

Bounds on Double-Sided Myopic Algorithms for Unconstrained Non-monotone Submodular Maximization

Norman Huang
University of Toronto
huangyum@cs.toronto.edu

Allan Borodin
University of Toronto
bor@cs.toronto.edu

April 24, 2014

Abstract

Unconstrained submodular maximization captures many NP-hard combinatorial optimization problems, including Max-Cut, Max-Di-Cut, and variants of facility location problems. Recently, Buchbinder *et al.* [8] presented a surprisingly simple linear time randomized *greedy-like* online algorithm that achieves a constant approximation ratio of $\frac{1}{2}$, matching optimally the hardness result of Feige *et al.* [19]. Motivated by the algorithm of Buchbinder *et al.*, we introduce a precise algorithmic model called *double-sided myopic algorithms*. We show that while the algorithm of Buchbinder *et al.* can be realized as a randomized online double-sided myopic algorithm, no such deterministic algorithm, even with adaptive ordering, can achieve the same approximation ratio. With respect to the Max-Di-Cut problem, we relate the Buchbinder *et al.* algorithm and our myopic framework to the online algorithm and inapproximation of Bar-Noy and Lampis [6].

1 Introduction

Submodularity emerges in natural settings such as economics, algorithmic game theory, and operations research; many combinatorial optimization problems can be abstracted as the maximization/minimization of submodular functions. A canonical example of such is $f(S) = \sum_{i \in S, j \notin S} w_{ij}$, the cut of a graph with edge weights w_{ij} , of the vertex set S . As with the special case of Min-Cut, the general problem of submodular minimization is solvable in polynomial time [24, 13, 28, 41].

Maximizing a submodular function, on the other hand, generalizes NP-hard problems such as Max-Cut [23], Max-Di-Cut [25, 3, 23], Maximum-Coverage [27, 16], expected influence in a social network [30] and facility location problems [12, 11]. Appropriately, this problem tends to be approached under the context of approximation heuristics in the literature. For monotone submodular functions, maximization under a cardinality constraint can be achieved by the greedy algorithm with an approximation ratio of $(1 - \frac{1}{e})$ [35], which is in fact optimal in the *value oracle model* [34]. The same approximation ratio is obtainable for the more general matroid constraints [9, 21], as well as knapsack constraints [42, 32].

We limit our discussion to unconstrained non-monotone submodular maximization — typical examples of which include Max-Cut and Max-Di-Cut in graphs and hypergraphs. Note that solving these problems as general submodular maximization may not yield the best approximation ratio. In fact, Goemans and Williamson [23] used semidefinite programming to approximate Max-Cut within 0.878, and Max-Di-Cut within 0.796. The approximation for Max-Di-Cut was later improved to 0.859 by Feige and Goemans [17] and to the currently best known ratio of 0.874 by Lewin, Livnat, and Zwick [33]. Trevisan [43] showed that $\frac{1}{2}$ approximation for Max-Di-Cut can also be achieved with linear programming. On the other hand, an approximation ratio of $\frac{1}{2}$ for the general unconstrained non-monotone submodular maximization problem is optimal unless exponentially many value oracle queries are made [19, 45]. In the former case, f is known to have a succinct representation (i.e. f is completely revealed once the algorithm queries every edge weight); while in general, an explicit representation of f might be exponential in the size of the ground set. On the other hand, instances of Max-Cut and Max-Di-Cut are convenient in establishing lower bounds for the general submodular maximization problem.

Recently, linear time (linear in counting one step per oracle call) *double-sided greedy* algorithms were developed by Buchbinder *et al.* [8] for unconstrained non-monotone submodular maximization. The deterministic version of their algorithm, stated formally in Algorithm 1, achieves an approximation ratio of $\frac{1}{3}$, while the randomized version achieves $\frac{1}{2}$ in expectation - improving upon the $\frac{2}{5}$ randomized local-search approach in [19], and the 0.42 simulated-annealing technique in [22, 20], in terms of approximation ratio, time complexity and arguably, algorithmic simplicity. While the hardness result of Feige *et al.* [19] implies optimality of the randomized algorithm, the gap between the deterministic and randomized variants remains an open problem. More specifically, is there any de-randomization that would preserve both the greedy aspect of the algorithm as well as the approximation?

To address this question, we adapt the framework of priority algorithms of Borodin *et al.* [7], a model for *greedy-like* algorithms that has since been used in studying the limits of certain optimization algorithms [40], graph algorithms [15], randomized greedy algorithms [4], and dynamic programming [1]. The idea is to derive information-theoretic lower bounds for entire *classes* of greedy-like algorithms using adversarial arguments similar to those employed in online competitive analysis. In our case, we define a *double-sided myopic algorithms* framework, and show that no such algorithm in the deterministic setting can de-randomize the Buchbinder *et al.* $\frac{1}{2}$ -ratio double-sided greedy algorithm.

1.1 Related Work

Our motivation is the double-sided greedy algorithms of Buchbinder *et al.* that we wish to formalize. Independently, Bar-Noy and Lampis [6] gave a $\frac{1}{3}$ deterministic online greedy algorithm for the Max-Di-Cut problem matching the deterministic approximation obtained by Buchbinder *et al.* for all unconstrained non-monotone submodular maximization problems. Interestingly, their algorithm is shown to be the de-randomization of the simple $\frac{1}{4}$ random-cut algorithm! Bar-Noy and Lampis give an improved $\frac{2}{3\sqrt{3}}$ approx-

imation for Max-Di-Cut when restricted to DAGs. Furthermore, they provide a precise online model with respect to which this approximation bound is essentially optimal.

In addition to the naive random-cut algorithm (and its de-randomization), there are a number of combinatorial algorithms for the Max-Di-Cut problem. Alimonti [2] gives a $\frac{2}{5}$ non-oblivious local search algorithm. Halperin and Zwick [25] also give a linear time randomized $\frac{9}{20}$ -approximation algorithm, as well as a bipartite-matching based algorithm with $\frac{1}{2}$ -approximation. Based on a novel LP formulation, a deterministic algorithm due to Datar *et al.* [14] achieves a $\frac{4}{9}$ -approximation for Max-Di-Cut. On the other hand, little is known about the limits of combinatorial approaches to Max-Di-Cut. In addition to the online inapproximation by Bar-Noy and Lampis, Feige and Joseph [18] give a randomized *oblivious algorithm*¹ (where only the total in/out weights of a vertex is given) that achieves a 0.483 ratio, but show that no oblivious algorithm can do better than 0.4899. In Section 6, we discuss the relation of oblivious algorithms to our work. Finally, we note that beyond combinatorial algorithms, Håstad [26] gives a $\frac{11}{12}$ inapproximability bound for Max-Di-Cut under the assumption that $NP \neq P$; whereas for Max-Cut, Khot *et al.* [31] show that the ratio of 0.878 is optimal under the Unique Games Conjecture.

Another relevant class of problems is submodular Max-Sat (in which the objective function is monotone submodular). Azar *et al.* [5] provide a randomized online algorithm that achieves $\frac{2}{3}$ -approximation, which is tight under their data model. They also demonstrate via a simple reduction the equivalence between submodular Max-Sat and monotone submodular maximization subject to binary partition matroids constraint. In fact, by incorporating the bipartition constraint, Buchbinder *et al.* show that their $\frac{1}{2}$ randomized double-sided greedy algorithm can be readily extended to a $\frac{3}{4}$ -approximation algorithm for submodular Max-Sat. While the $\frac{3}{4}$ ratio was already achievable by randomized algorithms [38, 44], the double-sided algorithm generalizes to submodular Max-Sat, all the while entailing a much simpler analysis (see Poloczek *et al.* [39] for details). On the other hand, Poloczek [37] shows that for Max-Sat, no deterministic adaptive priority algorithm (in a more general input model than in Azar *et al.*) can achieve an approximation ratio of 0.729. This rules out the possibility of de-randomizing the above $\frac{3}{4}$ algorithms using any greedy algorithm.² This stands in contrast to the fact that the naive randomized algorithm can be de-randomized (by the method of conditional expectations), and as shown by Yannakakis [46] becomes Johnson’s [29] deterministic algorithm. Chen *et al.* [10] show that Johnson’s algorithm is a $\frac{2}{3}$ approximation for Max-Sat.

Independent of our work, Paul, Poloczek and Williamson [36] have very recently derived a number of deterministic algorithms, and deterministic and randomized inapproximations for Max-Di-Cut with respect to the priority algorithm framework.

¹Feige [18] uses the term oblivious in a different sense than Alimonti [2] who uses the term to indicate that in each iteration of the local-search, the algorithm uses an auxiliary function rather than the given objective function.

²The inapproximation result of Poloczek also applies to submodular Max-Sat, as submodularity encompasses all modular (linear) functions.

1.2 Basic Definitions

A set function $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}$ is *submodular* if for any $S, T \subseteq \mathcal{N}$,

$$f(S \cup T) + f(S \cap T) \leq f(S) + f(T).$$

We say that f is *monotone* if $f(S) \leq f(T)$ for all $S \subseteq T \subseteq \mathcal{N}$, and *non-monotone* otherwise. An equivalent, and perhaps more intuitive definition of submodular functions captures the principle of *diminishing returns*: $f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T)$, whenever $S \subseteq T$ and $x \in \mathcal{N} \setminus T$.

In the unconstrained non-monotone submodular maximization problem, we are given a finite subset \mathcal{N} and the goal is to find a subset $S \subseteq \mathcal{N}$ (where \mathcal{N} is finite) so as to maximize $f(S)$ for a specified submodular function f . In general, since the specification of f requires knowing its value on all possible subsets, f is accessed via a value oracle which given $X \subseteq \mathcal{N}$, returns $f(X)$. We state the deterministic version of the Buchbinder *et al.* double greedy algorithm, which approximates this problem with $\frac{1}{3}$ guarantee, in Algorithm 1.

Algorithm 1 DeterministicUSM(f, \mathcal{N})

```

1:  $S_0 \leftarrow \emptyset, T_0 \leftarrow \mathcal{N}$ 
2: for  $i = 1$  to  $n$  do
3:    $a_i \leftarrow f(S_{i-1} \cup \{u_i\}) - f(S_{i-1})$ 
4:    $b_i \leftarrow f(T_{i-1} \setminus \{u_i\}) - f(T_{i-1})$ 
5:   if  $a_i \geq b_i$  then
6:      $S_i \leftarrow S_{i-1} \cup \{u_i\}, T_i \leftarrow T_{i-1}$ 
7:   else
8:      $T_i \leftarrow T_{i-1} \setminus \{u_i\}, S_i \leftarrow S_{i-1}$ 
9:   end if
10: end for
11: return  $S_n$ 

```

For some explicitly defined submodular functions, such as Max-Cut and Max-Di-Cut, we are given as input an edge weighted graph (or directed graph) $G = (V, E, w)$. Here we interpret the ground set \mathcal{N} to be vertex set V , and $f(X) = \sum_{u \in X, v \in V \setminus X} w(u, v)$ to be the cut function, which can be assumed to be computed at unit cost. But, of course, in such an explicitly defined problem, if an algorithm is given the edge weights then it may deduce the complete mapping of f without value oracle calls. For online computations, we usually assume that the graph is revealed one vertex at a time, in the sense that the revealed vertex specifies the edges and their weights to all previously revealed vertices (while the number of vertices may be given a priori). As argued, by Bar-Noy and Lampis, for cut problems, the revealed vertex must also give global information about each node, namely the total weight of in-edges and the total weight of out-edges.

1.3 Our Contribution

We introduce a formalization of *double-sided myopic algorithms* - an adaptation of the priority framework under a *restricted* value oracle model. To make this precise, we will introduce three types of *relevant oracle queries*. We show that the double-sided greedy algorithms of Buchbinder *et al.* can be realized as online double-sided myopic algorithms. Moreover, our framework also captures the online algorithm of Bar-Noy and Lampis for Max-Di-Cut. Even our most restrictive model (see query Q-Type 1 in Section 2.2) allows for plausible ways to extend these algorithms. As our main contribution, we establish the following lower bounds for deterministic algorithms under this framework with respect to the stronger Q-Types as defined in Section 2.2.

Theorem 1.1. *No deterministic online double-sided myopic algorithm (with respect to our strongest oracle model Q-Type 3) can achieve a competitive ratio of $\frac{2}{3\sqrt{3}} + \epsilon \approx 0.385 + \epsilon$ for any $\epsilon > 0$ for the unconstrained non-monotone submodular maximization problem.*

Theorem 1.1 is obtained by directly applying the hardness result of Bar-Noy and Lampis in Max-Di-Cut [6], which, to the best of our knowledge, has not been studied in the context of the Buchbinder *et al.* double greedy algorithm. We also show that the deterministic algorithm with $\frac{1}{3}$ approximation ratio guarantee for Max-Di-Cut by Bar-Noy and Lampis is in fact an instantiation of the double-sided greedy algorithm when the submodular function f is the directed cut function.

Extending to the class of (deterministic) fixed and adaptive priority algorithms in Theorem 1.2 and 1.4, we construct submodular functions and corresponding adversarial strategies that would force an inapproximability ratio strictly less than $\frac{1}{2}$. In terms of oracle restrictiveness, our inapproximation holds for the *already attained partial solution query* model (Q-Type 2), which is more powerful than what is sufficient to achieve the $\frac{1}{2}$ ratio by the online randomized greedy algorithm (Q-Type 1). A comprehensive description of the different oracle models will be covered in Section 2.2. Our proof is computer assisted, in that the objective function value for each possible subset is explicitly computed through linear programming.

Theorem 1.2. *There exists a problem instance such that no fixed priority double-sided myopic algorithm with respect to oracle model Q-Type 2 can achieve an approximation ratio better than 0.428.*

Theorem 1.3. *There exists a problem instance such that no fixed priority double-sided myopic algorithm with respect to oracle model Q-Type 3 can achieve an approximation ratio better than 0.450.*

Theorem 1.4. *There exists a problem instance such that no adaptive priority double-sided myopic algorithm with respect to oracle model Q-Type 2 can achieve an approximation ratio better than 0.432.*

2 The Double-Sided Myopic Algorithms Framework

By integrating a *restricted* value oracle model in a priority framework, we propose a general class of *double-sided myopic algorithms* that captures the Buchbinder *et al.* double-sided greedy algorithm. We rephrase the double-sided procedure as a single-sided sweep, but with access to a pair of complementary objective functions - as opposed to simultaneously evolving a bottom-up and a top-down solution. This interpretation admits a myopic behavior common to most greedy algorithms; and we show how this facilitates an adaptation of a priority-like framework. To make this formalization precise, we must specify how information about input items is represented and accessed. The generality of the unconstrained submodular maximization problem raises some representational issues leading to a number of different input presentations. To address these issues, we employ a *marginal value* representation that is compatible with both the value oracle model and the priority framework.

On the one hand, a precise description of a data item is necessary in determining an ordering of the input set, as well as in quantifying the availability of information in the decision step. On the other hand, the value oracle model measures complexity in terms of information access. An apparent incompatibility arises when an exact description of a data item can trivialize query complexity - as in the case of the *marginal value* representation, where exponentially many queries are needed to fully describe an item when f is an arbitrary submodular function. For this reason, we propose a hierarchy of oracle restrictions that categorizes the concept of myopic *short-sightedness*, while preserving the fundamental characteristics of the priority framework. This consequently gives rise to a hierarchy of algorithmic models, which will be actualized once we define the algorithm's *internal memory*. We conclude this section by expressing the Buchbinder *et al.* double-sided greedy algorithm under the double-sided myopic framework.

2.1 Value Oracle and the Marginal Value Representation

For succinctly encoded problems, the objective is a function of an explicitly given collection of input item attributes; that is, weighted adjacencies in Max-Cut and Max-Di-Cut, and distances and opening costs in facility location problems. In contrast, in general submodular maximization problems, other than labels, the elements of \mathcal{N} by themselves do not convey pertinent information. Therefore we interpret \mathcal{N} as a fixed ground set, and the input instance as the objective function itself, drawn from the family $\mathcal{F} = \{f | f : 2^{\mathcal{N}} \rightarrow \mathbb{R}\}$ of all submodular set functions over \mathcal{N} . To avoid having an exponentially sized input, we employ a *value oracle* as an intermediary between the algorithm and the input function. That is, given a query $S \subseteq \mathcal{N}$, the value oracle will answer the question: "What is the value of $f(S)$?" By an abuse of notation, we will interchangeably refer to f as the objective function (*i.e.* when referring to $f(S)$ as a real number) and as the value oracle (*i.e.* when an algorithm submits a query to f).

We also introduce for notational convenience a complementary oracle \bar{f} , such that $\bar{f}(X) = f(\mathcal{N} \setminus X)$, for input set \mathcal{N} . This allows us to express the double-sided myopic algorithm similar to the priority setting in [7], where the solution set X is constructed

item by item, using only locally available information. In other words, the introduction of \bar{f} allows access to $f(\mathcal{N} \setminus X)$ using X (which is composed of items that have already been considered) as query argument, instead of $\mathcal{N} \setminus X$. The **double-sidedness** of the framework follows in the sense that f and \bar{f} can be simultaneously accessed.

We wish to model *greedy-like* algorithms that process the problem instance item by item. But what is an item when considering an arbitrary submodular function? While the natural choice is that an item is an element of \mathcal{N} (to include or not include in the solution S), for arbitrary submodular maximization the input is a function f (or more precisely, an oracle interface of f) and thus the notion of a *data item* becomes somewhat problematic. To address this issue, we propose the *marginal value representation*, in which f is instilled into the elements of \mathcal{N} . Specifically, we could describe a data item as an element $u \in \mathcal{N}$, plus a list of marginal differences $\rho(u|S) = f(S \cup \{u\}) - f(S)$, and $\bar{\rho}(u|S) = \bar{f}(S \cup \{u\}) - \bar{f}(S)$ for every subset $S \subseteq \mathcal{N}$. The impracticality in using a complete representation in this form is evident as the space required is exponential in $|\mathcal{N}|$. Furthermore, such a complete marginal representation would lead to a trivial optimal greedy algorithm by adaptively choosing the next item to be one that is included in an optimal solution (given what has already been accepted). Therefore we assume that an oracle query must be made by the algorithm in order to access each marginal value. In terms of inapproximability arguments, if we impose certain constraints on what oracle queries are allowed during the computation, then the model allows us to justify when input items are indistinguishable. In fact, our inapproximation results will not be based on bounding of the number of oracle queries but rather will be imposed by the restricted myopic nature of the algorithm.

2.2 Classes of Relevant Oracle Queries

The **myopic** condition of our framework is imposed both by the nature of the ordering in the priority model, which we describe later on, as well as by restricting the algorithm to make only certain types of oracle queries that are *relevant*. To avoid ambiguity, we emphasize that the value oracles are given in terms of f and \bar{f} , with ρ and $\bar{\rho}$ being used purely for notational simplicity. That is, when we say that the algorithm learns the value $\rho(u|S)$, we assume that it queries both $f(S \cup \{u\})$ and $f(S)$. At iteration i , define (respectively) X_{i-1} and Y_{i-1} to be the currently accepted and rejected sets, and u_i to be the next item considered by the algorithm. We now introduce three models of relevant oracle queries in hierarchical ordering, starting with the most restrictive:

Next attainable partial solution query (Q-Type 1)

The algorithm is permitted to only query $f(X_{i-1} \cup \{u_i\})$ and $\bar{f}(Y_{i-1} \cup \{u_i\})$, corresponding to the values of the next possible partial solution. In terms of the data item model, this is equivalent to learning $\rho(u_i|X_{i-1})$ and $\bar{\rho}(u_i|Y_{i-1})$ for the item u_i .

We note that in this and in all our models, we can then use this information in any way. For example, the deterministic algorithm of Buchbinder *et al.* greedily

chooses to add u_i to X_{i-1} if $\rho(u_i|X_{i-1}) = a \geq b = \bar{\rho}(u_i|Y_{i-1})$. In our model, the decision about u_i can be any (even non-computable) function of a and b and the “history” of the algorithm thus far.

Already attained partial solutions query (Q-Type 2)

In this model we allow queries of the form $\rho(u_i|X_j)$ and $\bar{\rho}(u_i|Y_j)$ for all $j < i$.

All subsets query (Q-Type 3)

By making an all subsets query on u_i , the algorithm learns the marginal gains $\rho(u_i|S)$ and $\bar{\rho}(u_i|S)$ for every $S \subseteq X_{i-1} \cup Y_{i-1}$. Essentially, the algorithm is given full disclosure of the submodular function f and \bar{f} over the set of currently revealed items. Note that in this very general model, the algorithm can potentially query exponentially many sets so that in principle such algorithms are not subject to the $\frac{1}{2}$ inapproximation result of Feige et al [18].

Finally, observe that the above classes are ordered by inclusion. That is, if $Q^i(\mathcal{N})$ is the set of all queries on input \mathcal{N} permitted under query type i , then $Q^i(\mathcal{N}) \subseteq Q^{i+1}(\mathcal{N})$.

2.3 Internal Memory or History

We define an algorithm’s *internal memory* or *history* as a record of the following:

- All previously considered items and the decisions made for these items.
- The outcomes of all previous relevant query results.
- Anything that can be deduced from all previously considered items, decisions and relevant queries. That is, in the priority framework (section 2.4), the order in which the items are considered will determine that certain items in \mathcal{N} *cannot* take on certain marginal values. In other words, the algorithm may rule out from \mathcal{F} all submodular functions that would contradict the observed ordering.

Recalling the definition of X_{i-1} and Y_{i-1} , let $\mathcal{N}_{i-1} = X_{i-1} \cup Y_{i-1}$ be the set of all previously seen (and decided upon) items at the start of iteration i . Let u_i be the next element, and allow the algorithm to perform any possible relevant queries. We summarize the algorithm’s internal memory under each query restriction type:

Q-Type 1 myopic model

An algorithm with Q-Type 1 oracle access has the record of:

- The decision made for every $u \in \mathcal{N}_{i-1}$.
- $\rho(u_j|X_{j-1})$ and $\bar{\rho}(u_j|Y_{j-1})$ for $0 < j \leq i$.

Q-Type 2 myopic model

An algorithm with Q-Type 2 oracle access has the record of:

- The decision made for every $u \in \mathcal{N}_{i-1}$.
- $\rho(u_j|X_k)$ and $\bar{\rho}(u_j|Y_k)$ for $0 \leq k < j \leq i$.

Q-Type 3 myopic model

An algorithm with Q-Type 3 oracle access has the record of:

- The decision made for every $u \in \mathcal{N}_{i-1}$.
- $\rho(u|S)$ and $\bar{\rho}(u|S)$ for all $u \in \mathcal{N}_i$ and all $S \subseteq \mathcal{N}_i$.

2.4 Priority Models

The other aspect of the priority framework that we need to specify is the order in which input items are considered. We present a high-level description of the generic templates of double-sided myopic algorithms. As in the priority framework, we categorize double-sided myopic algorithms into the following subclasses: **online**, **fixed** priority, and **adaptive** priority. For all templates, the decision step remains the same. Namely, based on the history (which depends on the particular value oracle model) of previous relevant queries, and relevant queries corresponding to current item being considered, the algorithm makes an irrevocable decision for the current item. Let \mathcal{N} be the input set, whose length n is the only information that is initially accessible (*i.e.* before any queries are made) to the algorithm. We note that the transparency of the input length n allows us to capture a broader and potentially more powerful class of algorithms compared to that of the original priority framework. Or more precisely, it prevents an adversary from abruptly ending a computation freeing an algorithm from this concern.

An online double-sided myopic algorithm conforms to the standard template of an online algorithm:

Online 2-Sided Myopic Algorithm

while not empty(\mathcal{N})

next := lowest index (determined by adversary) of items remaining in \mathcal{N}

Relevant Query: Perform any set of relevant queries and update the *internal memory*

Decision: As a function of the *internal memory*, irrevocably accept or reject u_{next} , and remove u_{next} from \mathcal{N}

A fixed priority algorithm has some limited ability to determine the ordering of the input items. Namely, such an algorithm specifies an injective priority function $\pi : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, such that the item that minimizes π corresponds to the item of highest priority. In the generality of submodular function maximization, lacking any other information, the priority of an input u is determined as a function of u 's marginal gains in f and \bar{f}

with respect to the empty set. The item $u_{next} \in \mathcal{N}$ which minimizes π is then given to the algorithm. Our inapproximations also hold if the algorithm is also (say) given the maximum value of any $f(\{u\})$. In such a case, the priority can be a more complex function of the precise values of these marginals. We emphasize that π is determined before the algorithm makes any oracle queries, and cannot be changed. The structure of a fixed priority algorithm is as follows:

Fixed Priority 2-Sided Myopic Algorithm

Ordering: Specify a priority function $\pi : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$

while not empty(\mathcal{N})

$next :=$ index i of the item in \mathcal{N} that minimizes $\pi(\rho(u_i|\emptyset), \bar{\rho}(u_i|\emptyset))$

Relevant Query: Using u_{next} as the next item, perform any set of relevant queries and update the *internal memory*

Decision: As a function of the *internal memory*, irrevocably accept or reject u_{next} , remove u_{next} from \mathcal{N} and update internal memory.

In the most general class of adaptive priority algorithms, a new ordering function may be specified after each item is processed. Here we calculate the priority of an item u by extending π to take as input all marginal differences for u permissible in the chosen query model. More precisely, the algorithm does not inquire the marginals for all items currently in \mathcal{N} , but is simply given the item of highest priority u , namely $u = \operatorname{argmin}_v[\pi(Q(v))]$. We define $Q(u)$ to be the vector of u 's marginal gains accessible under the appropriate relevant query model. Observe that in Q-type 2 and 3, the length of Q increases with the iterations.

Adaptive Priority 2-Sided Myopic Algorithm

while not empty(\mathcal{N})

Ordering: Based on the *internal memory*, specify a priority function π

$next :=$ index i of the item in \mathcal{N} that minimizes $\pi(Q(u_i))$

Relevant Query: Using u_{next} as the next item, perform any set of relevant queries and update the *internal memory*

Decision: As a function of the *internal memory*, irrevocably accept or reject u_{next} , remove u_{next} from \mathcal{N} and update internal memory.

We remind the reader that for the fixed and adaptive priority models, updating the internal memory also means deducing that certain marginal descriptions cannot exist and applying relevant queries (for the given Q -type) to obtain additional information.

In particular, if $u_{next} \in \mathcal{N}$ is the item with the minimum π value, then any item u_j with $\pi(Q(u_j)) < \pi(Q(u_{next}))$ cannot appear later. Knowledge of what items (in terms of marginal representation) cannot be in \mathcal{N} can be assumed to be part of the internal memory.

Finally, we note that all conditions mentioned here are devoid of complexity assumptions, and thus our inapproximability arguments result from information theoretic arguments. That is, the complexity or even computability of the ordering and decision steps is arbitrary, and there is no limitation on the size of the memory.

2.5 Remodeling the Buchbinder *et al.* Double-Sided Greedy Algorithm

Claim 2.1. *The deterministic double-sided greedy algorithm in [8] can be modeled by Algorithm 2, a Q-Type 1 online double-sided myopic algorithm.*

Proof. To prove this claim, we relabel the variables and function calls on the syntactic level without modifying the algorithmic behavior, until we obtain a description that complies with the online myopic model. First we recall the formal description of the deterministic double-sided greedy algorithm (called *DeterministicUSM*) in Algorithm 1.

Define the variables $X_i = S_i$, $Y_i = \mathcal{N} \setminus T_i$, and rewrite Algorithm 1 in terms of X_i and Y_i . To establish an online setting, we assume a predetermined ordering over \mathcal{N} . Furthermore, as argued in section 2.1, we will replace \mathcal{N} by \bar{f} as input parameter in order to demonstrate the algorithm's item-by-item behavior. This is possible in Line 1, since $X_0 = Y_0 = \emptyset$ by definition. In the later steps, recovering S_i from X_i is trivial, but $T_i = \mathcal{N} \setminus Y_i$ would require access to \mathcal{N} . To resolve this problem, we make use of the complementary value oracle \bar{f} , with the following equalities

$$T_{i-1} \setminus \{u_i\} = (\mathcal{N} \setminus Y_i) \setminus \{u_i\} = \mathcal{N} \setminus (Y_i \cup \{u_i\}) \quad (1)$$

$$\begin{aligned} f(T_{i-1} \setminus \{u_i\}) - f(T_{i-1}) &= f(\mathcal{N} \setminus (Y_i \cup \{u_i\})) - f(\mathcal{N} \setminus Y_i) \\ &= \bar{f}(Y_i \cup \{u_i\}) - \bar{f}(Y_i) \end{aligned} \quad (2)$$

In this sense, we can replace \mathcal{N} by \bar{f} in the input parameter, since the necessary information about \mathcal{N} is implicitly encoded in the function \bar{f} . Due to the absence of a predefined item ordering, we assume the items are labeled by an adversary. We emphasize that the ability to conceal n is consequential only in the online model, and that our lower bounds for fixed and adaptive order algorithms hold even if n is revealed to the algorithm.

Combining these ideas, we describe the double-sided greedy algorithm as an online double-sided myopic algorithm in Algorithm 2 (*OnlineMyopic*). Finally, observe that the value oracle access in Line 3 and 4 conform to the specifications imposed by Q-Type 1.

□

Claim 2.2. *The randomized double-sided greedy algorithm in [8] can be modeled by a random-choice Q-Type 1 online double-sided myopic algorithm.*

Algorithm 2 OnlineMyopic(f, \bar{f})

```
1:  $X_0 \leftarrow \emptyset, Y_0 \leftarrow \emptyset$ 
2: for  $i = 1$  to  $n$  do
3:    $a_i \leftarrow f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})$ 
4:    $b_i \leftarrow \bar{f}(Y_{i-1} \cup \{u_i\}) - \bar{f}(Y_{i-1})$ 
5:   if  $a_i \geq b_i$  then
6:      $X_i \leftarrow X_{i-1} \cup \{u_i\}, Y_i \leftarrow Y_{i-1}$ 
7:   else
8:      $Y_i \leftarrow Y_{i-1} \cup \{u_i\}, X_i \leftarrow X_{i-1}$ 
9:   end if
10: end for
11: return  $X_n$ 
```

Proof. Using the same reduction from Claim 2.1, the proof follows identically. \square

3 A $\frac{2}{3\sqrt{3}}$ Inapproximation for the Online Case

Given a digraph $G(V, E)$ with non-negatively weighted edges $w(e)$, the cut value of a subset $V' \subseteq V$ is defined to be the weight of the edge set $C = \{(u, v) | u \in V', v \in V \setminus V'\}$. In Max-Di-Cut, the objective is to find the vertex set that maximizes $\sum_{e \in C} w(e)$. Let $f : V \rightarrow \mathbb{R}^+$ be the cut value function, it can be easily verified that f is non-monotone submodular. Therefore, to prove an inapproximability result (within some algorithmic model) for the general non-monotone submodular maximization problem, it suffices to demonstrate the existence of a hard instance of Max-Di-Cut that forces a bad approximation ratio for the algorithmic model under consideration. In particular, we utilize the online model and results of Bar-Noy and Lampis³ [6].

In the online model of [6], when an input item v is revealed, the algorithm is given access to the following information:

- $w_{in}(v)$, the total weight of incoming edges to v
- $w_{out}(v)$, the total weight of outgoing edges from v
- The weights of both in and out edges connecting v to previously revealed vertices

Bar-Noy and Lampis prove the following theorem for their online model:

Theorem 3.1. [6] *No deterministic algorithm (within their data model) can achieve a competitive ratio of $\frac{2}{3\sqrt{3}} + \epsilon \approx 0.385 + \epsilon$ for any $\epsilon > 0$ for the online Max-Di-Cut problem on DAGs.*⁴

³To the best of our knowledge, the Buchbinder *et al.* [8] and Bar-Noy and Lampis [6] papers were independent of each other.

⁴When restricted to DAGs, Bar-Noy and Lampis also provide an algorithm for Max-Di-Cut with matching approximation ratio.

In order to generalize Theorem 3.1 to non-monotone submodular maximization, we need to show that the data model in [6] is powerful enough to simulate relevant value oracle queries.

Let X_{i-1} and Y_{i-1} be the set of vertices accepted and rejected, respectively, by the algorithm when v_i is revealed. Notice that from the third item in the above list, the algorithm can calculate the cut between v_i and S , for any $S \subseteq X_{i-1} \cup Y_{i-1}$.

Claim 3.2. *For Max-Di-Cut, any information obtainable through Q-Type 3 oracle queries can be inferred in the data model of [6]. The converse, however, does not hold.*

Proof. Recall that the allowable queries under Q-Type 3 are $\rho(u|S)$, and $\bar{\rho}(u|S)$ for any $S \subseteq \mathcal{N}_i$. It suffices to show that these values can be calculated using the Max-Di-Cut data model. Given that f is the directed cut function, $f(S)$ (*resp.* $\bar{f}(S)$) can be expressed as the sum of outgoing (*resp.* incoming) edge weights of vertex set S ,

$$f(S) = \sum_{s \in S} w_{out}(s) - \sum_{s, t \in S} w(s, t)$$

$$\bar{f}(S) = \sum_{s \in S} w_{in}(s) - \sum_{s, t \in S} w(s, t)$$

Then the marginal difference for a vertex $v \notin S$ with respect to S is

$$f(S \cup \{v\}) - f(S) = w_{out}(v) - c(S, v) - c(v, S) \quad (3)$$

$$\bar{f}(S \cup \{v\}) - \bar{f}(S) = w_{in}(v) - c(v, S) - c(S, v) \quad (4)$$

Since all elements of S have been revealed to the algorithm previously, the data description of v includes the complete edge weight information between v and every vertex of S . Thus the cut (in either direction) between v and S can be directly computed.

For the converse, consider the directed 3-cycle $G = (V, E)$ depicted in Figure 1 as counterexample. We show that even with complete disclosure of the cut function $f : 2^V \rightarrow \mathbb{R}$, we cannot reconstruct the edge weights $w : E \rightarrow \mathbb{R}$. Since $f(\emptyset) = f(V) = 0$ is trivial, there are $2^{|V|} - 2 = 6$ available linear equations. In comparison, there are $|E| = 6$ unknown edge weights. Thus, with equal number of variables and linear equations, a non-trivial solution exist if and only if the determinant is non-zero. We enumerate all subsets and their corresponding cut values in terms of the edge weights:

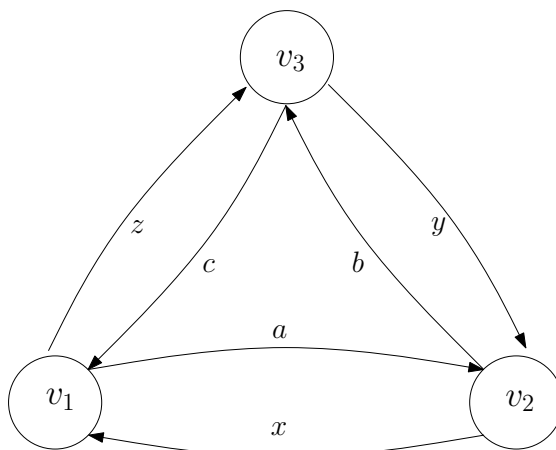


Figure 1: Directed 3 cycle with 6 edge weights.

$$\begin{aligned}
 c_1 &= f(\{v_1\}) = a + z \\
 c_2 &= f(\{v_2\}) = b + x \\
 c_3 &= f(\{v_3\}) = c + y \\
 c_4 &= f(\{v_1, v_2\}) = b + z \\
 c_5 &= f(\{v_2, v_3\}) = c + x \\
 c_6 &= f(\{v_3, v_1\}) = a + y
 \end{aligned}$$

It's easy to see that $c_1 + c_2 + c_3 = c_4 + c_5 + c_6$, implying linear dependency in the above system. Therefore, a unique solution can not be determined from the cut values alone. The argument can be extended to any input instance of size $n > 3$, by constructing a graph that contains a disjoint 3-cycle.

□

Claim 3.3. *The inapproximation of Theorem 3.1 holds even if the number of vertices is initially revealed to the algorithm.*

Proof. We prove this claim using a simple padding strategy. Define \mathcal{A} to be the adversary used in Theorem 3.1, we will construct a modified adversary \mathcal{A}' against algorithms that knows the length of the input. Let n be the size of the largest graph that \mathcal{A} might present, then \mathcal{A}' announces n as the input size to the algorithm. \mathcal{A}' will simply copy the actions of \mathcal{A} . If \mathcal{A} terminates at iteration $k \leq n$, then \mathcal{A}' sets v_{k+1}, \dots, v_n as isolated vertices. Since an isolated vertex v_{iso} has no adjacent edges, $f(S \cup \{v_{iso}\}) = f(S)$ for any set S . In the last $n - k$ iterations, the value of the algorithm's solution is therefore unchanged, regardless of its decision. On the other hand, using the same optimal solution as \mathcal{A} will result in the desired approximation ratio.

□

We now prove Theorem 1.1:

Proof of Theorem 1.1. Assume the contrary, then by Claim 3.2 and 3.3 we can simulate such an algorithm with one under the online Max-Di-Cut model when f is a cut function on a DAG. This would contradict Theorem 3.1. \square

3.1 Reinterpreting the Online Algorithm of Bar-Noy and Lampis

In this section we re-examine the doubling online algorithm in [6] for the Max-Di-Cut problem on general graphs and compare it to the deterministic version of the double-sided greedy algorithm by Buchbinder *et al.*.

Claim 3.4. *The doubling online algorithm of [6] is an instantiation of the deterministic double-sided greedy algorithm of [8].*

Proof. The doubling online algorithm for Max-Di-Cut can be described simply by the following comparison:

$$C_0(v_{i+1}) + \frac{P_0(v_{i+1})}{2} \stackrel{?}{<} C_1(v_{i+1}) + \frac{P_1(v_{i+1})}{2} \quad (5)$$

where $C_0(v)$ (*resp.* $C_1(v)$) is the certain payoff of accepting (*resp.* rejecting) v , while $P_0(v)$ (*resp.* $P_1(v)$) is the potential payoff of accepting (*resp.* rejecting) v . Then vertex v_{i+1} is accepted if Inequality 5 holds, and rejected otherwise. On the other hand, the double-sided greedy algorithm of Buchbinder *et al.* compares the incremental gain as follows

$$f(X_i \cup \{v_{i+1}\}) - f(X_i) \stackrel{?}{<} \bar{f}(Y_i \cup \{v_{i+1}\}) - \bar{f}(Y_i) \quad (6)$$

To show that the two algorithms are in fact the same, we prove that (5) and (6) are equivalent. Express the terms in (5) explicitly as follows

$$\begin{aligned} C_0(v_{i+1}) &= c(v_{i+1}, Y_i) \\ C_1(v_{i+1}) &= c(X_i, v_{i+1}) \\ P_0(v_{i+1}) &= w_{out}(v) - [c(v_{i+1}, Y_i) + c(v_{i+1}, X_i)] \\ P_1(v_{i+1}) &= w_{in}(v) - [c(Y_i, v_{i+1}) + c(X_i, v_{i+1})] \end{aligned}$$

Then (5) becomes

$$\begin{aligned} &c(v_{i+1}, Y_i) + \frac{1}{2}w_{out}(v) - \frac{1}{2}[c(v_{i+1}, Y_i) + c(v_{i+1}, X_i)] \\ &\stackrel{?}{<} c(X_i, v_{i+1}) + \frac{1}{2}w_{in}(v) - \frac{1}{2}[c(Y_i, v_{i+1}) + c(X_i, v_{i+1})] \\ \Leftrightarrow &w_{out}(v) - c(v_{i+1}, X_i) - c(X_i, v_{i+1}) \\ &\stackrel{?}{<} w_{in}(v) - c(Y_i, v_{i+1}) - c(v_{i+1}, Y_i) \end{aligned} \quad (7)$$

Replacing the two sides of the inequality by that of Equation 3 and 4 (by substituting X_i and Y_i as S , and v_{i+1} as v) yields Inequality 6. □

As a corollary, the doubling online algorithm of Bar-Noy and Lampis can be simulated by an online double-sided myopic algorithm. Furthermore, it follows from Claim 2.1 that Q-Type 1 relevant oracle access is sufficient for this simulation. In contrast, Claim 3.2 indicates that the .385 online inapproximation extends to double-sided myopic algorithms under Q-Type 3 oracle constraints.

4 0.428 and 0.450 Inapproximations for Fixed Priority Algorithms

Our lower bound argument for fixed priority algorithms is achieved by constructing a hard input instance using linear programming. The solution of the LP gives the complete mapping of the objective submodular function $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}$, where \mathcal{N} is a small finite ground set. In the construction of f , we require certain items to be indistinguishable to the algorithm, granting the adversary control over the input ordering. The drawback in this computer assisted argument is that f may not have a succinct representation like a cut function or a coverage function. Therefore, we first establish the intuition behind the adversarial strategy on a small Max-Di-Cut example, albeit with a worse lower bound.

4.1 Adversarial Strategy on Max-Di-Cut

Consider $G = (V, E)$, a directed 6-cycle with unit edge weights, and f the directed cut function on G . Since a fixed order priority algorithm must determine a total ordering before the input is revealed, the priority of an input item i can only depend on $f(i)$ and $\bar{f}(i)$ - corresponding to the total weights of out-edges and in-edges of vertex i , respectively. Clearly, $f(\emptyset) = \bar{f}(\emptyset) = 0$. Since G is regular, $\rho(i|\emptyset) = f(i) = f(j) = \rho(j|\emptyset)$, $\bar{\rho}(i|\emptyset) = \bar{f}(i) = \bar{f}(j) = \bar{\rho}(j|\emptyset)$ for all $i, j \in V$. Consequently, every input item in this instance has identical ordering priority; and due to the algorithm being deterministic, the adversary can choose any permutation as a feasible input ordering.

Denote by k the number of initial steps taken by the algorithm that the adversary will anticipate. In other words, for 2^k possible partial solutions, the adversary prepares an input ordering (consistent with the algorithm's queries in these k steps) such that any extendible solution has a bad approximation ratio.

Theorem 4.1. *For the unweighted Max-Di-Cut problem, no fixed order Q-Type 3 double-sided myopic algorithm can achieve an approximation ratio greater than $\frac{2}{3}$.*

Proof. Let v_1, \dots, v_6 be the vertices along a directed 6-cycle G with unit edge weights. Clearly, $OPT = 3$ is achieved by $\{v_1, v_3, v_5\}$ or $\{v_2, v_4, v_6\}$. The regularity of G ensures that $\pi(v_1) = \dots = \pi(v_6)$ for any π , allowing the adversary to specify any input ordering. Suppose the algorithm accepts (*resp.* rejects) v_1 in the first step, the adversary fixes

$u_{i_2} = \{v_3, v_4\}$ as the next set of possible inputs in the sequence. At this point, v_3 and v_4 are indistinguishable to the algorithm, since $\rho(v_3|S) = \rho(v_4|S) = 1$ and $\bar{\rho}(v_3|S) = \bar{\rho}(v_4|S) = 1$ for any $S \subseteq \mathcal{N}_1 = \{v_1\}$. If the algorithm accepts u_{i_2} , then the adversary sets $u_{i_2} = v_4$ (resp. $u_{i_2} = v_3$); otherwise it sets $u_{i_2} = v_3$ (resp. $u_{i_2} = v_4$). Both cases contradict the optimal solutions, and the maximum cut value is now at most 2. \square

Although this inapproximation bound does not match the currently best deterministic greedy $\frac{1}{3}$ approximation algorithm due to Buchbinder *et al.* and Bar-Noy and Lampis for Max-Di-Cut, the inapproximation shows that the SDP based 0.828 approximation cannot be realized by fixed-order deterministic double sided myopic algorithms in our model.

4.2 LP Construction for Q-Type 2

We now generalize the adversarial strategy described in the previous section to obtain a sharper inapproximability ratio for the general unconstrained submodular maximization problem. First, we consider the fixed priority setting under query type 2.

Lemma 4.2. *No fixed Q-Type 2 double-sided myopic algorithm can achieve an approximation ratio greater than $\frac{1}{c}$ if there exists a (non-negative) submodular function f with the following conditions*

Cond. 1 $f(\{u\}) = f(\{v\}), \bar{f}(\{u\}) = \bar{f}(\{v\}), \forall u, v \in \mathcal{N}$. *This imposes initial indistinguishability in the input set.*

Cond. 2 *There exist subsets $A = \{a_1, \dots, a_k\} \subseteq \mathcal{N}$, $R = \{r_1, \dots, r_k\} \subseteq \mathcal{N}$, $A \cap R = \emptyset$, such that for every $1 \leq i < j \leq k$ and every $\mathcal{C}_i \in \{a_1, r_1\} \times \{a_2, r_2\} \times \dots \times \{a_i, r_i\}$,*

$$\begin{aligned} f((\mathcal{C}_i \cap A) \cup \{a_j\}) &= f((\mathcal{C}_i \cap A) \cup \{r_j\}) \\ \bar{f}((\mathcal{C}_i \cap R) \cup \{a_j\}) &= \bar{f}((\mathcal{C}_i \cap R) \cup \{r_j\}) \end{aligned}$$

Although \mathcal{C}_i is defined as an i -vector, we abuse notation and treat \mathcal{C}_i as a set of size i . Semantically, A (resp. R) is the set of items that the algorithm is tricked into accepting (resp. rejecting). This is achievable if a_j, r_j are indistinguishable to the algorithm at round j , in the Q-Type 2 restricted oracle model.

Cond. 3 *For every $\mathcal{C}_k \in \{a_1, r_1\} \times \{a_2, r_2\} \times \dots \times \{a_k, r_k\}$, any solution $S \subseteq \mathcal{N}$ such that $\mathcal{C}_k \cap A \subseteq S$ and $S \cap \mathcal{C}_k \cap R = \emptyset$ (i.e. S is an extension of \mathcal{C}_k) must have $f(S) \leq 1$. This ensures the algorithm does not recover after making k decisions.*

Cond. 4 *There exists a set $S^* \in 2^{\mathcal{N}}$ such that $f(S^*) \geq c$.*

Proof. By **Cond. 1**, any permutation of the input items can be chosen by the adversary, since the algorithm assigns identical priority values to all items. The adversary starts by forcing the algorithm to only accept items in A , or reject items in R in the first k

rounds. That is, if the algorithm accepts (*resp.* rejects) in the j^{th} round (for $j \leq k$), then the adversary chooses an ordering of \mathcal{N} such that a_j (*resp.* r_j) is the j^{th} item and r_j (*resp.* a_j) is the l^{th} item, for some arbitrary $k < l \leq n$. To clarify, the adversary will choose either a_j or r_j as the j^{th} item and then postpone the item not chosen until after the first k items have been decided upon. By induction, assume this is achievable in the first $j < k$ steps. Then there is a set of choices $\mathcal{C}_j \in \{a_1, r_1\} \times \dots \times \{a_j, r_j\}$ such that $\mathcal{C}_j \cap A = X_j$ and $\mathcal{C}_j \cap R = Y_j$, where X_j (*resp.* Y_j) is the set of accepted (*resp.* rejected) items so far. Consider what the algorithm knows about $u_{j+1} \in \{a_{j+1}, r_{j+1}\}$ at this point. Using Q-Type 2 queries, the algorithm may learn the marginal difference in f (*resp.* \bar{f}) for u_{j+1} w.r.t. all X_i (*resp.* Y_i) for $i \leq j$. However, due to **Cond. 2**, the following must hold

$$\begin{aligned} \rho(a_{j+1}|X_0) &= \rho(r_{j+1}|X_0), \bar{\rho}(a_{j+1}|Y_0) = \bar{\rho}(r_{j+1}|Y_0) \\ &\vdots \\ \rho(a_{j+1}|X_j) &= \rho(r_{j+1}|X_j), \bar{\rho}(a_{j+1}|Y_j) = \bar{\rho}(r_{j+1}|Y_j) \end{aligned}$$

Since this is the only information available to the algorithm, the item a_{j+1} and r_{j+1} cannot be distinguished by the algorithm. Therefore if the algorithm accepts, then the adversary chooses a_{j+1} as the $j + 1^{\text{st}}$ input item, and r_{j+1} otherwise. The base case follows from the same argument, as $C_0 = X_0 = Y_0 = \emptyset$.

After k steps, the algorithm constructs a partial solution (along with partial rejection) corresponding to some $\mathcal{C}_k \in \{a_1, r_1\} \times \dots \times \{a_k, r_k\}$. By **Cond. 3**, any complete solution that can be extended now has a value bounded by 1. Thus, with the optimum solution having a value of at least c by **Cond. 4**, we have forced the algorithm to return a solution with at most $\frac{1}{c}OPT$. \square

The conditions in Lemma 4.2 can be expressed as a system of linear inequalities, which we formulate below as a linear program that maximizes the value of c . For fixed n and k , we work with a ground set of size n in the form $\mathcal{N} = \{s_1, \dots, s_{\lfloor \frac{n}{2} \rfloor}, o_1, \dots, o_{\lceil \frac{n}{2} \rceil}\}$, and designate the optimum set as $O = \{o_1, \dots, o_{\lceil \frac{n}{2} \rceil}\}$. Set $A = \{s_1, \dots, s_k\}$ and $R = \{o_1, \dots, o_k\}$ so as to deter the algorithm from the optimum set as much as possible. For every possible subset $S \subseteq \mathcal{N}$, we associate with it an LP variable x_S . As this construction entails exponentially many variables, this is indeed only feasible in practice when restricted to a small ground set. Semantically, the solution to each x_S corresponds to the value of f on S - and as such we abuse notation and refer to $f(S)$ directly as the LP variable for S . For interpretability, we may refer to the variable $\bar{f}(S_i)$ as alias for the variable $f(\mathcal{N} \setminus S_i)$ (following the definition of \bar{f}). Define the linear program as follows:

Inequality (C1) is a necessary and sufficient condition for submodularity [35], and (C5) and (C6) constrain f to be non-negative and normalized. The remaining constraints correspond to conditions 1-3 of Lemma 4.2. If the LP is feasible, then the objective value in (obj) is a lower bound for c in condition 4.

LP for Q-Type 2 Fixed Priority

Objective

$$\max f(\{o_1, \dots, o_{\lceil \frac{n}{2} \rceil}\}) \quad (\text{obj})$$

Constraints

$$f(S \cup \{v\}) + f(S \cup \{u\}) - f(S \cup \{v, u\}) \geq f(S) \quad \forall S \subseteq \mathcal{N}, (v, u) \in \mathcal{N} \setminus S \quad (\text{C1})$$

$$f(\{v\}) - f(\{u\}) = 0,$$

$$\bar{f}(\{v\}) - \bar{f}(\{u\}) = 0 \quad \forall v, u \in \mathcal{N} \quad (\text{C2})$$

$$f((\mathcal{C}_i \cap A) \cup \{a_j\}) - f((\mathcal{C}_i \cap A) \cup \{r_j\}) = 0,$$

$$\bar{f}((\mathcal{C}_i \cap R) \cup \{a_j\}) - \bar{f}((\mathcal{C}_i \cap R) \cup \{r_j\}) = 0 \quad \forall \mathcal{C}_i \in \{a_1, r_1\} \times \dots \times \{a_i, r_i\},$$

$$0 < i < j \leq k \quad (\text{C3})$$

$$f(S) \leq 1 \quad \forall S \text{ s.t. } \exists \mathcal{C}_k, \mathcal{C}_k \cap A \subseteq S$$

$$\wedge S \cap \mathcal{C}_k \cap R = \emptyset \quad (\text{C4})$$

$$f(S) \geq 0 \quad \forall S \quad (\text{C5})$$

$$f(\emptyset) = 0 \quad (\text{C6})$$

Proof of Theorem 1.2. Running the LP with $n = 8$ and $k = 4$, a feasible solution is found with objective value of 2.3333. By Lemma 4.2, this demonstrates a lower bound of $\frac{1}{2.3333} \approx 0.428$. \square

4.3 Extending to Q-Type 3

Notice that in the fixed case, the difference between oracle query types is only reflected in the decision step, since the algorithm does not gain additional power in the ordering step. In other words, we can apply the same adversarial strategy to a stronger Q-Type as long as a_i and r_i are still indistinguishable. The following lemma extends the inapproximation result to fixed priority algorithms in the Q-Type 3 model by tightening the indistinguishability constraints.

Lemma 4.3. *No Q-Type 3 fixed double-sided myopic algorithm can achieve an approximation ratio greater than $\frac{1}{c}$ if there exists a (non-negative) submodular function f that satisfies **Cond. 1-4** of Lemma 4.2, as well as the following:*

Cond. 2 \dagger *There exist subsets $A = \{a_1, \dots, a_k\} \subseteq \mathcal{N}$, $R = \{r_1, \dots, r_k\} \subseteq \mathcal{N}$, $A \cap R = \emptyset$, such that for every $0 < i < k$ and every $\mathcal{C}_i \in \{a_1, r_1\} \times \{a_2, r_2\} \times \dots \times \{a_i, r_i\}$ and every subset $S \subseteq \mathcal{C}_i$,*

$$f(S \cup \{a_{i+1}\}) = f(S \cup \{r_{i+1}\})$$

$$\bar{f}(S \cup \{a_{i+1}\}) = \bar{f}(S \cup \{r_{i+1}\})$$

To decipher this statement, recall that each \mathcal{C}_i corresponds to one of the valid input configurations in the first i rounds. Then $\mathcal{N}_i = \mathcal{C}_i$ and the equality constraints simply forces a_{i+1} and r_{i+1} to have the same marginal differences that the algorithm is allowed to query under Q-Type 3. Notice also that this condition subsumes **Cond. 2** of the previous section, when we take $S = \mathcal{C}_i \cap A$ and $S = \mathcal{C}_i \cap R$.

Proof. Regardless of the relevant query type, a fixed algorithm must decide on a priority function based only on the singleton values. Therefore, as in the previous case, **Cond. 1** allows the adversary to introduce any input ordering. To adapt the proof for Lemma 4.2 to Q-Type 3, it suffices to show that the new conditions are strong enough to push the induction step. Specifically, we would like a_{i+1} and r_{i+1} to have the same marginal descriptions at iteration $i + 1$ assuming the adversary has been successful so far. This is easy to show, since any valid input corresponds to some $\mathcal{C}_i \in \{a_1, r_1\} \times \{a_2, r_2\} \times \dots \times \{a_i, r_i\}$, and this is precisely the set of items that the algorithm has seen so far. Under Q-Type 3, the algorithm may query the marginal value of the $i + 1^{\text{st}}$ item with respect to any subset of \mathcal{C}_i — and so from **Cond. 2**†, a_{i+1} and r_{i+1} are indistinguishable. The rest of the proof then follows identically. \square

LP for Q-Type 3 Fixed Priority

Objective

$$\max f(\{o_1, \dots, o_{\lceil \frac{n}{2} \rceil}\})$$

Constraints

$$\begin{aligned} f(S \cup \{v\}) + f(S \cup \{u\}) - f(S \cup \{v, u\}) - f(S) &\geq 0 & \forall S \subseteq \mathcal{N}, (v, u) \in \mathcal{N} \setminus S \\ f(S \cup \{a_{i+1}\}) - f(S \cup \{r_{i+1}\}) &= 0, \\ \bar{f}(S \cup \{a_{i+1}\}) - \bar{f}(S \cup \{r_{i+1}\}) &= 0 & \forall S \subseteq \mathcal{C}_i, \\ & & \forall \mathcal{C}_i \in \{a_1, r_1\} \times \dots \times \{a_i, r_i\}, \\ & & 0 \leq i < k \\ f(S) &\leq 1 & \forall S \text{ s.t. } \exists \mathcal{C}_k, \mathcal{C}_k \cap A \subseteq S \\ & & \wedge S \cap \mathcal{C}_k \cap R = \emptyset \\ f(S) &\geq 0 & \forall S \\ f(\emptyset) &= 0 \end{aligned}$$

Proof of Theorem 1.3. Running the LP with $n = 8$ and $k = 4$, a feasible solution is found with objective value of 2.2222. By Lemma 4.2, this demonstrates a lower bound of $\frac{1}{2.2222} \approx 0.450$. \square

5 A 0.432 Inapproximation for Adaptive Priority Algorithms

In this section we extend the inapproximation to adaptive priority algorithms by applying a more restricted adversarial construction of that from the previous section. Surprisingly, this resulted in only a slight loss of tightness in the lower bound.

Lemma 5.1. *No Q-Type 2 adaptive double-sided myopic algorithm can achieve an approximation ratio greater than $\frac{1}{c}$ if there exists a (non-negative) submodular function f that satisfies **Cond. 1-4** of Lemma 4.2, as well as the following:*

Cond. 2* *There exist subsets $A = \{a_1, \dots, a_k\} \subseteq \mathcal{N}$, $R = \{r_1, \dots, r_k\} \subseteq \mathcal{N}$, $A \cap R = \emptyset$, such that for every $i < k$ and every $\mathcal{C}_i \in \{a_1, r_1\} \times \{a_2, r_2\} \times \dots \times \{a_i, r_i\}$ and all pairs $v, u \in \mathcal{N} \setminus \mathcal{C}_i$,*

$$\begin{aligned} f((\mathcal{C}_i \cap A) \cup \{u\}) &= f((\mathcal{C}_i \cap A) \cup \{v\}) \\ \bar{f}((\mathcal{C}_i \cap R) \cup \{u\}) &= \bar{f}((\mathcal{C}_i \cap R) \cup \{v\}) \end{aligned}$$

*This condition subsumes both **Cond. 1** and **Cond. 2** of Lemma 4.2, by requiring all unfixed items to have identical marginal descriptions - thus nullifying the benefit of adaptive ordering since the entire input will always be indistinguishable in the first k rounds.*

Proof. Here, the algorithm is permitted to reorder the input at each iteration. The adversary responds by imposing indistinguishably on *all* unfixed items in the first k rounds. Consider the ordering step at the start of iteration $i < k$. By **Cond. 2***, then $\rho(u|X_j) = \rho(v|X_j)$, $\bar{\rho}(u|Y_j) = \bar{\rho}(v|Y_j)$, $j = 0, \dots, i - 1$ for all $u, v \in \mathcal{N} \setminus (X_{i-1} \cup Y_{i-1})$. This captures the full description of all unfixed items obtainable through Q-Type 2 value oracle access. Furthermore, any other information in the internal memory is independent of the remaining items, and thus provides no additional power. Hence, all items are indistinguishable and must be assigned the same priority. The adversary can now choose to place either a_i or r_i as the i^{th} item, depending on the algorithm's decision. The rest of the proof is now identical to that of Lemma 4.2. \square

Proof of Theorem 1.4. Running the modified LP again using $n = 8$ and $k = 4$ produces a feasible solution with objective value of $c = 2.3158$, giving us an inapproximability of $\frac{1}{c} \approx 0.432$. \square

Objective

$$\max f(\{o_1, \dots, o_{\lceil \frac{n}{2} \rceil}\})$$

Constraints

$$\begin{aligned} f(S \cup \{v\}) + f(S \cup \{u\}) - f(S \cup \{v, u\}) - f(S) &\geq 0 && \forall S \subseteq \mathcal{N}, (v, u) \in \mathcal{N} \setminus S \\ f((\mathcal{C}_i \cap A) \cup \{u\}) - f((\mathcal{C}_i \cap A) \cup \{v\}) &= 0, \\ \bar{f}((\mathcal{C}_i \cap R) \cup \{u\}) - \bar{f}((\mathcal{C}_i \cap R) \cup \{v\}) &= 0 && \forall \mathcal{C}_i \in \{a_1, r_1\} \times \dots \times \{a_i, r_i\}, \\ &&& 0 \leq i < k, (u, v) \in \mathcal{N} \setminus \mathcal{C}_i \\ f(S) &\leq 1 && \forall S \text{ s.t. } \exists \mathcal{C}_k, \mathcal{C}_k \cap A \subseteq S \\ &&& \wedge S \cap \mathcal{C}_k \cap R = \emptyset \\ f(S) &\geq 0 && \forall S \\ f(\emptyset) &= 0 \end{aligned}$$

6 Discussion of the Double-Sided Myopic Model and Open Problems

Adapting the priority framework [7], we define the class of double-sided myopic algorithms and show how the double greedy algorithm of Buchbinder *et al.* can be realized as an online double-sided myopic algorithm. We show that the double-sided interpretation of the double greedy algorithm of [8] satisfies the deterministic online model of Bar-Noy and Lampis [6], for which they prove an online inapproximability of $\frac{2}{3\sqrt{3}}$ for the Max-Di-Cut problem. As in Poloczek's [37] priority inapproximation for Max-Sat, this provides evidence that the randomized $\frac{1}{2}$ -approximation double greedy for USM cannot be de-randomized.

Our inapproximation follows from an LP formulation of possible algorithmic decisions, and at present does not yield a succinctly defined problem. However, we provide a $\frac{2}{3}$ -inapproximation for Max-Di-Cut for fixed priority double-sided myopic algorithms. We wish to emphasize the generality of the myopic framework as it allows very general orderings to be defined by the algorithm, and does not impose any greedy aspect to the decisions (as to reject or accept an input) in each iteration of the algorithm. We also observe that *non-greediness* appears to be essential in both the randomized double-greedy algorithm of Buchbinder *et al.* as well as the deterministic approximation of Bar-Noy and Lampis for Max-Di-Cut on DAGs; while the deterministic double-greedy does make greedy decisions.

The gap between the $\frac{1}{3}$ deterministic double greedy and the .432 inapproximation of our myopic framework remains open as does the gap between the $\frac{1}{3}$ approximation and known myopic inapproximations for explicit functions such as Max-Di-Cut. It is

not clear how much further we can extend the LP formalization to improve our bounds, or if we can derive such bounds for succinct functions. We do not know if there is any provable difference between the online, fixed priority, and adaptive priority myopic models. In particular, can we improve upon the $\frac{1}{3}$ approximation for Max-Di-Cut by using a fixed or adaptive myopic algorithm?

In addition to the above questions for deterministic algorithms, the next obvious direction is to establish limitations for *randomized* double-sided myopic algorithms. Randomization can be utilized in the ordering step and/or the decision step of the algorithm. Of particular interest is randomization in the decision step as in the Buchbinder et al algorithm. While we know that the $\frac{1}{2}$ approximation is tight under the value oracle model and under standard complexity assumptions, it is still of interest to show such an inapproximation for myopic algorithms without any complexity constraints.

Acknowledgments

The authors would like to thank Yuval Filmus for the idea of employing linear programming, and Matthias Poloczek and Charles Rackoff for their comments and suggestions.

References

- [1] M. ALEKHNovich, A. BORODIN, J. BURESH-OPPENHEIM, R. IMPAGLIAZZO, A. MAGEN, AND T. PITASSI, *Toward a model for backtracking and dynamic programming*, in Computational Complexity, 2005. Proceedings. Twentieth Annual IEEE Conference on, IEEE, 2005, pp. 308–322.
- [2] P. ALIMONTI, *Non-oblivious local search for MAX 2-CCSP with application to max dicut*, in Graph-Theoretic Concepts in Computer Science, Springer, 1997, pp. 2–14.
- [3] N. ALON, B. BOLLOBÁS, A. GYÁRFÁS, J. LEHEL, AND A. SCOTT, *Maximum directed cuts in acyclic digraphs*, Journal of Graph Theory, 55 (2007), pp. 1–13.
- [4] S. ANGELOPOULOS, *Randomized priority algorithms*, in Approximation and Online Algorithms, Springer, 2004, pp. 27–40.
- [5] Y. AZAR, I. GAMZU, AND R. ROTH, *Submodular MAX-SAT*, in Algorithms–ESA 2011, Springer, 2011, pp. 323–334.
- [6] A. BAR-NOY AND M. LAMPIS, *Online maximum directed cut*, Journal of Combinatorial Optimization, 24 (2012), pp. 52–64.
- [7] A. BORODIN, M. N. NIELSEN, AND C. RACKOFF, *(Incremental) priority algorithms*, in Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms, SODA '02, Philadelphia, PA, USA, 2002, Society for Industrial and Applied Mathematics, pp. 752–761.
- [8] N. BUCHBINDER, M. FELDMAN, J. S. NAOR, AND R. SCHWARTZ, *A tight linear time (1/2)-approximation for unconstrained submodular maximization*, in Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science, FOCS '12, Washington, DC, USA, 2012, IEEE Computer Society, pp. 649–658.
- [9] G. CALINESCU, C. CHEKURI, M. PÁL, AND J. VONDRÁK, *Maximizing a submodular set function subject to a matroid constraint*, in Integer programming and combinatorial optimization, Springer, 2007, pp. 182–196.
- [10] J. CHEN, D. K. FRIESEN, AND H. ZHENG, *Tight bound on Johnson’s algorithm for maximum satisfiability*, J. Comput. Syst. Sci., 58 (1999), pp. 622–640.
- [11] G. CORNUEJOLS, M. FISHER, AND G. L. NEMHAUSER, *On the uncapacitated location problem*, in Studies in Integer Programming, B. K. P.L. Hammer, E.L. Johnson and G. Nemhauser, eds., vol. 1 of Annals of Discrete Mathematics, Elsevier, 1977, pp. 163 – 177.
- [12] G. CORNUEJOLS, M. L. FISHER, AND G. L. NEMHAUSER, *Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms*, Management Science, 23 (1977), pp. 789–810.

- [13] W. H. CUNNINGHAM, *On submodular function minimization*, *Combinatorica*, 5 (1985), pp. 185–192.
- [14] M. DATAR, T. FEDER, A. GIONIS, R. MOTWANI, AND R. PANIGRAHY, *A combinatorial algorithm for MAX CSP*, *Information Processing Letters*, 85 (2003), pp. 307–315.
- [15] S. DAVIS AND R. IMPAGLIAZZO, *Models of greedy algorithms for graph problems*, in *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2004, pp. 381–390.
- [16] U. FEIGE, *A threshold of $\ln n$ for approximating set cover*, *J. ACM*, 45 (1998), pp. 634–652.
- [17] U. FEIGE AND M. GOEMANS, *Approximating the value of two power proof systems, with applications to MAX 2SAT and MAX DICUT*, in *Theory of Computing and Systems*, 1995. *Proceedings.*, Third Israel Symposium on the, IEEE, 1995, pp. 182–189.
- [18] U. FEIGE AND S. JOZEPH, *Oblivious algorithms for the maximum directed cut problem*, arXiv preprint arXiv:1010.0406, (2010).
- [19] U. FEIGE, V. S. MIRROKNI, AND J. VONDRÁK, *Maximizing non-monotone submodular functions*, in *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science, FOCS '07*, Washington, DC, USA, 2007, IEEE Computer Society, pp. 461–471.
- [20] M. FELDMAN, J. S. NAOR, AND R. SCHWARTZ, *Nonmonotone submodular maximization via a structural continuous greedy algorithm*, in *Automata, Languages and Programming*, Springer, 2011, pp. 342–353.
- [21] Y. FILMUS AND J. WARD, *A tight combinatorial algorithm for submodular maximization subject to a matroid constraint*, in *Foundations of Computer Science (FOCS)*, 2012 IEEE 53rd Annual Symposium on, IEEE, 2012, pp. 659–668.
- [22] S. O. GHARAN AND J. VONDRÁK, *Submodular maximization by simulated annealing*, in *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, 2011, pp. 1098–1116.
- [23] M. X. GOEMANS AND D. P. WILLIAMSON, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, *J. ACM*, 42 (1995), pp. 1115–1145.
- [24] M. GRÖTSCHEL, L. LOVÁSZ, AND A. SCHRIJVER, *The ellipsoid method and its consequences in combinatorial optimization*, *Combinatorica*, 1 (1981), pp. 169–197.
- [25] E. HALPERIN AND U. ZWICK, *Combinatorial approximation algorithms for the maximum directed cut problem*, in *Proc. of 12th SODA*, 2001, pp. 200–1.

- [26] J. HÅSTAD, *Some optimal inapproximability results*, Journal of the ACM (JACM), 48 (2001), pp. 798–859.
- [27] D. S. HOCHBAUM, *Approximation algorithms for NP-hard problems*, PWS Publishing Co., Boston, MA, USA, 1997, ch. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems, pp. 94–143.
- [28] S. IWATA, L. FLEISCHER, AND S. FUJISHIGE, *A combinatorial strongly polynomial algorithm for minimizing submodular functions*, J. ACM, 48 (2001), pp. 761–777.
- [29] D. S. JOHNSON, *Approximation algorithms for combinatorial problems*, J. Comput. Syst. Sci., 9 (1974), pp. 256–278.
- [30] D. KEMPE, J. M. KLEINBERG, AND É. TARDOS, *Maximizing the spread of influence through a social network*, in KDD, 2003, pp. 137–146.
- [31] S. KHOT, G. KINDLER, E. MOSSEL, AND R. O’DONNELL, *Optimal inapproximability results for MAX-CUT and other 2-variable CSPs?*, SIAM Journal on Computing, 37 (2007), pp. 319–357.
- [32] J. LEE, V. S. MIRROKNI, V. NAGARAJAN, AND M. SVIRIDENKO, *Non-monotone submodular maximization under matroid and knapsack constraints*, in Proceedings of the 41st annual ACM symposium on Theory of computing, ACM, 2009, pp. 323–332.
- [33] M. LEWIN, D. LIVNAT, AND U. ZWICK, *Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems*, in Integer Programming and Combinatorial Optimization, Springer, 2006, pp. 67–82.
- [34] G. L. NEMHAUSER AND L. A. WOLSEY, *Best algorithms for approximating the maximum of a submodular set function*, Mathematics of Operations Research, 3 (1978), pp. 177–188.
- [35] G. L. NEMHAUSER, L. A. WOLSEY, AND M. L. FISHER, *An analysis of approximations for maximizing submodular set functions-I*, Mathematical Programming, 14 (1978), pp. 265–294.
- [36] A. PAUL, M. POLOCZEK, AND D. WILLAMSON, *Priority algorithms for MAX DICUT*. Cornell University, Unpublished manuscript, January 1, 2014.
- [37] M. POLOCZEK, *Bounds on greedy algorithms for MAX SAT*, in Algorithms–ESA 2011, Springer, 2011, pp. 37–48.
- [38] M. POLOCZEK AND G. SCHNITGER, *Randomized variants of Johnson’s algorithm for MAX SAT*, in Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2011, pp. 656–663.
- [39] M. POLOCZEK, D. P. WILLIAMSON, AND A. VAN ZUYLEN, *On some recent max sat approximation algorithms*, arXiv preprint arXiv:1308.3405, (2013).

- [40] O. REGEV, *Priority algorithms for makespan minimization in the subset model*, Information Processing Letters, 84 (2002), pp. 153–157.
- [41] A. SCHRIJVER, *A combinatorial algorithm minimizing submodular functions in strongly polynomial time*, J. Comb. Theory Ser. B, 80 (2000), pp. 346–355.
- [42] M. SVIRIDENKO, *A note on maximizing a submodular set function subject to a knapsack constraint*, Operations Research Letters, 32 (2004), pp. 41–43.
- [43] L. TREVISAN, *Parallel approximation algorithms by positive linear programming*, Algorithmica, 21 (1998), pp. 72–88.
- [44] A. VAN ZUYLEN, *Simpler 3/4-approximation algorithms for MAX SAT*, in Approximation and Online Algorithms, Springer, 2012, pp. 188–197.
- [45] J. VONDRÁK, *Symmetry and approximability of submodular maximization problems*, SIAM Journal on Computing, 42 (2013), pp. 265–304.
- [46] M. YANNAKAKIS, *On the approximation of maximum satisfiability*, J. Algorithms, 17 (1994), pp. 475–502.

Appendices

A LP Solution for Q-Type 3 Fixed Priority Inapproximation

We include for completeness the numerical solution of our LP formulation from Section 4 in Table 2. Define the ground set $\mathcal{N} = \{1, 2, 3, 4, 5, 6, 7, 8\}$, with the optimal solution $OPT = \{5, 6, 7, 8\}$ and $f(OPT) = 2.2222$. We set $k = 4$, with $A = \{1, 2, 3, 4\}$ and $R = \{5, 6, 7, 8\}$. Specifically, the adversary chooses $i_1 \in \{1, 5\}$, $i_2 \in \{2, 6\}$, $i_3 \in \{3, 7\}$, and $i_4 \in \{4, 8\}$ depending on the algorithm's decisions; that is, the adversary sets $i_1 = 1$ (*resp.* $i_1 = 5$) if the algorithm accepts (*resp.* rejects) the first item, and so on. To verify that our LP generated function f supports the adversarial construction described in Section 5, it suffices to verify **Cond. 1**, **2†**, **3** and **4**. Figure 2 demonstrates that **Cond. 2†** is satisfied: for each $0 \leq i < 4$ we show equality in the Q-Type 3 marginals⁵ between a_{i+1} and r_{i+1} for all allowable partial solutions. **Cond. 3** can be verified by examining Table 2 and observing that if $f(S) > 1$, then the subset S is disallowed by the adversary. Specifically, for such a subset S , there is some $i \in [1, 4]$ such that $a_i \notin S$ and $r_i \in S$; this is a contradiction since the adversary always forces the algorithm to either accept a_i or reject r_i . **Cond. 4** follows by setting $c = 2.2222$, corresponding to the optimal solution. Finally, we checked for submodularity separately by brute force, by verifying that $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$ for all $S, T \subseteq \mathcal{N}$.

Table 1: Complete description of $f : 2^{\mathcal{N}} \rightarrow \mathbb{R}^+$ used in Theorem 1.3

S	$f(S)$	S	$f(S)$	S	$f(S)$	S	$f(S)$
1	0.5555556	2,5,6	1.3333333	2,3,5,7	1.4444444	2,3,4,5,7	1.3333333
2	0.5555556	2,5,7	1.4444444	2,3,5,8	1.3888889	2,3,4,5,8	1.3333333
3	0.5555556	2,5,8	1.3333333	2,3,6,7	1.0000000	2,3,4,6,7	1.0000000
4	0.5555556	2,6,7	1.1111111	2,3,6,8	1.0000000	2,3,4,6,8	1.0000000
5	0.5555556	2,6,8	1.0000000	2,3,7,8	1.0000000	2,3,4,7,8	1.0000000
6	0.5555556	2,7,8	1.1111111	2,4,5,6	1.5555556	2,3,5,6,7	1.2777778
7	0.5555556	3,4,5	1.4444444	2,4,5,7	1.5555556	2,3,5,6,8	1.2777778
8	0.5555556	3,4,6	1.2222222	2,4,5,8	1.3333333	2,3,5,7,8	1.3333333
1,2	0.8888889	3,4,7	1.0000000	2,4,6,7	1.3333333	2,3,6,7,8	1.0000000
1,3	0.8888889	3,4,8	0.8888889	2,4,6,8	1.0000000	2,4,5,6,7	1.6666667
1,4	0.7777778	3,5,6	1.6666667	2,4,7,8	1.1111111	2,4,5,6,8	1.5555556
1,5	0.8888889	3,5,7	1.4444444	2,5,6,7	1.6666667	2,4,5,7,8	1.3333333
1,6	0.8888889	3,5,8	1.4444444	2,5,6,8	1.5555556	2,4,6,7,8	1.3333333
1,7	0.8888889	3,6,7	1.3333333	2,5,7,8	1.5555556	2,5,6,7,8	1.6666667
1,8	0.7777778	3,6,8	1.2222222	2,6,7,8	1.3333333	3,4,5,6,7	1.6666667
2,3	0.8888889	3,7,8	1.0000000	3,4,5,6	1.7777778	3,4,5,6,8	1.6666667
2,4	0.7777778	4,5,6	1.6666667	3,4,5,7	1.4444444	3,4,5,7,8	1.3333333
2,5	1.1111111	4,5,7	1.6666667	3,4,5,8	1.3888889	3,4,6,7,8	1.3333333

Continued on next page

⁵We show the value $f(\{u\} \cup S) = \rho(u|S) + f(S)$ instead of $\rho(u|S)$ as it allows for direct comparison with Table 2. Clearly, $\rho(v|S) = \rho(u|S) \Leftrightarrow f(\{u\} \cup S) = f(\{v\} \cup S)$.

Table 1 Continued from previous page

S	$f(S)$	S	$f(S)$	S	$f(S)$	S	$f(S)$
2,6	0.7777778	4,5,8	1.1111111	3,4,6,7	1.3333333	3,5,6,7,8	1.6666667
2,7	0.8888889	4,6,7	1.6666667	3,4,6,8	1.2222222	4,5,6,7,8	1.6666667
2,8	0.7777778	4,6,8	1.1111111	3,4,7,8	1.0000000	1,2,3,4,5,6	0.7222222
3,4	0.8888889	4,7,8	1.1111111	3,5,6,7	1.6666667	1,2,3,4,5,7	0.7777778
3,5	1.1111111	5,6,7	1.6666667	3,5,6,8	1.7777778	1,2,3,4,5,8	0.8888889
3,6	1.1111111	5,6,8	1.6666667	3,5,7,8	1.5555556	1,2,3,4,6,7	0.7777778
3,7	0.8888889	5,7,8	1.6666667	3,6,7,8	1.3333333	1,2,3,4,6,8	0.8888889
3,8	0.8888889	6,7,8	1.6666667	4,5,6,7	2.2222222	1,2,3,4,7,8	0.8888889
4,5	1.1111111	1,2,3,4	1.0000000	4,5,6,8	1.6666667	1,2,3,5,6,7	0.7222222
4,6	1.1111111	1,2,3,5	1.0000000	4,5,7,8	1.5555556	1,2,3,5,6,8	0.7222222
4,7	1.1111111	1,2,3,6	0.8888889	4,6,7,8	1.6666667	1,2,3,5,7,8	0.7777778
4,8	0.5555556	1,2,3,7	1.0000000	5,6,7,8	2.2222222	1,2,3,6,7,8	0.7777778
5,6	1.1111111	1,2,3,8	1.0000000	1,2,3,4,5	0.9444444	1,2,4,5,6,7	1.1111111
5,7	1.1111111	1,2,4,5	1.0000000	1,2,3,4,6	0.8888889	1,2,4,5,6,8	1.0000000
5,8	1.1111111	1,2,4,6	1.0000000	1,2,3,4,7	1.0000000	1,2,4,5,7,8	0.8888889
6,7	1.1111111	1,2,4,7	1.1111111	1,2,3,4,8	1.0000000	1,2,4,6,7,8	0.8888889
6,8	1.1111111	1,2,4,8	1.0000000	1,2,3,5,6	0.8888889	1,2,5,6,7,8	1.1111111
7,8	1.1111111	1,2,5,6	1.0000000	1,2,3,5,7	0.8888889	1,3,4,5,6,7	1.1111111
1,2,3	1.0000000	1,2,5,7	1.1111111	1,2,3,5,8	0.9444444	1,3,4,5,6,8	1.1111111
1,2,4	1.0000000	1,2,5,8	1.0555556	1,2,3,6,7	0.8333333	1,3,4,5,7,8	0.7777778
1,2,5	1.0000000	1,2,6,7	1.0555556	1,2,3,6,8	0.8888889	1,3,4,6,7,8	0.8888889
1,2,6	0.9444444	1,2,6,8	1.0000000	1,2,3,7,8	1.0000000	1,3,5,6,7,8	1.1111111
1,2,7	1.0000000	1,2,7,8	1.1111111	1,2,4,5,6	1.0000000	1,4,5,6,7,8	1.1111111
1,2,8	1.0000000	1,3,4,5	1.0000000	1,2,4,5,7	1.1111111	2,3,4,5,6,7	1.1111111
1,3,4	1.0000000	1,3,4,6	1.2222222	1,2,4,5,8	1.0000000	2,3,4,5,6,8	1.1111111
1,3,5	0.8888889	1,3,4,7	1.0000000	1,2,4,6,7	1.1111111	2,3,4,5,7,8	1.1111111
1,3,6	1.2222222	1,3,4,8	1.0000000	1,2,4,6,8	1.0000000	2,3,4,6,7,8	1.0000000
1,3,7	1.0000000	1,3,5,6	1.2222222	1,2,4,7,8	1.0000000	2,3,5,6,7,8	1.1111111
1,3,8	1.0000000	1,3,5,7	1.0000000	1,2,5,6,7	1.1111111	2,4,5,6,7,8	1.1111111
1,4,5	1.0000000	1,3,5,8	1.0000000	1,2,5,6,8	1.0555556	3,4,5,6,7,8	1.1111111
1,4,6	1.1111111	1,3,6,7	1.1666667	1,2,5,7,8	1.1111111	1,2,3,4,5,6,7	0.5555556
1,4,7	1.1111111	1,3,6,8	1.2222222	1,2,6,7,8	1.1111111	1,2,3,4,5,6,8	0.5555556
1,4,8	0.7777778	1,3,7,8	1.0000000	1,3,4,5,6	1.2222222	1,2,3,4,5,7,8	0.5555556
1,5,6	1.2222222	1,4,5,6	1.3333333	1,3,4,5,7	1.0000000	1,2,3,4,6,7,8	0.5555556
1,5,7	1.2222222	1,4,5,7	1.3333333	1,3,4,5,8	0.9444444	1,2,3,5,6,7,8	0.5555556
1,5,8	1.0555556	1,4,5,8	1.0000000	1,3,4,6,7	1.1111111	1,2,4,5,6,7,8	0.5555556
1,6,7	1.2222222	1,4,6,7	1.4444444	1,3,4,6,8	1.2222222	1,3,4,5,6,7,8	0.5555556
1,6,8	1.1111111	1,4,6,8	1.1111111	1,3,4,7,8	0.8888889	2,3,4,5,6,7,8	0.5555556
1,7,8	1.1111111	1,4,7,8	1.0000000	1,3,5,6,7	1.1666667	1,2,3,4,5,6,7,8	0.0000000
2,3,4	0.8888889	1,5,6,7	1.5555556	1,3,5,6,8	1.2222222		
2,3,5	1.4444444	1,5,6,8	1.3888889	1,3,5,7,8	1.0000000		
2,3,6	1.0000000	1,5,7,8	1.3333333	1,3,6,7,8	1.1111111		
2,3,7	1.0000000	1,6,7,8	1.4444444	1,4,5,6,7	1.6666667		
2,3,8	0.8888889	2,3,4,5	1.3888889	1,4,5,6,8	1.3333333		
2,4,5	1.3333333	2,3,4,6	1.0000000	1,4,5,7,8	1.1111111		
2,4,6	1.0000000	2,3,4,7	1.0000000	1,4,6,7,8	1.2222222		
2,4,7	1.1111111	2,3,4,8	0.8888889	1,5,6,7,8	1.6666667		
2,4,8	0.7777778	2,3,5,6	1.4444444	2,3,4,5,6	1.2777778		

Concluded

B LP Solution for Q-Type 2 Adaptive Priority Inapproximation

Similarly, we include the numerical solution of our LP formulation for the adaptive priority case. We employ identical notations as that of Appendix A. In this case, indistinguishability is required between *all* unfixed items with respect to Q-Type 2 (as in **Cond. 2***). We show this in Table 3, in which we enumerate all configurations in the first 4 steps, and for each configuration show equality in the marginal differences of all remaining items. The verification of the remaining conditions follows the description in Appendix A. In particular, observe that $f(OPT) = f(\{5, 6, 7, 8\}) = 2.3158$ gives us the claimed bound in Theorem 1.4.

Table 2: Complete description of $f : 2^N \rightarrow \mathbb{R}^+$ used in Theorem 1.4

S	$f(S)$	S	$f(S)$	S	$f(S)$	S	$f(S)$
1	0.5789474	2,5,6	1.2631579	2,3,5,7	1.3157895	2,3,4,5,7	1.3157895
2	0.5789474	2,5,7	1.4210526	2,3,5,8	1.3684211	2,3,4,5,8	1.2631579
3	0.5789474	2,5,8	1.3684211	2,3,6,7	1.0000000	2,3,4,6,7	1.0000000
4	0.5789474	2,6,7	1.2105263	2,3,6,8	1.1052632	2,3,4,6,8	1.0000000
5	0.5789474	2,6,8	1.1052632	2,3,7,8	1.1052632	2,3,4,7,8	1.0000000
6	0.5789474	2,7,8	1.2105263	2,4,5,6	1.3684211	2,3,5,6,7	1.2105263
7	0.5789474	3,4,5	1.2631579	2,4,5,7	1.4210526	2,3,5,6,8	1.3157895
8	0.5789474	3,4,6	1.1578947	2,4,5,8	1.2631579	2,3,5,7,8	1.3157895
1,2	0.8947368	3,4,7	1.0000000	2,4,6,7	1.2105263	2,3,6,7,8	1.1052632
1,3	0.8947368	3,4,8	0.9473684	2,4,6,8	1.0000000	2,4,5,6,7	1.4736842
1,4	0.8947368	3,5,6	1.4210526	2,4,7,8	1.1052632	2,4,5,6,8	1.3684211
1,5	0.8947368	3,5,7	1.2105263	2,5,6,7	1.5263158	2,4,5,7,8	1.3157895
1,6	0.8947368	3,5,8	1.2631579	2,5,6,8	1.4736842	2,4,6,7,8	1.2105263
1,7	0.8947368	3,6,7	1.2631579	2,5,7,8	1.6315789	2,5,6,7,8	1.7368421
1,8	0.8947368	3,6,8	1.2105263	2,6,7,8	1.4210526	3,4,5,6,7	1.5263158
2,3	0.8947368	3,7,8	1.1052632	3,4,5,6	1.4736842	3,4,5,6,8	1.4736842
2,4	0.8947368	4,5,6	1.5263158	3,4,5,7	1.3157895	3,4,5,7,8	1.2105263
2,5	1.1578947	4,5,7	1.3157895	3,4,5,8	1.3157895	3,4,6,7,8	1.2105263
2,6	0.8947368	4,5,8	1.2105263	3,4,6,7	1.2631579	3,5,6,7,8	1.7368421
2,7	0.8947368	4,6,7	1.4210526	3,4,6,8	1.2105263	4,5,6,7,8	1.7368421
2,8	0.8947368	4,6,8	1.2105263	3,4,7,8	1.0000000	1,2,3,4,5,6	0.8947368
3,4	0.8947368	4,7,8	1.1052632	3,5,6,7	1.5789474	1,2,3,4,5,7	0.8947368
3,5	1.0526316	5,6,7	1.7368421	3,5,6,8	1.5789474	1,2,3,4,5,8	0.8947368
3,6	1.0526316	5,6,8	1.7368421	3,5,7,8	1.4210526	1,2,3,4,6,7	0.8947368
3,7	0.8947368	5,7,8	1.7368421	3,6,7,8	1.4210526	1,2,3,4,6,8	0.8947368
3,8	0.8947368	6,7,8	1.7368421	4,5,6,7	1.6842105	1,2,3,4,7,8	0.8947368
4,5	1.0000000	1,2,3,4	1.0000000	4,5,6,8	1.6315789	1,2,3,5,6,7	0.6315789
4,6	1.1052632	1,2,3,5	1.0000000	4,5,7,8	1.5263158	1,2,3,5,6,8	0.8947368
4,7	0.8947368	1,2,3,6	1.0000000	4,6,7,8	1.5263158	1,2,3,5,7,8	0.8947368
4,8	0.7894737	1,2,3,7	1.0000000	5,6,7,8	2.3157895	1,2,3,6,7,8	0.8947368
5,6	1.1578947	1,2,3,8	1.0000000	1,2,3,4,5	1.0000000	1,2,4,5,6,7	1.0526316
5,7	1.1578947	1,2,4,5	1.0000000	1,2,3,4,6	1.0000000	1,2,4,5,6,8	0.8947368
5,8	1.1578947	1,2,4,6	1.0000000	1,2,3,4,7	1.0000000	1,2,4,5,7,8	0.8947368
6,7	1.1578947	1,2,4,7	1.1052632	1,2,3,4,8	1.0000000	1,2,4,6,7,8	0.8947368

Continued on next page

Table 2 Continued from previous page

S	$f(S)$	S	$f(S)$	S	$f(S)$	S	$f(S)$
6,8	1.1578947	1,2,4,8	1.0000000	1,2,3,5,6	0.9473684	1,2,5,6,7,8	1.1578947
7,8	1.1578947	1,2,5,6	1.0000000	1,2,3,5,7	0.8947368	1,3,4,5,6,7	1.1578947
1,2,3	1.0000000	1,2,5,7	1.1052632	1,2,3,5,8	1.0000000	1,3,4,5,6,8	1.1052632
1,2,4	1.0000000	1,2,5,8	1.1052632	1,2,3,6,7	0.8947368	1,3,4,5,7,8	0.8947368
1,2,5	1.0000000	1,2,6,7	1.1052632	1,2,3,6,8	1.0000000	1,3,4,6,7,8	0.8947368
1,2,6	1.0000000	1,2,6,8	1.1052632	1,2,3,7,8	1.0000000	1,3,5,6,7,8	1.1578947
1,2,7	1.0000000	1,2,7,8	1.1052632	1,2,4,5,6	1.0000000	1,4,5,6,7,8	1.1578947
1,2,8	1.0000000	1,3,4,5	1.0000000	1,2,4,5,7	1.1052632	2,3,4,5,6,7	1.1578947
1,3,4	1.0000000	1,3,4,6	1.2631579	1,2,4,5,8	1.0000000	2,3,4,5,6,8	1.1578947
1,3,5	1.0000000	1,3,4,7	1.0000000	1,2,4,6,7	1.1052632	2,3,4,5,7,8	1.0000000
1,3,6	1.1578947	1,3,4,8	1.0000000	1,2,4,6,8	1.0000000	2,3,4,6,7,8	0.8947368
1,3,7	1.0000000	1,3,5,6	1.2631579	1,2,4,7,8	1.0000000	2,3,5,6,7,8	1.1578947
1,3,8	1.0000000	1,3,5,7	1.0000000	1,2,5,6,7	1.1052632	2,4,5,6,7,8	1.1578947
1,4,5	1.0000000	1,3,5,8	1.1052632	1,2,5,6,8	1.0526316	3,4,5,6,7,8	1.1578947
1,4,6	1.2105263	1,3,6,7	1.2105263	1,2,5,7,8	1.2105263	1,2,3,4,5,6,7	0.5789474
1,4,7	1.1052632	1,3,6,8	1.1578947	1,2,6,7,8	1.2105263	1,2,3,4,5,6,8	0.5789474
1,4,8	0.8947368	1,3,7,8	1.1052632	1,3,4,5,6	1.2105263	1,2,3,4,5,7,8	0.5789474
1,5,6	1.2105263	1,4,5,6	1.3157895	1,3,4,5,7	1.0000000	1,2,3,4,6,7,8	0.5789474
1,5,7	1.2105263	1,4,5,7	1.2105263	1,3,4,5,8	1.0000000	1,2,3,5,6,7,8	0.5789474
1,5,8	1.1052632	1,4,5,8	1.0000000	1,3,4,6,7	1.2105263	1,2,4,5,6,7,8	0.5789474
1,6,7	1.2105263	1,4,6,7	1.4210526	1,3,4,6,8	1.1578947	1,3,4,5,6,7,8	0.5789474
1,6,8	1.2105263	1,4,6,8	1.2105263	1,3,4,7,8	1.0000000	2,3,4,5,6,7,8	0.5789474
1,7,8	1.2105263	1,4,7,8	1.1052632	1,3,5,6,7	1.2105263	1,2,3,4,5,6,7,8	0.0000000
2,3,4	1.0000000	1,5,6,7	1.5263158	1,3,5,6,8	1.2631579		
2,3,5	1.3684211	1,5,6,8	1.4210526	1,3,5,7,8	1.1052632		
2,3,6	1.0000000	1,5,7,8	1.4210526	1,3,6,7,8	1.2105263		
2,3,7	1.0000000	1,6,7,8	1.5263158	1,4,5,6,7	1.4736842		
2,3,8	1.0000000	2,3,4,5	1.3684211	1,4,5,6,8	1.3157895		
2,4,5	1.2631579	2,3,4,6	1.0000000	1,4,5,7,8	1.2105263		
2,4,6	1.0000000	2,3,4,7	1.0000000	1,4,6,7,8	1.2105263		
2,4,7	1.1052632	2,3,4,8	1.0000000	1,5,6,7,8	1.7368421		
2,4,8	0.8947368	2,3,5,6	1.3157895	2,3,4,5,6	1.3157895		

Concluded

Table 3: Indistinguishability between all unfixed items in the first 3 iterations with respect to Q-Type 2.

$X_0 = \{\}, Y_0 = \{\}$ $f(1) = 0.5789474$ $\bar{f}(1) = 0.5789474$ $f(2) = 0.5789474$ $\bar{f}(2) = 0.5789474$ $f(3) = 0.5789474$ $\bar{f}(3) = 0.5789474$ $f(4) = 0.5789474$ $\bar{f}(4) = 0.5789474$ $f(5) = 0.5789474$ $\bar{f}(5) = 0.5789474$ $f(6) = 0.5789474$ $\bar{f}(6) = 0.5789474$ $f(7) = 0.5789474$ $\bar{f}(7) = 0.5789474$ $f(8) = 0.5789474$ $\bar{f}(8) = 0.5789474$	$X_1 = \{1\}, Y_1 = \{\}$ $f(12) = 0.8947368$ $\bar{f}(2) = 0.5789474$ $f(13) = 0.8947368$ $\bar{f}(3) = 0.5789474$ $f(14) = 0.8947368$ $\bar{f}(4) = 0.5789474$ $f(15) = 0.8947368$ $\bar{f}(5) = 0.5789474$ $f(16) = 0.8947368$ $\bar{f}(6) = 0.5789474$ $f(17) = 0.8947368$ $\bar{f}(7) = 0.5789474$ $f(18) = 0.8947368$ $\bar{f}(8) = 0.5789474$	$X_1 = \{1, 2\}, Y_1 = \{\}$ $\bar{f}(123) = 1$ $\bar{f}(3) = 0.5789474$ $f(124) = 1$ $\bar{f}(4) = 0.5789474$ $f(125) = 1$ $\bar{f}(5) = 0.5789474$ $f(126) = 1$ $\bar{f}(6) = 0.5789474$ $f(127) = 1$ $\bar{f}(7) = 0.5789474$ $f(128) = 1$ $\bar{f}(8) = 0.5789474$	$X_2 = \{1, 2, 3\}, Y_2 = \{\}$ $\bar{f}(1234) = 1$ $\bar{f}(4) = 0.5789474$ $f(1235) = 1$ $\bar{f}(5) = 0.5789474$ $f(1236) = 1$ $\bar{f}(6) = 0.5789474$ $f(1237) = 1$ $\bar{f}(7) = 0.5789474$ $f(1238) = 1$ $\bar{f}(8) = 0.5789474$
$X_2 = \{1, 2\}, Y_2 = \{7\}$ $\bar{f}(123) = 1$ $\bar{f}(73) = 0.8947368$ $f(124) = 1$ $\bar{f}(74) = 0.8947368$ $f(125) = 1$ $\bar{f}(75) = 0.8947368$ $f(126) = 1$ $\bar{f}(76) = 0.8947368$ $f(128) = 1$ $\bar{f}(78) = 0.8947368$	$X_2 = \{1\}, Y_2 = \{6\}$ $f(12) = 0.8947368$ $\bar{f}(62) = 0.8947368$ $f(13) = 0.8947368$ $\bar{f}(63) = 0.8947368$ $f(14) = 0.8947368$ $\bar{f}(64) = 0.8947368$ $f(15) = 0.8947368$ $\bar{f}(65) = 0.8947368$ $f(17) = 0.8947368$ $\bar{f}(67) = 0.8947368$ $f(18) = 0.8947368$ $\bar{f}(68) = 0.8947368$	$X_3 = \{1, 3\}, Y_3 = \{6\}$ $\bar{f}(132) = 1$ $\bar{f}(62) = 0.8947368$ $f(134) = 1$ $\bar{f}(64) = 0.8947368$ $f(135) = 1$ $\bar{f}(65) = 0.8947368$ $f(137) = 1$ $\bar{f}(67) = 0.8947368$ $f(138) = 1$ $\bar{f}(68) = 0.8947368$	$X_3 = \{1\}, Y_3 = \{6, 7\}$ $\bar{f}(12) = 0.8947368$ $\bar{f}(672) = 1$ $\bar{f}(13) = 0.8947368$ $\bar{f}(673) = 1$ $\bar{f}(14) = 0.8947368$ $\bar{f}(674) = 1$ $\bar{f}(15) = 0.8947368$ $\bar{f}(675) = 1$ $\bar{f}(18) = 0.8947368$ $\bar{f}(678) = 1$
$X_3 = \{\}, Y_3 = \{5\}$ $\bar{f}(1) = 0.5789474$ $\bar{f}(51) = 0.8947368$ $f(2) = 0.5789474$ $\bar{f}(52) = 0.8947368$ $f(3) = 0.5789474$ $\bar{f}(53) = 0.8947368$ $f(4) = 0.5789474$ $\bar{f}(54) = 0.8947368$ $f(6) = 0.5789474$ $\bar{f}(56) = 0.8947368$ $f(7) = 0.5789474$ $\bar{f}(57) = 0.8947368$ $f(8) = 0.5789474$ $\bar{f}(58) = 0.8947368$	$X_3 = \{2\}, Y_3 = \{5\}$ $\bar{f}(21) = 0.8947368$ $\bar{f}(51) = 0.8947368$ $\bar{f}(23) = 0.8947368$ $\bar{f}(53) = 0.8947368$ $\bar{f}(24) = 0.8947368$ $\bar{f}(54) = 0.8947368$ $\bar{f}(26) = 0.8947368$ $\bar{f}(56) = 0.8947368$ $\bar{f}(27) = 0.8947368$ $\bar{f}(57) = 0.8947368$ $\bar{f}(28) = 0.8947368$ $\bar{f}(58) = 0.8947368$	$X_3 = \{2, 3\}, Y_3 = \{5\}$ $\bar{f}(231) = 1$ $\bar{f}(51) = 0.8947368$ $\bar{f}(234) = 1$ $\bar{f}(54) = 0.8947368$ $\bar{f}(236) = 1$ $\bar{f}(56) = 0.8947368$ $\bar{f}(237) = 1$ $\bar{f}(57) = 0.8947368$ $\bar{f}(238) = 1$ $\bar{f}(58) = 0.8947368$	$X_3 = \{2\}, Y_3 = \{5, 7\}$ $\bar{f}(21) = 0.8947368$ $\bar{f}(571) = 1$ $\bar{f}(23) = 0.8947368$ $\bar{f}(573) = 1$ $\bar{f}(24) = 0.8947368$ $\bar{f}(574) = 1$ $\bar{f}(26) = 0.8947368$ $\bar{f}(576) = 1$ $\bar{f}(28) = 0.8947368$ $\bar{f}(578) = 1$
$X_3 = \{\}, Y_3 = \{5, 6\}$ $\bar{f}(1) = 0.5789474$ $\bar{f}(561) = 1$ $\bar{f}(2) = 0.5789474$ $\bar{f}(562) = 1$ $\bar{f}(3) = 0.5789474$ $\bar{f}(563) = 1$ $\bar{f}(4) = 0.5789474$ $\bar{f}(564) = 1$ $\bar{f}(7) = 0.5789474$ $\bar{f}(567) = 1$ $\bar{f}(8) = 0.5789474$ $\bar{f}(568) = 1$	$X_3 = \{3\}, Y_3 = \{6, 7\}$ $\bar{f}(31) = 0.8947368$ $\bar{f}(561) = 1$ $\bar{f}(32) = 0.8947368$ $\bar{f}(562) = 1$ $\bar{f}(34) = 0.8947368$ $\bar{f}(564) = 1$ $\bar{f}(37) = 0.8947368$ $\bar{f}(567) = 1$ $\bar{f}(38) = 0.8947368$ $\bar{f}(568) = 1$	$X_3 = \{\}, Y_3 = \{5, 6, 7\}$ $\bar{f}(1) = 0.5789474$ $\bar{f}(5671) = 1$ $\bar{f}(2) = 0.5789474$ $\bar{f}(5672) = 1$ $\bar{f}(3) = 0.5789474$ $\bar{f}(5673) = 1$ $\bar{f}(4) = 0.5789474$ $\bar{f}(5674) = 1$ $\bar{f}(8) = 0.5789474$ $\bar{f}(5678) = 1$	