mathcal{X} = \{X \in \mathcal{X} : f(X) < f(Y)\}$
 $\mathcal{Y} = \mathcal{Y} + Y$
output \mathcal{X} and $Y_0 \in \mathcal{Y}$ with $f(Y)$ minimum

2.1 Subroutine PARITY-UNCROSS. We introduce the technique of *parity uncrossing* that, given a collection \mathcal{X} with $\bigcap \{X \in \mathcal{X}\}$ odd, finds an odd set Y with $f(Y) \leq f(X)$ for some $X \in \mathcal{X}$. The family of odd sets must satisfy (*) but not necessarily (**). We give a divide-and-conquer parity uncrossing procedure that makes $O(n)$ calls to the *evaluation* oracle. Notice that in case of graph cuts the evaluation oracle is $\tilde{\Theta}(n)$ times faster than the minimization oracle, hence in order not to exceed the time for finding $O(n)$ minimum cuts, we may make a total of $O(n^2)$ evaluation calls in the minimization algorithm. Our procedure stays within this bound as long as the algorithm calls parity uncrossing $O(n)$ times. However we may not use a more straightforward non-divide-and-conquer approach for parity uncrossing since that would cost a total of $O(n^3)$ evaluation calls.

We start the description of Algorithm PARITY-UNCROSS with the easy cases. If we have a single set X_1 in the input, it must be odd by the requirement on the intersection and thus we may immediately return this $Y = X_1$. For two sets X_1, X_2 we proceed by selecting either X_1 or X_2 if one of them is odd. If this is not the case, by Property (*) $X_1 \cup X_2$ must be odd if (as assumed) $X_1 \cap X_2$ is odd. Thus we may select the odd set $X_1 \cup X_2$ or $X_1 \cap X_2$ with smaller f -value; by submodularity this value cannot be more than $\max\{f(X_1), f(X_2)\}$. Notice that here we do not require X and Y cross; in fact if $X \cap Y = \emptyset$, then \emptyset is odd and if $V - X \cup Y = \emptyset$, then V is odd. In general we want to avoid these two trivial sets as minimizers but we cannot do it right at this step.

For a general input $\mathcal{X} = \{X_1, \dots, X_k\}$ with $k > 2$ we use divide-and-conquer: we recurse on the two halves of the set \mathcal{X} ; then on the outputs Y_1 and Y_2 of the recursive calls we perform the steps for case $k = 2$. Unfortunately we rely on the fact that the intersection of \mathcal{X} is odd, but in a recursive call we may not expect this for either half of \mathcal{X} . Hence we must extend the input by a *restriction set* R ; instead of requiring certain sets to be odd, we only require their intersection with R be odd. Thus initially $R = V$ and at a general recursive call it is a set with $R \cap \bigcap_i X_i$ odd.

We are ready with the description of Algorithm PARITY-UNCROSS if we describe the selection of the restriction set R . For the first half $\{X_1, \dots, X_{\lceil k/2 \rceil}\}$ we may add

the natural choice R as the intersection of all remaining X_i currently not processed. However for the second call we may simplify the proof if, instead of taking the corresponding choice by exchanging the role of the two halves of the input $\{X_i\}$, we take R as the output Y_1 of the first recursive call. Notice that Y_1 has an odd intersection with the previous $R = X_{\lceil k/2 \rceil + 1} \cap \dots \cap X_k$ by the definition of parity uncrossing; this is exactly the requirement for the new $R = Y_1$ for the second recursive call. Finally the output Y_2 of this second call, again by the definition of parity uncrossing, has odd intersection with Y_1 and thus we may perform the steps of case $k = 2$ for Y_1 and Y_2 .

```

PARITY-UNCROSS( $X_1, \dots, X_k, R$ )
  if  $k = 1$  then
  1  return  $Y = X_1$ 
      $Y_1 \leftarrow \text{PARITY-UNCROSS}(X_1, \dots, X_{\lceil k/2 \rceil}, R \cap \bigcap_{i > \lceil k/2 \rceil} X_i)$ 
      $Y_2 \leftarrow \text{PARITY-UNCROSS}(X_{\lceil k/2 \rceil + 1}, \dots, X_k, R \cap Y_1)$ 
     if  $Y_i \cap R$  ( $i = 1$  or  $2$ ) is odd then
  2  return  $Y = Y_i$ 
     else
  3  return either  $Y = Y_1 \cup Y_2$  or  $Y = Y_1 \cap Y_2$  with smaller  $f$ -value

```

Lemma 1. *If $X_1 \cap \dots \cap X_k \cap R$ is odd, then the set Y returned by Algorithm PARITY-UNCROSS has $Y \cap R$ odd and $f(Y) \leq \max\{X_1, \dots, X_k\}$.*

Proof. By induction on the recursive calls. First of all we show that the recursive calls are valid, i.e. their input satisfies that

$$X_1 \cap \dots \cap X_{\lceil k/2 \rceil} \cap R \cap \bigcap_{i > \lceil k/2 \rceil} X_i \quad \text{and} \quad X_{\lceil k/2 \rceil + 1} \cap \dots \cap X_k \cap R \cap Y_1$$

are odd. For the first intersection this follows since we have $X_1 \cap \dots \cap X_k \cap R$ odd by the requirement on the input of Algorithm PARITY-UNCROSS. Next we apply the inductive hypothesis for the first recursive call to obtain $Y_1 \cap R \cap \bigcap_{i > \lceil k/2 \rceil} X_i$ odd. However this is exactly the requirement for the input of the second recursive call.

Now we show the first part of the claim stating $Y \cap R$ odd for the output of the algorithm. The claim is trivial for $k = 1$ (line 1) and for $Y_i \cap R$ odd (line 2). For the remaining case (line 3) we notice that the inductive hypothesis gives $Y_1 \cap Y_2 \cap R = (Y_1 \cap R) \cap (Y_2 \cap R)$ odd for the output of the second recursive call. Since we are not exiting in line 2, we must have $Y_i \cap R$ even for $i = 1, 2$. But then

$$R \cap (Y_1 \cup Y_2) = (R \cap Y_1) \cup (R \cap Y_2)$$

must also be odd by Property (*).

We prove the second part again separate for the three output types in lines 1, 2 and 3 by induction. The claim is trivial for $k = 1$ (line 1). For the output $Y = Y_1$ at line 2 we get

$$\begin{aligned} f(Y_1) &\leq \max\{f(X_1), \dots, f(X_{\lceil k/2 \rceil})\} \\ &\leq \max\{f(X_1), \dots, f(X_k)\} \end{aligned}$$

where the first inequality follows by induction. A similar proof can be obtained for $Y = Y_2$:

$$f(Y_2) \leq \max\{f(X_1), \dots, f(X_k)\}.$$

Finally if we exit at line 3,

$$\begin{aligned} \min\{f(Y_1 \cap Y_2), f(Y_1 \cup Y_2)\} &\leq \max\{f(Y_1), f(Y_2)\} \\ &\leq \max\{f(X_i) : i \leq k\} \end{aligned}$$

where the first inequality follows by submodularity while the second by the previous two inequalities for $f(Y_1)$ and $f(Y_2)$. \square

Lemma 2. *Algorithm PARITY-UNCROSS runs with $O(k)$ calls to the evaluation oracle.*

Proof. The number of calls to find $f(X)$ for $X \subset V$ satisfies the recurrence

$$T(k) = 2T(k/2) + O(1). \square$$

3 The simplest case: when \emptyset and V are not in the triple family

Our first minimization algorithm applies when the empty set and the ground set are even (not in the triple family). This assumption is not only necessary for ensuring an outcome different from the trivial minimizers \emptyset and V but also technically necessary for the correctness of our first algorithm below.

The algorithm proceeds as follows. First it computes a collection \mathcal{X} of $n - 1 = |V| - 1$ sets with the property that

for each pair $u, v \in V$ there exists an $X \in \mathcal{X}$ such that $f(X)$ is minimum over all sets X that contain exactly one of u and v .

In terms of cuts in graphs, X is the smaller of a minimum u - v or a minimum v - u cut. The collection \mathcal{X} can be computed by $2n - 2$ minimization oracle calls as proved in [4] or [1, Chapter 8.7]. Note that this fact is a weakening of the result of Gomory and Hu [8] since the collection \mathcal{X} cannot be cross-free in general.

Given the above collection \mathcal{X} , we proceed by discarding its members by Algorithm GENERIC-MINIMIZE-ODDSETS. If a discarded set X minimizes f over sets containing exactly one of u, v , then no odd set S with value less than $f(X)$ may separate u and v since $f(Y) \leq f(X) \leq f(S)$ for all sets S separating u and v . Hence we may contract u and v for the further steps, or instead we may consider the partition given by the remaining elements of \mathcal{X} defined below.

Definition 2. *For a set system \mathcal{X} , let $\mathcal{P}(\mathcal{X})$ be the partition of the ground set V defined with both $u, v \in P$ for some $P \in \mathcal{P}$ if and only if either $u, v \in X$ or $u, v \notin X$ for all $X \in \mathcal{X}$.*

The correctness of the algorithm immediately follows by the next two theorems. The first theorem will also give a way of finding, at each iteration of the algorithm, a subcollection of \mathcal{X} with odd intersection. The description of the algorithm is completed with this additional step in Section 3.2.

Theorem 1. *Let us consider a family of odd sets satisfying $(*)$, $(**)$ with \emptyset and V even. Let \mathcal{X} be an arbitrary collection of sets such that all non-empty subcollections $\mathcal{X}' \subseteq \mathcal{X}$ have $\bigcap\{X \in \mathcal{X}'\}$ even. Then $\mathcal{P}(\mathcal{X})$ contains no odd sets.*

Theorem 2. *Assume the family of odd sets satisfy $(*)$ with \emptyset even. Let us run Algorithm GENERIC-MINIMIZE-ODDSETS with a collection \mathcal{X} containing, for each pair $u, v \in V$, a minimizer of f over all sets X that contain exactly one of u and v . If $\mathcal{P}(\mathcal{X})$ contains no odd sets, then \mathcal{Y} contains the minimizer of f over the odd sets.*

3.1 Proofs. First we prove Theorem 2. The proof requires odd sets satisfy $(*)$ with \emptyset even; neither $(**)$ nor the fact that V is even need to be used. These facts are used only for Theorem 1 that we prove later via a sequence of lemmas. In these lemmas we explicitly show where we use the requirement that V is even; we in fact get a weaker result that also applies when V is odd.

Proof (Theorem 2). Let S be an odd minimizer of f . First we show that S subdivides at least one set $P \in \mathcal{P}(\mathcal{X})$. This follows since the partition consists exclusively of even sets. The union of disjoint even sets is even by the repeated use of Property $(*)$ and the fact that \emptyset is even. Thus the odd set S may not arise as the union of some sets in $\mathcal{P}(\mathcal{X})$.

Next consider two elements $u, v \in V$ with $u \in P \cap S$ and $v \in P - S$. The initial set system \mathcal{X} contains a set X minimizing f over all sets that contain exactly one of u and v . This set must, at some stage of the algorithm, be discarded from \mathcal{X} ; at the same time, a set Y is added to \mathcal{Y} with $f(Y) \leq f(X)$.

Now we are ready with the proof. Since S is a set containing exactly one of u and v , we must have $f(S) \geq f(X)$, thus $f(S) \geq f(Y)$. On the other hand the algorithm returns an odd set $Y_0 \in \mathcal{Y}$ with $f(Y_0) \leq f(Y)$. In conclusion the odd set Y_0 returned at termination minimizes f over odd sets since $f(Y_0) \leq f(S)$. \square

Lemma 3. *Consider an arbitrary classification of subsets of V as even and odd and an arbitrary set system \mathcal{X} . Let $P \in \mathcal{P}(\mathcal{X})$ be odd with the maximum number of sets $X \in \mathcal{X}$ with $P \subseteq X$. Then all $P' \in \mathcal{P}(\mathcal{X})$ with $P' \neq P$ and $P' \subseteq \bigcap\{X \in \mathcal{X} : P \subseteq X\}$ are even. Here the intersection of an empty collection $\{X \in \mathcal{X} : P \subseteq X\}$ is defined to be the entire V .*

Proof. Assume in contrary that some $P' \neq P$ as above is odd. Consider some $X_0 \in \mathcal{X}$ that separate P and P' , i.e. have $P \not\subseteq X_0$ and $P' \subseteq X_0$. Then P' is contained by one more set of \mathcal{X} than P , a contradiction. \square

Lemma 4. *Let odd sets satisfy $(*)$ and $(**)$. Let \emptyset be even. Let \mathcal{X} and P be as in Lemma 3. Then $\bigcap\{X \in \mathcal{X} : P \subseteq X\}$ is odd.*

Proof. Let $Z = \bigcap\{X \in \mathcal{X} : P \subseteq X\}$. By Lemma 3 all $P' \in \mathcal{P}(\mathcal{X})$ with $P' \neq P$, $P' \subset Z$ are even. By repeatedly applying $(*)$ we get that $\bigcup\{P' \in \mathcal{P}(\mathcal{X}) : P \subset Z, P' \neq P\} = Z - P$ is even. The proof is completed by applying $(**)$ to Z and $Z - P$. \square

The lemma almost implies Theorem 1: we may parity uncross $\{X \in \mathcal{X} : P \subseteq X\}$ whenever $P \in \mathcal{P}(\mathcal{X})$ is odd, provided the subcollection of \mathcal{X} is *non-empty*. However for the possible element $P_0 = V - \bigcup\{X \in \mathcal{X}\}$ of $\mathcal{P}(\mathcal{X})$ we intersect the empty subcollection; while that is V by our definition and it could be odd, we cannot proceed by parity uncrossing. The final lemma shows that this may happen only if V is odd.

Lemma 5. *Let odd sets satisfy (*) and (**). Let \emptyset and V be even. For a set system \mathcal{X} , let $P_0 = V - \bigcup\{X \in \mathcal{X}\}$. If all $P \neq P_0, P \in \mathcal{P}(\mathcal{X})$ are even, then so is P_0 .*

Proof. By repeatedly applying (*) we get that $\bigcup\{P \in \mathcal{P}(\mathcal{X}) : P \neq P_0\} = V - P_0$ is even. Then we are done by applying (**) to V and $V - P_0$. \square

3.2 Implementation. In Algorithm MINIMIZE-ODDSETS we give an efficient implementation for the algorithm of Section 3. Notice that currently we use an unspecified selection rule for \mathcal{X}' in Algorithm GENERIC-MINIMIZE-ODDSETS that could take exponential time. Our procedure below, in contrast, requires only $O(n^2)$ time in addition to parity uncrossing.

MINIMIZE-ODDSETS

$\mathcal{X} \leftarrow$ a family of sets that, for all $u, v \in V$ with $u \neq v$, contains a minimizer of $f(X)$ with exactly one of u and v in X
 $\mathcal{Y} \leftarrow \emptyset$
while $\exists P \in \mathcal{P}(\mathcal{X})$ odd **do**
 $P \leftarrow$ odd element $P \in \mathcal{P}(\mathcal{X})$ with $|\{X : P \subseteq X\}|$ maximum
 $Y \leftarrow$ PARITY-UNCROSS($\{X : P \subseteq X\}, V$)
 $\mathcal{X} = \{X \in \mathcal{X} : f(X) < f(Y)\}$
 $\mathcal{Y} = \mathcal{Y} + Y$
output $Y_0 \in \mathcal{Y}$ with $f(Y_0)$ minimum

In our implementation we use Lemma 3 by selecting $P \in \mathcal{P}(\mathcal{X})$ odd with the maximum number of sets in \mathcal{X} containing P ; these elements of \mathcal{X} will have odd intersection then. For this selection we must maintain the partition $\mathcal{P}(\mathcal{X})$ as well as counters for each member P of the partition with the number of sets in \mathcal{X} containing P .

First we build the initial partition $\mathcal{P}(\mathcal{X})$. We do this by adding sets X in the order of increasing value $f(X)$. After each addition we maintain the partition \mathcal{P} as a linked list; an update of the list as well as the counters takes $O(n)$ time for a set X .

The implementation of removals from \mathcal{X} is easy, given the way we build the initial partition. Since we increment \mathcal{X} with sets of increasing f -value and we always delete sets of value at least a certain $f(Y)$, we may perform deletion by simply moving back in history to the stage when the first set of f -value at least $f(Y)$ was added. The history may be preserved in the linked list of $\mathcal{P}(\mathcal{X})$ as follows: whenever a set X is inserted and it subdivides some $P \in \mathcal{P}$ into two sets P_1 and P_2 , we insert P_2 in the list as an immediate successor of P_1 . If we mark the link from P_1 to P_2 by set X , we may remove X from \mathcal{X} in $O(n)$ time by traversing the linked list once.

4 Algorithms

In this section we describe four algorithms based on the parity uncrossing technique. In two steps we extend the algorithm of Section 3 first to the case when V (Section 4.1), then to the case when both \emptyset and V belong to the parity family (Section 4.2). These algorithms may return both \emptyset and V as minimizer and thus they are useless for example in

the case of graph cuts. However their running time remains within the asymptotic bound of the algorithm of Section 3. The third algorithm is guaranteed to output a minimizer different from \emptyset and V for the price of using $O(n)$ times more steps (Section 4.3).

The final Section 4.4 gives us a way of applying parity uncrossing to parity families (or parity subfamilies of lattices), a more general problem class introduced by Goemans and Ramakrishnan [6] where odd sets satisfy only (*). Unfortunately our algorithm is $O(n)$ times slower than its counterpart for triple families and thus only achieves the same efficiency as that of Goemans and Ramakrishnan.

4.1 The case \emptyset even, V odd. We consider a triple family satisfying (*) and (**) where V belongs to the family (V is odd) while \emptyset does not (\emptyset even). Recall that in this case Algorithm MINIMIZE-ODDSETS may terminate with a collection \mathcal{X} with

$$P_0 = V - \bigcap \{X \in \mathcal{X}\} \text{ odd.}$$

We can easily modify the proof of Theorem 2 to get that if the odd minimizer S is not Y_0 returned by Algorithm MINIMIZE-ODDSETS, then it must arise as the union of certain members of partition $\mathcal{P}(\mathcal{X})$ and must contain P_0 .

We will continue minimization hence by finding the minimal odd set containing P_0 in a modified ground set with all members of partition $\mathcal{P}(\mathcal{X})$ contracted to a single vertex. In this ground set V' a single element $v \in V'$ corresponding to P_0 has $\{v\}$ odd, while all other single-point sets are even.

Now we apply Algorithm GENERIC-MINIMIZE-ODDSETS in the following implementation. We start with a different

$$\mathcal{X} = \{X_u : f(X_u) \text{ is minimum among all sets } X \text{ with } v \in X, u \notin X\}.$$

Now we call Algorithm PARITY-UNCROSS always with the *entire* \mathcal{X} . Hence the algorithm terminates with a collection \mathcal{X} such that $Z = \bigcap \{X \in \mathcal{X}\}$ is even. Notice that all sets containing Z are even since they arise as the union of Z and single-point sets other than $\{v\}$. The union of disjoint even sets is however even by (*) and the fact that \emptyset is even. We are thus done by the next lemma:

Lemma 6. *Let us run Algorithm GENERIC-MINIMIZE-ODDSETS with a family of odd sets satisfying (*), with $\mathcal{X} = \{X_u\}$ as above, and by always parity uncrossing over the entire remaining collection \mathcal{X} . Let $f(Y_0)$ be minimum in \mathcal{Y} and let $Z = \bigcap \mathcal{X}$ for the output \mathcal{X} and \mathcal{Y} . Then if an odd set S containing v has $f(S) < f(Y_0)$, then $S \supset Z$.*

Proof. By contradiction let $u \in Z - S$. Since $u \notin X_u$, set X_u must have been discarded from \mathcal{X} to obtain the intersection Z with $u \in Z$. Thus the output Y_0 of the algorithm has $f(Y_0) \leq f(X_u)$. On the other hand S has $v \in S$ and $u \notin S$; by the minimality of $f(X_u)$ we have $f(X_u) \leq f(S)$. We get the contradiction $f(Y_0) \leq f(S)$. \square

4.2 A general algorithm that may return \emptyset or V . The previous algorithms may give an incorrect result if \emptyset is odd: the proofs rely on the fact that the union of disjoint even sets is even and this fails if \emptyset is odd. We may, in case V is even, turn to the dual (either complement every set or exchange the notions of union and intersection) to obtain a correct algorithm. However we need additional steps if both \emptyset and V are odd.

Our algorithm builds on Lemma 6. First we make two subproblems by selecting a vertex v and finding the odd minimizer containing and not containing v separately. Since the second subproblem is the dual of the other, we only discuss the first one. Now by the lemma we may immediately get an even set Z such that the minimizer is either already found or else it contains Z . The second scenario can now be handled by the algorithm of Section 4.1 by considering a new ground set $V' = V - Z$ where set $X \subseteq V'$ is odd iff $X \cup Z$ is (originally) odd; over V' we define a new submodular function $f'(X) = f(X \cup Z)$. Since \emptyset is even in the new family, we may obtain the odd minimizer Y of f' . Set Y is precisely the odd minimizer of f among sets containing Z .

4.3 Minimization with sets \emptyset and V excluded. The final algorithm finds the minimum f -value odd set different from \emptyset and V even if \emptyset or V is in the triple family. In order to avoid the possible trivial minimizers \emptyset and V , we pay the price of calling the algorithm of the previous section $O(n)$ times. Note that the algorithm of Goemans and Ramakrishnan [6] also has this drawback and hence we are still faster by a factor of $O(n)$ in this case.

Our algorithm is based on a very simple reduction. As we saw in the previous section, we may restrict attention to sets containing an arbitrarily prescribed $v \in V$. Now we find the odd minimizer S of f containing v by $n - 1$ calls to the algorithm of Section 4.2 as follows. Since $S = V$ as minimizer is excluded, there exists $w \in V$ with $w \notin S$. For all $n - 1$ possible such w we proceed by finding the minimizer over $V' = V - \{v, w\}$ with $f'(X) = f(X + v)$ by the algorithm of Section 4.2. The result is an odd minimizer of f containing v but not containing w ; we take the minimum over all choices of w .

Note that in case of even cardinality T -cuts C , we may also find the minimizer with $C \cap T \neq \emptyset, T$ by selecting both vertices u and v from T .

4.4 Parity families. A parity family is defined as a collection of subsets of V satisfying only (*), i.e. if neither X nor Y belongs to the family, then $X \cap Y$ and $X \cup Y$ must either both belong to the family or both be outside the family. We may use the terminology of even and odd sets just like in the case of triple families.

For parity families Algorithm MINIMIZE-ODDSETS fails in Theorem 1 since parity families do not satisfy Property (**). Thus it may happen that the partition $\mathcal{P}(\mathcal{X})$ given by the f -minimizer sets \mathcal{X} contains an odd set P while, unlike in Lemma 4, the intersection $Z = \bigcap \{X \in \mathcal{X} : P \subseteq X\}$ is even. In this case the minimizer may be a set containing P but not the entire Z . We may apply Lemma 6 to find the minimizer for this case; however in the worst case we may have $\Omega(n)$ such odd P and use $\Omega(n^2)$ minimization oracle calls. In this algorithm and proof of correctness we never take complements of sets and thus we can also apply it to parity subfamilies of lattices.

5 Conclusion

We have given improved algorithms for the minimum odd or even cut problems and their generalizations. The algorithms achieve a factor n speedup over the previous best ones. While we believe that an algorithm with $O(n)$ flow computations is optimal for example for the directed minimum odd cut problem, we conjecture that some of the remaining open questions may have a positive answer:

- Can one solve the *undirected* minimum odd (even) cut problem by pipelining the $O(n)$ flow computations needed to a single one, in the flavor of the Hao–Orlin algorithm [12]?
- Can one optimize over parity families that are not triple families (cf. [6]) with only $O(n)$ minimization oracle calls (max-flows)?
- Can one improve the algorithm in the case \emptyset or V is odd (such as the minimum even cut problem)?

6 Acknowledgement

The second author would like to thank András Frank for fruitful discussions.

References

1. Ahuja, R.K., T.L. Magnanti and J.B. Orlin, *Network flows: Theory, Algorithms and Applications*. Prentice Hall (1993).
2. Barahona, F. and M. Conforti, A construction for binary matroids. *Discr. Math.* **66**:213–218 (1987)
3. Benczúr, A. A., Counterexamples for directed and node capacitated cut-trees. *Siam J. Comp.* **24**(3):505–510 (1995)
4. Cheng, C. K. and T. C. Hu, Ancestor trees for arbitrary multi-terminal cut functions. *Ann. Op. Res.* **33**:199–213 (1991)
5. Gabow, H. N., M. X. Goemans and D. P. Williamson, An efficient approximation algorithm for the survivable network design problem. *Proc. 3rd IPCO*, pp. 57–74 (1993)
6. Goemans, M. X. and V. S. Ramakrishnan, Minimizing submodular functions over families of sets. *Combinatorica* **15**(4):499–513 (1995)
7. Goemans, M. X. and D. P. Williamson, The Primal-Dual Method for Approximation Algorithms and its Application to Network Design Problems, Chapter 4 in *Approximation Algorithms*, D. Hochbaum, Ed. (1997)
8. Gomory, R. E. and T. C. Hu, Multi-terminal network flows, *J. SIAM* **9**:551–570 (1961)
9. Grötschel, M., L. Lovász and A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* **1**:169–197 (1981)
10. Grötschel, M., L. Lovász and A. Schrijver, Corrigendum to our paper The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* **4**:291–295 (1984)
11. Grötschel, M., L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, Berlin, 1988.
12. Hao, J. and J. B. Orlin, A faster algorithm for finding the minimum cut in a graph, *Proc. of 3rd SODA*, pp. 165–174 (1992)
13. Lovász, L., Submodular functions and convexity, in: A. Bachem, M. Grötschel and B. Korte (eds.), *Mathematical Programming: The State of the Art, Bonn, 1982*. Springer, Berlin, pp. 235–257 (1983)
14. Padberg, M. W. and M. R. Rao, Odd minimum cut-sets and b -matchings. *Math. Op. Res.* **7**:67–80 (1982)
15. Picard, J. C. and M. Queyranne. Selected applications of minimum cuts in networks, *I.N.F.O.R.: Canadian Journal of Operations Research and Information Processing* **20**:394–422 (1982)
16. Rizzi, R., Excluding a simple good pair approach to directed cuts. *Graphs and Combinatorics*, to appear