Video Distribution Under Multiple Constraints*

Boaz Patt-Shamir Dror Rawitz
boaz@eng.tau.ac.il rawitz@eng.tau.ac.il
School of Electrical Engineering
Tel Aviv University
Tel Aviv, 69978
Israel

Abstract

We consider the optimization problem of providing a set of video streams to a set of clients, where each stream has costs in m possible measures (such as communication bandwidth, processing bandwidth etc.), and each client has its own utility function for each stream. We assume that the server has a budget cap on each of the m cost measures; each client has an upper bound on the utility that can be derived from it, and potentially also upper bounds in each of the m cost measures. The task is to choose which streams the server will provide, and out of this set, which streams each client will receive. The goal is to maximize the overall utility subject to the budget constraints. We give an efficient approximation algorithm with approximation factor of $O(m)$ with respect to the optimal possible utility for any input, assuming that clients have only a bound on their maximal utility. If, in addition, each client has at most m_c capacity constraints, then the approximation factor increases by another factor of $O(m_c \log n)$, where n is the input length. We also consider the special case of "small" streams, namely where each stream has cost of at most $O(1/\log n)$ fraction of the budget cap, in each measure. For this case we present an algorithm whose approximation ratio is $O(\log n)$.

Keywords: Approximation Algorithms, Budgeted Set Cover, Resource Allocation, Scheduling, Video Distribution.

*An extended abstract of this paper was presented at the 28th International Conference on Distributed Computing Systems (ICDCS), 2008. Research supported in part by the Next Genration Video (NeGeV) consortium, Israel, and by the Israel Science Foundation (grant 664/05).

1 Introduction

The following model is an abstraction of the way cable TV is distributed in many cases (see Figure 1). There are many available *streams* to multicast, and there are *clients* (or *users*), each with his own *utility* for each stream. A client may be an individual household, or a neighborhood video gateway, and the utility may represent the revenue generated by the client, or a measure of user satisfaction. The server (which can be a cable head-end serving video gateways, or a video gateway serving households) transmits a subset of the available streams over a multicast-capable network (typically Ethernet or DOCSIS): A transmitted stream can be received by all clients. The objective of the system is to maximize overall utility, but there are several constraints which any solution must respect. At the server, these constraints typically include limited outgoing communication bandwidth, and may also include limited processing bandwidth, limited number of input ports, etc. In general, transmitting a stream incurs a cost at the server in each of m possible measures. In our scenario, each of these m cost measures has a given *budget* cap that may not be exceeded. At the client side, the system main constraint is that only a bounded amount of utility can be derived from each client. Clients may have other limited resources: for example, a client typically has a maximal incoming bandwidth limit. In general, we assume that each client has up to m_c budgets, and each stream has a cost in each of the clients' budgets. The task is, subject to the given constraints, to select streams to broadcast by the server, and to select streams to deliver to each user, so as to maximize the overall utility of the system.

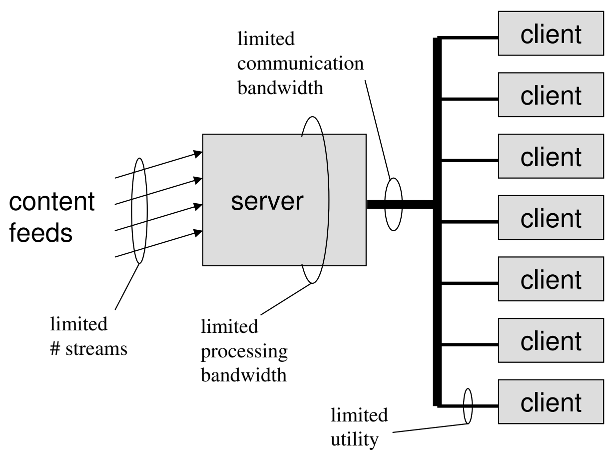

Figure 1: *Schematic representation of a typical system. The server serves contents to clients, using a bounded number of input streams, bounded computational bandwidth, and bounded outgoing communication bandwidth. Each client can generate bounded utility.*

It is easy to see that finding the optimal solution to this very practical problem is computationally hard: even if there were a single user, the problem is a strict generalization of the Knapsack Problem; from another perspective, even if there were a single cost measure, and each stream had either unit cost and unit utility or zero cost and zero utility for each user, then the problem is a generalization of the Maximum Coverage Problem [12]. We therefore resort to near-optimal solutions, which guarantee worst-case approximation ratio with respect to the optimal solution.

It may be worthwhile to note that most solutions in use today employ a simple threshold-based admission control policy, where requests are admitted so long as they do not go over certain "safety margins" for the resources in question. While the choice of the threshold can be quite sophisticated (see, e.g., [5, 4]), it appears that this approach is somewhat naïve, in that it ignores the possibly very different utilities of different stream, which is the main difficulty we tackle in this paper. We also note that there is infra-structure to support more sophisticated policies (such as the one we propose); one such example is Cisco's architecture, that allows for a server called Broadband Policy Manager (BPM) [7, 8].

1.1 Our Results

We present several approximation algorithms for the problem. First, we provide an algorithm for the general case. Our general algorithm uses as a building block an algorithm (with smaller approximation ratio) for the special case where the only constraint at the client side is caps on the client utilities. Our second main algorithm is for the special case where all streams have small costs with respect to the budget caps: for that case, we can guarantee a much better approximation ratio.

To state the results precisely, we need to define the problem formally. (A glossary summarizing the notation we use appears in Figure 2.)

Multi-Budget Multi-Client Distribution (MMD)

Input:

- A collection $\mathcal{S}$ of *streams*, a set U of *users*, and two integers $m, m_c > 0$.
- A *server cost* $c_i(S) \geq 0$, for each $S \in \mathcal{S}$ and $1 \leq i \leq m$, and a *server budget* $B_i \in \mathbb{R}^+ \cup \{\infty\}$, for each $1 \leq i \leq m$.
  We assume that $c_i(S) \leq B_i$ for every i and S.
- A *user load* $k_j^u(S) \geq 0$, for each $1 \leq j \leq m_c$, stream $S \in \mathcal{S}$, and user $u \in U$, and a *user capacity* $K_j^u \in \mathbb{R}^+ \cup \{\infty\}$, for each $1 \leq j \leq m_c$ and user $u \in U$.
- A *user utility* $w_u(S)$ for each user u and stream S.
  We note that $w_u(S) = 0$ means that user u does not want or cannot receive stream S. We assume that $w_u(S) = 0$ if $k_j^u(S) > K_j^u$ for some j.

Output: an assignment of a set of streams $A(u)$ to each user u maximizing

$$w(A) \stackrel{\text{def}}{=} \sum_{u \in U} \sum_{S \in A(u)} w_u(S) \ ,$$

such that

- *Server budget constraints:* For each $1 \leq i \leq m$,

$$\sum_{S \in \cup_{u \in U} A(u)} c_i(S) \leq B_i \ .$$

- *User capacity constraints:* For each $1 \leq j \leq m_c$ and user u,

$$\sum_{S \in A(u)} k_j^u(S) \leq K_j^u \ .$$

Quantities related to an MMD instance:

- $\mathcal{S}$: streams set
- U: users set
- c_i: ith cost function
- B_i: ith budget
- m: number of server budgets
- k_j^u: jth load function of users u
- K_j^u: jth capacity of u
- m_c: number of user budgets
- $w(S) \stackrel{\text{def}}{=} \sum_{u \in S} w_u(S)$: total utility of stream S where $f(\mathcal{C}) \stackrel{\text{def}}{=} \sum_{S \in \mathcal{C}} f(S)$ for any subset $\mathcal{C} \subseteq \mathcal{S}$ and function $f : \mathcal{S} \to \mathbb{R}^+$.

Quantities related to an assignment A:

- $\mathcal{S}(A) \stackrel{\text{def}}{=} \bigcup_{u \in U} A(u)$, also called the *range* of A: the set of streams that are assigned to users by A.
- $c_i(A) \stackrel{\text{def}}{=} c_i(\mathcal{S}(A))$: ith cost of A
- $k_j^u(A) \stackrel{\text{def}}{=} k_j^u(A(u))$: jth load of A on u
- $w_u(A) \stackrel{\text{def}}{=} w_u(A(u))$: utility of A with respect to u

Figure 2: Glossary of Notation

We also consider the special case of MMD where there is only one server budget constraint, and also there is only one capacity constraint per user (i.e., $m = m_c = 1$). We refer to this special case as the **Single-Budget Multi-Client Distribution** problem, abbreviated henceforth SMD.

Before we state our results, we need to define yet another concept. Given a capacity measure i and a user u, one can compare all streams in terms of their cost-benefit ratio: how much utility is generated by a stream for unit load. We define the *local skew* of user u at capacity measure i to be the ratio between the largest and smallest cost-benefit ratios (not including zero utility streams). The *local skew of an instance*, denoted α henceforth, is the maximum, over all users u and all load measures i, of the local skew of u at i. (A formal definition is given in Section 3.) Note that $\alpha \geq 1$ always, and equality holds if and only if all load functions of each user u are proportional to his utility w_u. Also, note that $\log \alpha = O(\log n)$ when all numbers in the input are polynomial in n (in this paper all logarithms are to base 2 unless otherwise stated).

Using the notion of local skew, we state our main result. For simplicity, we consider the case where all costs and utilities are polynomial in the input length n.

Theorem 1.1. *There exists an $O(n^2)$ time $O(mm_c \log(2\alpha m_c))$-approximation algorithm for* MMD*, where α is the local skew of the instance, m is the number of cost measures, m_c is the maximal number of capacity constrains at a user, and n is the input length.*

Note that if each user has only a single capacity constraint, and if the local skew is 1 (which means that the user is only limited by the maximal utility it can generate), then our algorithm guarantees an $O(m)$ approximation. If all costs and utilities are polynomial in the input length n, then the approximation ratio is $O(mm_c \log n)$.

Our second result deals with streams with small costs and loads. We first generalize the local skew α as follows. Given an MMD instance, we define the *global skew* of the instance denoted by γ. Intuitively, the global skew γ bounds the ratio between the best and the worst streams, in terms of utility for each unit cost. (The exact definition is given in Section 5.) We note that $\gamma \geq \alpha$ for all instances of MMD, and that γ is polynomial in n if all numbers in the input are polynomial in n.

Theorem 1.2. *Given an* MMD *instance, let* $\mu \stackrel{\text{def}}{=} 2\gamma(m + |U|m_c) + 1$*, where* γ *is the global skew of the instance,* m *is the number of cost measures, and* m_c *is the maximal number of capacity constrains at a user. Suppose that* $c_i(S) \leq \frac{B_i}{\log \mu}$ *for every* i*, and* $k_i^u(S) \leq \frac{K^u}{\log \mu}$ *for every* i *and* u*. Then an* $O(\log \mu)$*-approximation can be found in polynomial time.*

The ratio is $O(\log n)$ if all numbers in the input are polynomial in n.

It is important to note that the algorithm we present to prove Theorem 1.2 is actually an *online* algorithm with competitive ratio $O(\log \mu)$. By "online" we mean that the algorithm considers streams one by one as they arrive, and decides whether to supply the stream and to which users, without knowledge of future arrivals.

1.2 Previous work

Our model can be viewed as a generalization of the Budgeted Set Cover problem [13], which is a variant of the Set Cover problem [11]. In the set cover problem, the input consists of a collection of sets with cost for each set; the goal is to find a subcollection of sets of minimal cost, whose union is the same as the union of the complete collection. Set cover admits $O(\log n)$ approximation [17] and not better, unless P = NP [9, 2]. In the budgeted set cover problem, the input consists of a "budget" B and a collection of sets of weighted elements, where each set has a cost. The goal is to find a subcollection of the sets whose cost is at most B, maximizing the total weight of the union. In the (unweighted) Maximum Coverage problem, the goal is to cover as many elements as possible, using at most B sets. In this case the natural greedy algorithm computes solutions whose weight is within a factor of $1 - (1 - \frac{1}{B})^B > 1 - \frac{1}{e} \approx 0.63$ from the optimum (see [14, 12]). This ratio holds even in the more general case of nonnegative, nondecreasing, submodular set function maximization [15, 10]. (A function f is called submodular if $f(T) + f(T') \geq f(T \cup T') + f(T \cap T')$ for every two sets T, T' in the domain of f.)

Khuller, Moss and Naor [13] show that budgeted set cover can be approximated to within $\frac{e}{e-1}$, and cannot be approximated to within any smaller factor unless NP $\subseteq$ DTIME$(n^{O(\log \log n)})$. This hardness result holds for Maximum Coverage as well. Sviridenko [16] extended the result from [13] to maximization of a nondecreasing submodular set function subject to a budget constraint. Ageev and Sviridenko [1] presented an approximation algorithm for budgeted set cover with unit costs whose approximation ratio is $1 - (1 - \frac{1}{d})^d$, where d is the maximum size of a set.

Another generalization of the budgeted set cover problem is the "group budget constraint" [6], where the sets are assumed to be partitioned into disjoint "groups" and at most one set from each group may be selected to the output. The task is to maximize the size of the union of the output sets, subject to a budget constraint. In [6] it is shown that if all sets have unit cost then approximation to within 2 is possible; if sets have different costs, then the approximation factor jumps to 12. We note that the problem we consider is a strict generalization of both variants of the budgeted set cover problem mentioned above.

The work by Awerbuch, Azar, and Plotkin [3] is also closely related to this paper. In [3] the question is whether to admit calls into a network (and how to route them), so as to maximize overall throughput subject to link capacity constraints. One important difference between the models is that in our case, the utility of a stream depends on the algorithm (which users receive the stream), whereas the "profit" of a call in [3] is part of the input.

1.3 Solution overview and paper organization

The algorithm which proves Theorem 1.1 applies a series of transformations as follows.

1. First, the multi-budget (MMD) instance is transformed into a single-budget (SMD) instance. This transformation may increase the local skew, but only by a multiplicative factor of m_c.
2. Second, we show how to transform a general SMD instance with skew $\alpha > 1$ into multiple SMD instances with unit skew each, and produce a result whose approximation ratio is blown up (with respect to the unit-skew solution) by a factor of $O(\log \alpha)$.
3. Finally, we solve the SMD problem for unit skew by a constant-factor approximation algorithm.

We describe the algorithm in a bottom-up fashion: In Section 2 we describe an $O(1)$-approximation algorithm for SMD with unit skew, the reduction from arbitrary to unit skew is described in Section 3, and in Section 4, we describe the transformation of MMD to SMD.

The algorithm for Theorem 1.2 is based on ideas from [3]. It is described and analyzed in Section 5.

2 The SMD Problem: Single Budget Constraint

In this section we consider the case of a single budget constraint and a single capacity constraint per user with unit skew $\alpha = 1$. Equivalently, each stream has a (single) cost at the server, and each user can generate bounded utility. We give constant factor approximation algorithms for this case. Our general approach, following the work of Khuller et al. [13], is to use a greedy algorithm for this case, namely to iteratively allocate the most *cost-effective* stream to all possible users. This part is described in Section 2.1. However, as in [13] the greedy algorithm is not good enough: In Section 2.2, we explain the problem and show how to fix it so as to yield a constant approximation factor.

We present an $O(n^2)$-time algorithm which produces utility at least $(e-1)/2e$ times the optimal utility, if we increase the capacity of every user u by $K^u + \bar{k}^u$, where $\bar{k}^u = \max_S k^u(S)$. This is the *resource augmentation* model. Without resource augmentation, the algorithm guarantees approximation factor of $\frac{3e}{e-1}$. For completeness, we present in Section 2.3 another algorithm whose approximation factors are better: $\frac{e}{e-1}$ with resource augmentation, and $\frac{2e}{e-1}$ without resource augmentation. However, the latter algorithm requires more running time (albeit polynomial).

Preliminaries. When the local skew is 1, either $w_u(S) = k_u(S)$ or $w_u(S) = 0$, for every u and S. Hence, in the remainder of this section, for each user u, we only consider his utility function w_u and his utility bound W_u.

In our algorithm, we may allocate a stream S to a user u even if the residual utility of the user is less than $w_u(S)$ so as to saturate the user (this happens at most once for each user). Such assignments, that satisfy the server constraints, but may violate the users' constraints are called *semi-feasible*. We extend the definition of $w(A)$ to semi-feasible assignments as follows:

$$w(A) \stackrel{\text{def}}{=} \sum_u \min \{W_u, w_u(A)\} \ .$$

This means that the utility that a user u contributes is never more W_u. In a similar way we define the *fractional residual utility* of a user u for a stream S with respect to an assignment A to be the utility that S adds to u if it is added to A. Formally, $\bar{w}_u^A(S) = 0$ for $S \in \mathcal{S}(A)$; if $S \notin \mathcal{S}(A)$, then

$$\bar{w}_u^A(S) \ = \ \min \{w_u(S), \max \{W_u - w_u(A), 0\}\} \ = \ \begin{cases} w_u(S) & W_u - w_u(A) \geq w_u(S) \ , \\ W_u - w_u(A) & 0 \leq W_u - w_u(A) < w_u(S) \ , \\ 0 & W_u - w_u(A) < 0 \ . \end{cases}$$

The *fractional residual utility* of S is $\bar{w}^A(S) = \sum_u \bar{w}_u^A(S)$.

Finally, we define the *cost effectiveness* of a stream S. Given a cost function c, the cost effectiveness of S with respect to a given assignment A is defined as $\bar{w}^A(S)/c(S)$.

2.1 Basic Algorithm: Greedy

Algorithm **Greedy**, specified formally below, starts with the empty assignment, and iteratively adds to the solution a stream with maximum cost effectiveness with respect to the current assignment. The algorithm uses fractional residual utilities. This allows the algorithm to assign a stream S to a user u even if $\sum_{S' \in A(u)} w_u(S') > W_u - w_u(S)$. (The semi-feasible assignment is a useful intermediate step in the analysis, but in the final solution, the assignment is feasible.)

Algorithm 1 - Greedy$(U, \mathcal{S}, c, w, W, B)$

$A(u) \leftarrow \emptyset$, for every u
$\mathcal{C} \leftarrow \mathcal{S}$
while $\mathcal{C} \neq \emptyset$ **do**
  Let S be a stream that maximizes $\bar{w}^A(S)/c(S)$
  if $c(A) + c(S) \leq B$ **then**
    $A(u) \leftarrow A(u) \cup \{S\}$ for every u such that $\bar{w}_u^A(S) > 0$.
  end if
  $\mathcal{C} \leftarrow \mathcal{C} \setminus \{S\}$
end while
return A

Complexity Analysis. We first consider the implementation of Algorithm **Greedy**, and explain how to get time complexity of $O(|S|n) = O(n^2)$. In each iteration, we find the stream S of maximum cost effectiveness. Given the stream residual utilities this can be done in $O(|\mathcal{S}|)$. If S is too expensive it is dropped. Otherwise, we assign S to the users that are not yet saturated. We then remove S and all users whose residual utility became 0. We also need to update the residual utility of the remaining streams. In a straightforward implementation all the above updates are done in $O(n)$ time. We update the residual utility of $O(|U|)$ users due to the assignment of S, and then the residual utility of each remaining stream S' is updated according to the residual utility of all users u for which $w_u(S') > 0$. Since the total number of iterations is $O(|\mathcal{S}|)$, the total running time is $O(|\mathcal{S}|n)$.

Performance Analysis. We analyze the utility of the solution computed by Algorithm **Greedy** by comparing it to the utility of any semi-feasible assignment SF (including the best such assignment).

The performance guarantee of Algorithm **Greedy** follows from the observation that the utility of semi-feasible assignments is a submodular function. More precisely, let us consider an assignment just by the set of streams provided by the server. The utility of a set of streams $\mathcal{T} \subseteq \mathcal{S}$ provided by the server for a given user u is defined by $w_u(\mathcal{T}) \stackrel{\text{def}}{=} \min\left\{W_u, \sum_{S \in \mathcal{T}} w_u(S)\right\}$. We also define $w(\mathcal{T}) \stackrel{\text{def}}{=} \sum_u w_u(\mathcal{T})$. Note that this definition ignores the actual assignment of streams to users, but it coincides with the utility achieved by semi-feasible assignments. Thus defined, it is not hard to see that the utility of a semi-feasible assignment is submodular.

Lemma 2.1. *The utility function $w : 2^{\mathcal{S}} \to \mathbb{R}$ is nonnegative, nondecreasing, submodular, and polynomially computable.*

Proof. It is not hard to verify that w is nonnegative, nondecreasing, and polynomially computable. It remains to prove that w is submodular.

We show that for any user u, and for any two stream sets $\mathcal{T}, \mathcal{T}'$,

$$w_u(\mathcal{T}) + w_u(\mathcal{T}') \geq w_u(\mathcal{T} \cup \mathcal{T}') + w_u(\mathcal{T} \cap \mathcal{T}') \ .$$

Without loss of generality, assume that $w_u(\mathcal{T}) \geq w_u(\mathcal{T}')$. Now, if $w_u(\mathcal{T}) < W_u$, then

$$\begin{aligned} w_u(\mathcal{T}) + w_u(\mathcal{T}') &= \sum_{S \in \mathcal{T}} w_u(S) + \sum_{S \in \mathcal{T}'} w_u(S) \\ &= \sum_{S \in \mathcal{T} \cup \mathcal{T}'} w_u(S) + \sum_{S \in \mathcal{T} \cap \mathcal{T}'} w_u(S) \\ &\geq w_u(\mathcal{T} \cup \mathcal{T}') + w_u(\mathcal{T} \cap \mathcal{T}') \ . \end{aligned}$$

Otherwise, if $w_u(\mathcal{T}) = W_u$, then

$$w_u(\mathcal{T}) + w_u(\mathcal{T}') \ = \ W_u + w_u(\mathcal{T}') \ \geq \ w_u(\mathcal{T} \cup \mathcal{T}') + w_u(\mathcal{T} \cap \mathcal{T}') \ .$$

It follows that $w(\mathcal{T}) + w(\mathcal{T}') \geq w(\mathcal{T} \cup \mathcal{T}') + w(\mathcal{T} \cap \mathcal{T}')$, for any two stream sets $\mathcal{T}, \mathcal{T}'$. ☐

We can therefore apply the result of Sviridenko [16] to obtain a performance guarantee.

First we need to define some notation. Let S_i denote the ith stream considered by the algorithm, i.e., S_i is considered in the ith iteration. Let k be the number of iterations that were executed by Algorithm **Greedy** until the first stream S_{k+1} from $\mathcal{S}(\text{SF}) \setminus \mathcal{S}(A)$ is considered, but not used by A (because its addition violates the budget constraint). For $i \leq k$, let A_i denote the assignment A after the ith iteration, i.e., after considering S_i (A_0 is the empty assignment). Also, denote by A_{k+1} the (infeasible) assignment that is obtained by adding S_{k+1} to A_k. With this notation, and the observation that the utility function of semi-feasible assignments is submodular, we obtain the following result.

Lemma 2.2. $w(A_{k+1}) = w(A_k) + \bar{w}^{A_k}(S_{k+1}) \geq (1 - \frac{1}{e}) \cdot w(\mathrm{SF})$.

Next, we provide a complete proof in our terminology which does not rely on [16].

Lemma 2.3. *For every $i \leq k+1$, either $A_i = A_{i-1}$ or $w(A_i) - w(A_{i-1}) \geq \frac{c(S_i)}{B}(w(\mathrm{SF}) - w(A_{i-1}))$.*

Proof. We assume that $A_i \neq A_{i-1}$ and prove that $w(A_i) - w(A_{i-1}) \geq \frac{c(S_i)}{B}(w(\mathrm{SF}) - w(A_{i-1}))$. Let $U_{i-1} = \{u : w_u(A_{i-1}) < W_u\}$, namely U_{i-1} is the set of users that are not saturated by A_{i-1}. Observe that directly from definitions, we have

$$w(\mathrm{SF}) - w(A_{i-1}) \;=\; \sum_{u \in U} w_u(\mathrm{SF}) - \sum_{u \in U} w_u(A_{i-1}) \;\leq\; \sum_{u \in U_{i-1}} (w_u(\mathrm{SF}) - w_u(A_{i-1})) \; .$$

Now, since the users in U_{i-1} are not saturated, it follows that A_{i-1} gains $\sum_{u \in U_{i-1}} w_u(S)$ utility, for every $S \in \mathcal{S}(A_{i-1})$, due to assigning it to users in U_{i-1}. Clearly SF cannot gain more from assigning streams from $\mathcal{S}(A_{i-1})$ to users in U_{i-1}. Hence,

$$\sum_{u \in U_{i-1}} (w_u(\mathrm{SF}) - w_u(A_{i-1})) \leq \sum_{u \in U_{i-1}} \Delta_{i-1}(u)$$

where $\Delta_{i-1}(u)$ is the utility gained by SF by assigning streams from $\mathcal{S}(\mathrm{SF}) \setminus \mathcal{S}(A_{i-1})$ to a user $u \in U'$.

The cost effectiveness of each stream $S \in \mathcal{S}(\mathrm{SF}) \setminus \mathcal{S}(A_{i-1})$ is at most $\bar{w}^{A_{i-1}}(S_i)/c(S_i)$, since S_i maximizes this ratio. Since $c(\mathcal{S}(\mathrm{SF}) \setminus \mathcal{S}(A_{i-1})) \leq c(\mathcal{S}(\mathrm{SF})) \leq B$, the total utility of users covered by streams in $\mathcal{S}(\mathrm{SF}) \setminus \mathcal{S}(A_{i-1})$ is at most $B \cdot \bar{w}^{A_{i-1}}(S_i)/c(S_i)$. Therefore,

$$\sum_{u \in U_{i-1}} \Delta_{i-1}(u) \leq B \cdot \frac{\bar{w}^{A_{i-1}}(S_i)}{c(S_i)} \; .$$

It follows that

$$w(\mathrm{SF}) - w(A_{i-1}) \leq B \cdot \frac{\bar{w}^{A_{i-1}}(S_i)}{c(S_i)} \; .$$

Since $\bar{w}^{A_{i-1}}(S_i) = w(A_i) - w(A_{i-1})$, we have that

$$w(A_i) - w(A_{i-1}) \geq \frac{c(S_i)}{B} \cdot (w(\mathrm{SF}) - w(A_{i-1})) \; .$$

The lemma follows. □

Lemma 2.4. *For every $i \leq k+1$, we have* $w(A_i) \geq \left(1 - \prod_{S \in \mathcal{S}(A_i)} (1 - \frac{c(S)}{B})\right) \cdot w(\mathrm{SF})$.

Proof. We prove the lemma by induction on i. The base case is $i = 1$, and we need to prove that $w(A_1) = \bar{w}^{A_0}(S_1) \geq \frac{c(S_1)}{B} \cdot w(\mathrm{SF})$. This inequality holds because for any assignment SF, $w(\mathrm{SF})/c(\mathrm{SF}) \leq \bar{w}^{A_0}(S_1)/c(S_1)$ (since S_1 maximizes the ratio $\bar{w}^{A_0}(S)/c(S)$ over all streams S), and because $w(\mathrm{SF})/c(\mathrm{SF}) \geq w(\mathrm{SF})/B$ (since SF is semi-feasible).

For the inductive step, assume that the lemma holds for $i-1$, and consider i. If $A_i = A_{i-1}$ then we are done. Otherwise, using Lemma 2.3 we get

$$\begin{aligned}
w(\mathrm{SF}) - w(A_i) &= w(\mathrm{SF}) - w(A_{i-1}) - (w(A_i) - w(A_{i-1})) \\
&\leq w(\mathrm{SF}) - w(A_{i-1}) - \frac{c(S_i)}{B}(w(\mathrm{SF}) - w(A_{i-1})) \\
&= \left(1 - \frac{c(S_i)}{B}\right) \cdot (w(\mathrm{SF}) - w(A_{i-1})) \\
&\leq \left(1 - \frac{c(S_i)}{B}\right) \cdot \prod_{S \in \mathcal{S}(A_{i-1})} \left(1 - \frac{c(S_t)}{B}\right) \cdot w(\mathrm{SF}) \\
&= \prod_{S \in \mathcal{S}(A_i)} \left(1 - \frac{c(S_t)}{B}\right) \cdot w(\mathrm{SF}) \ ,
\end{aligned}$$

and the induction step is complete. □

And now we are ready to prove Lemma 2.2.

Proof of Lemma 2.2. By Lemma 2.4 we have that

$$w(A_{k+1}) \geq \left(1 - \prod_{S \in \mathcal{S}(A_{k+1})} (1 - \frac{c(S_t)}{B})\right) \cdot w(\mathrm{SF}) \ .$$

Observe that if $\gamma_1, \ldots, \gamma_q \in \mathbb{R}^+$ satisfy $\sum_i \gamma_i \leq \Gamma$, then the maximum of the function $\prod_{i=1}^q (1 - \frac{\gamma_i}{\Gamma})$ is at $\gamma_i = \Gamma/q$ for every i. Hence,

$$\prod_{S \in \mathcal{S}(A_{k+1})} (1 - \frac{c(S_t)}{B}) \ \leq \prod_{S \in \mathcal{S}(A_{k+1})} (1 - \frac{c(S_t)}{c(A_{k+1})}) \ \leq \ \left(1 - \frac{1}{|\mathcal{S}(A_{k+1})|}\right)^{|\mathcal{S}(A_{k+1})|} \ \leq \ \frac{1}{e} \ ,$$

where the first inequality holds because $c(A_{k+1}) > B$. It follows that $w(A_{k+1}) \geq (1 - \frac{1}{e}) \cdot w(\mathrm{SF})$. □

We note that the use of the stream S_{k+1} is essential for the analysis, as otherwise, the ratio between the optimum utility and the utility of the solution computed by greedy may be unbounded. As an immediate corollary to Lemma 2.2, we state below the performance guarantee of Algorithm **Greedy** by comparing the output of the algorithm with an optimal solution that has a smaller budget.

Theorem 2.5. *Let A be the solution computed by Algorithm **Greedy**, and let OPT^- denote the utility of the optimal solution with reduced budget $B - c_{\max}$, where $c_{\max} = \max\{c(S) \mid S \in \mathcal{S}\}$. Then $w(A) \geq (1 - 1/e) \cdot \mathrm{OPT}^-$.*

2.2 Fixing the Greedy Algorithm

In Theorem 2.5, the performance of the algorithm was guaranteed only after adding the stream S_{k+1}. We now show how to modify Algorithm **Greedy** to obtain approximate assignments without resource augmentation.

First, let us explain what is the weakness of the greedy algorithm. Roughly speaking, the problem with a greedy solution is that it may assign a stream S_1 with large cost-effectiveness but low absolute utility, and S_1 may block from inclusion another stream S_2 whose cost effectiveness is slightly smaller, but whose absolute utility is much larger. For example, S_2 may require the whole bandwidth budget, so even a tiny stream S_1 that was assigned will block S_2 from being assigned.

This "hole" in the behavior of Greedy is handled by the following trick: we find the best single-stream solution, compare it to the greedy solution, and pick the best.

More formally, let $S_{\max} = \operatorname{argmax}\{w(S) \mid S \in \mathcal{S}\}$, and let $A_{\max}$ be the assignment that assigns the single stream $S_{\max}$ to all possible users. The modified algorithm computes assignment A_G by Algorithm **Greedy**, computes assignment $A_{\max}$, and outputs the better one. We denote the latter assignment by $\tilde{A}$. Note that $\tilde{A}$ may still be semi-feasible. However, it is $(\frac{2e}{e-1})$-approximate:

Lemma 2.6. $w(\tilde{A}) \geq \frac{e-1}{2e} \cdot \mathrm{OPT}$.

Proof. By Lemma 2.2, $w(A_k) + \bar{w}^{A_k}(S_{k+1}) \geq \frac{e-1}{e} \cdot \mathrm{OPT}$. Since $\bar{w}^{A_k}(S_{k+1}) \leq w(S_{k+1}) \leq w(S_{\max})$, we get that $w(A_k) + w(A_{\max}) \geq \frac{e-1}{e} \cdot \mathrm{OPT}$, and the lemma follows. □

A performance guarantee with resource augmentation follows directly:

Corollary 2.7. *There exists an algorithm that computes $(\frac{2e}{e-1})$-approximate solutions that may use a capacity of $K^u + \bar{k}^u$ for every user u, where $\bar{k}_u = \max\{k^u(S) \mid S \in \mathcal{S}\}$.*

We are also able to obtain an approximation algorithm that does not rely on resource augmentation. A crude lower bound can be obtained as follows.

Theorem 2.8. *There exists an $O(n^2)$ time $(\frac{3e}{e-1})$-approximation algorithm for the* SMD *problem.*

Proof. Consider the assignment A that was computed by the greedy algorithm. Let S_u be the last stream that was assigned to u by the greedy algorithm. Define $A_1(u) = A(u) \setminus \{S_u\}$ and $A_2(u) = \{S_u\}$, for every user u. ($A_1(u) = A_2(u) = \emptyset$ if S_u does not exist.) Clearly, $A(u) = A_1(u) \cup A_2(u)$ for every u. Both A_1 and A_2 are feasible assignments and $w(A_1) + w(A_2) \geq w(A)$. It follows that $w(A_1) + w(A_2) + w(A_{\max}) \geq (1 - 1/e) \cdot \mathrm{OPT}$, which means that one of A_1, A_2, and $A_{\max}$ achieves approximation factor of a most $\frac{3e}{e-1}$. □

2.3 Better Approximation Factor for SMD

In this section we present an algorithm that computes $(\frac{e}{e-1})$-approximate solutions with resource augmentation, or $(\frac{2e}{e-1})$-approximate solutions without resource augmentation. Our approach is based on the $(\frac{e}{e-1})$-approximation algorithm for maximization of nondecreasing submodular set functions subject to a budget constraint by Sviridenko [16]. This algorithm consists of partial enumeration combined with a greedy algorithm.

Observe that when considering semi-feasible solutions Lemma 2.1 implies that SMD is a maximization problem of nonnegative, nondecreasing, submodular and polynomially computable set functions subject to a budget constraint. It follows that

Theorem 2.9. *There exists a polynomial time algorithm that computes* $(\frac{e}{e-1})$*-approximate solutions for* SMD *that may use a capacity of* $K^u + \bar{k}^u$ *for every user* u*, where* $\bar{k}_u = \max_S k^u(S)$*.*

The proof of the next theorem is similar to the proof of Theorem 2.8.

Theorem 2.10. *There exists an polynomial-time* $(\frac{e-1}{2e})$*-approximation algorithm for the* SMD *problem.*

3 Instances with Arbitrary Skew

In this section we explain how to deal with instances of SMD with arbitrary local skew. The idea is to use the "classify and select" approach: we reduce an instance of SMD with arbitrary skew to a set of instances of SMD where each of the new instances has $O(1)$ skew, and pick the best solution over the sub-instances.

Before we present the reduction, we formally define the local skew. Given an MMD instance, scale the k_i^u functions and their corresponding capacities so that for every user u and cost measure i we have $\frac{w_u(S)}{k_i^u(S)} \geq 1$ for any stream S for which $w_s(S) > 0$, with equality for at least one stream. Given this normalization, the *local skew* of the instance is defined by

$$\alpha \stackrel{\text{def}}{=} \max_{u,S,i} \left\{ \frac{w_u(S)}{k_i^u(S)} \ : \ w_u(S) > 0 \right\} .$$

Notice that $\alpha \geq 1$ always, and equality holds if and only if all capacity functions of each user u are proportional to his utility w_u.

Now, suppose that we are given an SMD instance I with local skew α. We construct t SMD instances $I_1, \ldots, I_t$, where $t = 1 + \lfloor \log \alpha \rfloor$, where I_i is defined as follows. The streams and users are the same as in the original instance, and so are the cost function c and the budget B. We define a new utility function w_u^i for every user u:

$$w_u^i(S) = \begin{cases} k^u(S) & 2^{i-1} \leq \frac{w_u(S)}{k^u(S)} < 2^i, \\ 0 & \text{otherwise.} \end{cases}$$

That is, the ith utility function w_u^i of u only considers sets whose utility per capacity ratio is between 2^{i-1} and 2^i. We also set $W_u^i = K^u$.

Theorem 3.1. *There exists an* $O(n^2)$ *time algorithm that computes* $O(\log(2\alpha))$*-approximate solutions for any instance* SMD *with skew* α*.*

Proof. Let I be a SMD instance of skew α, and let $I_1, \ldots, I_t$ be the SMD instances that are obtained as above. Clearly, each user-stream pair appears with non-zero utility in exactly one of the SMD instances $I_1, \ldots, I_t$. Hence, $\sum_i \text{OPT}_i \geq \frac{\text{OPT}}{2}$, where OPT_i the optimum value of I_i. It follows that there exists i such that $\text{OPT}_i \geq \frac{\text{OPT}}{2t}$. Hence, by finding an approximate solution for every SMD instance I_i, and choosing the one with maximum utility, we get an $O(\log(2\alpha))$-approximate solution for I.

As for the running time, let $G = (\mathcal{S}, U, E)$ be the bipartite graph that corresponds to the problem instance I, namely where $(S, u) \in E$ if $w_u(S) > 0$. The reduction places each edge from E in exactly one of the instances $I_1, \ldots, I_t$. Hence, $\sum_i n_i = O(n)$, where n_i is the size of the instance I_i. By Theorem 2.8, an $O(1)$-approximation can be computed in $O(n_i^2)$ for every SMD instance I_i. It follows that the total running time is $O(\sum_i n_i^2) = O(n^2)$. ☐

4 Multiple Budget Constraints

In this section we show how to reduce MMD to SMD. If the server has m finite budget constraints, and a user has at most m_c budget constraints, then the reduction results in losing an approximation factor of $O(mm_c)$. The local skew may also increase by a factor of at most m_c. We remark that our technique can be used to maximize arbitrary submodular functions under multiple budget constraints, extending the results of Sviridenko [16].

4.1 Reduction from Multiple Constraints to Single Constraint

The main idea in the reduction is to normalize and add all cost measures to single cost, and similarly to normalize and add all capacity measures to single capacity for every user. Specifically, given an instance I_M of MMD, we apply the following transformation to construct an instance I_S of SMD. The users, streams, and utility functions in I_S are just the same as in I_M. The single server cost function in I_S is defined by $c(S) = \sum_{i=1}^{m} \frac{c_i(S)}{B_i}$ for each stream $S \in \mathcal{S}$, and the single budget in I_S is $B = m$. Similarly, we define in I_S the single capacity constraint of each user u by $k^u(S) = \sum_i \frac{k_i^u(S)}{K_i^u}$ and $K^u = m_c$. This concludes the description of the input transformation. The output transformation is described later.

We first bound the skew of transformed instance.

Lemma 4.1. *Let α_S and α_M denote the skews of I_S and I_M, respectively. Then $\alpha_S \le m_c \cdot \alpha_M$.*

Proof. Assume that I_M is normalized. We compute the local skew of I_S. First,

$$\frac{w_u(S)}{k^u(S)} = \frac{w_u(S)}{\sum_i \frac{k_i^u(S)}{K_i^u}} \le \frac{w_u(S)}{\frac{k_i^u(S)}{K_i^u}} = \frac{K_i^u \cdot w_u(S)}{k_i^u(S)} \le K_i^u \cdot \alpha_M$$

for every user and every i. Hence, $\frac{w_u(S)}{k^u(S)} \le K_{\min}^u \cdot \alpha_M$, where $K_{\min}^u = \min_i K_i^u$. On the other hand,

$$\frac{w_u(S)}{k^u(S)} = \frac{w_u(S)}{\sum_i \frac{k_i^u(S)}{K_i^u}} \ge \frac{w_u(S)}{\sum_i \frac{k_i^u(S)}{K_{\min}^u}} = \frac{K_{\min}^u \cdot w_u(S)}{\sum_i k_i^u(S)} \ge \frac{K_{\min}^u}{m_c} .$$

It follows that the local skew of I_S is at most $m_c \cdot \alpha_M$. ☐

Next we relate a solution to I_S to a solution to I_M.

Lemma 4.2. *Let A be an r-approximate assignment to I_S. Then (1) $c_i(A) \le m \cdot B_i$, for every i, (2) $k_i^u(A) \le m_c \cdot K_i^u$, for every u and i, and (3) $w(A) \ge \frac{\mathrm{OPT}_M}{r}$, where OPT_M is the optimum for I_M.*

Proof. To prove 1 and 2, note that the cost of a stream S is $c(S) = \sum_{i=1}^{m} \frac{c_i(S)}{B_i}$, therefore

$$\frac{c_i(A)}{B_i} = \sum_{S \in \mathcal{S}(A)} \frac{c_i(S)}{B_i} \le \sum_{S \in \mathcal{S}(A)} c(S) = c(A) \le m .$$

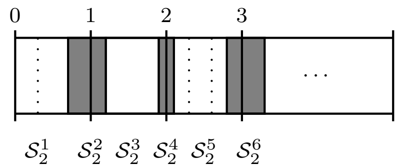

Figure 3: *Decomposition of $\mathcal{S}_2$. Shaded areas represent $\mathcal{S}_2^{2\ell}$ sets, and white areas represent $\mathcal{S}_2^{2\ell-1}$ sets. The dotted lines are boundaries between streams that belong to the same subset.*

It follows that $c_i(A) \le m \cdot B_i$ for every i. Similarly, $k_i^u(A) \le m_c \cdot K_i^u$ for every u and i. We now prove 3. Let A^* be an optimal solution for I_M. We claim that A^* is a feasible assignment to I_S. First,

$$c(A^*) = \sum_{S \in \mathcal{S}(A^*)} \sum_{i=1}^{m} \frac{c_i(S)}{B_i} = \sum_{i=1}^{m} \frac{c_i(A^*)}{B_i} \le \sum_{i=1}^{m} \frac{B_i}{B_i} = m \ .$$

Similarly, for every user u,

$$k^u(A^*) = \sum_{S \in A^*(u)} \sum_{i=1}^{m_c} \frac{k_i^u(S)}{K_i^u} = \sum_{i=1}^{m_c} \frac{k_i^u(A^*)}{K_i^u} \le \sum_{i=1}^{m_c} \frac{K_i^u}{K_i^u} = m_c \ .$$

Hence, $w(A^*) = \mathrm{OPT}_M \le \mathrm{OPT}_S$, where OPT_S is the optimum of I_S. If A is an r-approximate assignment for I_S, then $w(A) \ge \mathrm{OPT}_S/r \ge \mathrm{OPT}_M/r$, and we are done. □

Output transformation. We now explain how to transform a solution A for I_S into a feasible solution for the original I_M. Let A be an assignment for I_S. Divide $\mathcal{S}(A)$ into two sets: $\mathcal{S}_1$ contains all streams whose (single) cost is larger than 1 (i.e., $\mathcal{S}_1 = \{S \in \mathcal{S}(A) \mid c(S) \ge 1\}$), and $\mathcal{S}_2$ contains the rest of the streams. Each stream in $\mathcal{S}_1$ is a possible complete solution: such assignment is feasible since $c_i(S) \le B_i$ for every S. Formally, for each $S \in \mathcal{S}_1$ we define the assignment $A|_{\{S\}}$, where $A|_{\mathcal{C}}(u) = A(u) \cap \mathcal{C}$. These are the assignments we consider from $\mathcal{S}_1$. Note that $\sum_{S \in \mathcal{S}_1} c(S) \ge |\mathcal{S}_1|$, and therefore $\sum_{S \in \mathcal{S}_2} c(S) \le m - |\mathcal{S}_1|$.

To define the assignments based on $\mathcal{S}_2$, divide $\mathcal{S}_2$ into subsets $\mathcal{S}_2^i$ as follows (see example in Figure 3). Let each set $S_j \in \mathcal{S}_2$ be represented by an interval of length $c(S_j)$, and order these intervals consecutively along the real line starting from 0 in arbitrary order. Now consider the integer points of the real line. For each such point ℓ, there is at most one stream whose interval contains ℓ; this stream (if exists) constitute the set $\mathcal{S}_2^{2\ell}$. The streams that lie to the right of $\ell - 1$ and to the left of ℓ constitute $\mathcal{S}_2^{2\ell-1}$. Notice that since we have that $\sum_{S \in \mathcal{S}_2} c(S) \le m - |\mathcal{S}_1|$, our partition induces at most $2m - 1$ subsets of $\mathcal{S}$.

Given these $2m-1$ subsets of $\mathcal{S}_1 \cup \mathcal{S}_2$, let A_i be the restriction of the SMD assignment to the set with largest utility. By construction, A_i satisfies the server constraints (as we show), but not necessarily the user constraints. To satisfy the user constraints, we use the same approach again. Namely, for every user u, we decompose the set $A_i(u)$ into at most $2m_c - 1$ subsets that satisfy the user capacity constraints, and remove from A_i the streams that do not belong the subset of $A_i(u)$ of maximum utility. This completes the specification of the output transformation.

We summarize in the following theorem.

Theorem 4.3. *An r-approximation algorithm for* SMD *implies an $O(mm_c r)$-approximation algorithm for* MMD*.*

Proof. Let A be an r-approximate assignment with respect to I_S, and consider the transformed output. We first argue that the output is feasible. At the server's side, if the solution is from $\mathcal{S}_1$ then it is feasible being a single stream, and if the solution is from $\mathcal{S}_2$ then it is feasible because its single cost is at most 1, and therefore its normalized cost in any measure is at most 1. Similarly, no user capacity constraint is violated.

Regarding the approximation factor, since the number of assignments we consider is bounded by $2m-1$, the assignment A_i we choose has utility which is at least a $\frac{1}{2m-1}$ fraction of the utility in the solution to I_S. In the last stage, we discard streams from users to obtain assignments that adhere to user constraints, and by the same argument, we get from each user at least a $\frac{1}{2m_c-1}$ fraction of the remaining utility. The theorem follows. ◻

We note that the analysis of Theorem 4.3 is tight up to a constant factor (see Section 4.2).

Theorem 4.3 leads us to the following result:

Theorem 4.4. *There exists an $O(n^2)$ time $O(mm_c \log(2\alpha m_c))$-approximation algorithm for* MMD*, where α is the local skew of the instance, m is the number of cost measures, m_c is the maximal number of capacity constraints at a user, and n is the input length.*

Note that if each user has only a single budget constraint with local skew $\alpha = 1$ (which means that the user is only limited by the maximal utility it can generate), then our algorithm guarantees an $O(m)$ approximation. Note further that if all costs and utilities are polynomial in the input length n, then the approximation ratio is $O(mm_c \log n)$.

As a final remark for this section, we note that our approach can be used to maximize nonnegative, nondecreasing, submodular, and polynomially computable set functions under m budget constraints, obtaining an $O(m)$ approximation ratio. The idea is to execute the reduction from multiple constraints to single constraint. This will result in a single budget constraint without changing the properties of the set function. An approximate solution for this instances can be found using Sviridenko's algorithm [16], and this solution can be transformed into an $O(m)$-approximate solution for the original multiple constraints instance.

4.2 Tightness of Theorem 4.3

In this section we present an MMD instance with unit skew on which Theorem 4.3 causes a deterioration by a factor of $m \cdot m_c$ in the utility of the solution.

Consider the following linear MMD instance I_M with m budget constraints and one user with m_c capacity constraints. There are $m + m_c - 1$ streams, where

$$c_i(S_j) = \begin{cases} \frac{1}{2} + \varepsilon & i = j < m \\ \frac{\frac{1}{2}+\varepsilon}{m_c} & i = m \text{ and } j \geq m, \\ 0 & \text{otherwise,} \end{cases}$$

where ε is a small enough, e.g., $\varepsilon = \frac{1}{m^2}$. Also, $B_i = 1$ for every $i \in \{1, \ldots, m\}$. There is only one user with m_c capacity functions, where

$$k_i^u(S_j) = \begin{cases} \frac{1}{2} + \varepsilon' & j = m + i - 1 \\ 0 & \text{otherwise,} \end{cases}$$

where ε' is a small enough, e.g., $\varepsilon' = \frac{1}{m_c^2}$. Also, $K_i^u = 1$ for every $i \in \{1, \ldots, m_c\}$. Finally,

$$w_u(S_j) = \begin{cases} 1 & j < m \\ \frac{1}{m_c} & j \geq m \ . \end{cases}$$

First, observe that $A(u) = \{S_1, \ldots, S_{m+m_c-1}\}$ is an optimal solution. This is because $c_i(A) = \frac{1}{2} + \varepsilon$ for every $i \in \{1, \ldots, m\}$, and $k_i^u(A) = \frac{1}{2} + \varepsilon$ for every $i \in \{1, \ldots, m_c\}$. Hence, $\text{OPT} = m$.

Now consider the SMD instance I_S obtained by the reduction. The cost function is:

$$c(S_j) = \begin{cases} \frac{1}{2} + \varepsilon & j < m \\ \frac{\frac{1}{2} + \varepsilon}{m_c} & j \geq m, \end{cases}$$

and the budget is $B = m$. The user capacity function is:

$$k^u(S_j) = \begin{cases} \frac{1}{2} + \varepsilon' & j \geq m \\ 0 & \text{otherwise,} \end{cases}$$

and the capacity bound is $K^u = m_c$.

The decomposition of A that is outlined in the proof of Theorem 4.3 may put the streams $S_m, \ldots, S_{m+m_c-1}$ in a single set $\mathcal{S}_2^1$ and every stream S_j for $j < m$ in a different set $\mathcal{S}_2^j$, and only one of the sets will survive. Say that $\mathcal{S}_2^1$ survives. In this case, we now turn to the decomposition of $A_1(u)$. Since $k^u(S_j) = \frac{1}{2} + \varepsilon$ for every $S_j \in A_1(u)$, it follows that only one stream S from $A_1(u)$ survives. Since $w_u(S) = \frac{1}{m_c}$ it follows that the utility of the computed stream is $\frac{\text{OPT}}{m m_c}$.

5 Allocating Small Streams

In this section we present an approximation algorithm for small streams. Specifically, when all numbers in the input are polynomial in n, the algorithm provides $O(\log n)$-approximate assignment for the case where each stream has cost which is at most a $O(1/\log n)$ fraction of each budget, and at most $O(1/\log n)$ fraction of each capacity. The algorithm we present is an online algorithm: it considers streams one by one as they arrive, and decides whether to supply the stream and to which users, without knowledge about future arrivals. Our algorithm is based on the work of Awerbuch, Azar, and Plotkin [3].

We focus on the special case of MMD where $m_c = 1$. The extension to the case of $m_c > 1$ is straightforward.

For the sake of brevity, we assume that for every user capacity function k^u, there exists a virtual cost function c_u such that $c_u(S) = k^u(S)$ for every S, and a virtual budget $B_u = K^u$. We denote the original set of budgets by M and we abuse notation by treating U as a set of users and also as a set of budgets.

We first generalize the local skew as follows. Given an MMD instance, normalize the costs such that

$$1 \le \frac{1}{m+|U|} \cdot \frac{\sum_{u \in X} w_u(S)}{c_i(S)} \le \gamma \ , \tag{1}$$

for any stream $S \in \mathcal{S}$, user set $X \subseteq \{u : w_u(S) > 0\}$, and cost function $i \in M \cup U$, such that $c_i(S) > 0$, where γ is as small as possible. The upper bound γ is called the *global skew* of the instance. The global skew γ bounds the ratio between the best and the worst streams, in terms of utility for each unit cost. Note that $\gamma \ge \alpha$ for all instances of MMD. Finally, we define

$$\mu \stackrel{\text{def}}{=} 2\gamma(m+|U|)+2 \ .$$

Given an assignment A, the *normalized load* on budget $i \in M$ incurred by A is $L_A(i) \stackrel{\text{def}}{=} \frac{1}{B_i}\sum_{S \in \mathcal{S}(A)} c_i(S)$, and similarly, for $u \in U$, the normalized load is $L_A(u) \stackrel{\text{def}}{=} \frac{1}{B_u}\sum_{S \in A(u)} c_u(S)$. We also define the *exponential cost function* of budget $i \in M \cup U$ by $C_A(i) \stackrel{\text{def}}{=} B_i(\mu^{L_A(i)} - 1)$.

Let $S_1, \dots, S_n$ be an arbitrary order of the streams. Algorithm **Allocate**, given formally below, starts with the empty assignment $A_0(u) = \emptyset$ for every u. Then for every stream S_j, it decides whether to allocate it and to which users, according to the exponential cost functions. Note that the maximal subset U_j may be obtained by starting with U and removing clients in decreasing order of $\frac{c_u(S_j)}{B_u} \cdot C_{A_{j-1}}(u)/w_u(S_j)$.

Since the order in which the algorithm considers streams is arbitrary, and since decisions are never revoked, the algorithm can be applied in an on-line scenario, where future requests are unknown.[1]

Algorithm 2 - Allocate$(U, \mathcal{S}, c, B, w)$

$A_0(u) = \emptyset$ for every u
for $j = 1$ to n **do**
  Let $C_{A_{j-1}}(i) = B_i[\mu^{L_{A_{j-1}}(i)} - 1]$ for every $i \in M \cup U$.
  if there exists a maximal (inclusion wise) subset $\emptyset \subsetneq U_j \subseteq U$ such that

$$\sum_{i \in M \cup U_j} \frac{c_i(S_j)}{B_i} \cdot C_{A_{j-1}}(i) \le \sum_{u \in U_j} w_u(S_j)$$

  then
    Assign S_j to the users in U_j: if $u \in U_j$ then $A_j(u) = A_{j-1}(u) \cup \{S_j\}$; otherwise $A_j(u) = A_{j-1}(u)$.
  else
    $A_j = A_{j-1}$
  end if
end for

We start out analysis by showing that the algorithm computes feasible assignments.

Lemma 5.1. *If $c_i(S) \le \frac{B_i}{\log \mu}$ for all i and S, then no budget constraints are ever violated.*

[1] The algorithm can also be extended to scenarios where streams have dynamic resource requirements, so long as their requirements are known when they arrive. This includes, for example, streams of finite duration. Details are similar to the algorithm of [3].

Proof. By contradition. Let S_j be the first stream that caused the normalized load on some budget i to exceed 1. This means that $L_{A_{j-1}(i)} > 1 - \frac{c_i(S_j)}{B_i} \geq 1 - \frac{1}{\log \mu}$. It follows that

$$\frac{C_{A_{j-1}(i)}}{B_i} = \mu^{L_{A_{j-1}}(i)} - 1 > \mu^{1-\frac{1}{\log \mu}} - 1 = 2^{\log \mu - 1} - 1 = \frac{\mu}{2} - 1 = \gamma(m + |U|) \ .$$

Hence, by the RHS of (1) we get that

$$\frac{c_i(S_j)}{B_i} \cdot C_{A_{j-1}}(i) > \gamma(m+|U|) \cdot c_i(S_j) \geq \sum_{u \in U_j} w_u(S_j)$$

which means that stream S_j could not have been assigned to U_j. □

We show that the approximation ratio of the algorithm is $O(1 + 2\log \mu)$ if $c_i(S) \leq \frac{B_i}{\log \mu}$ for every stream S and $i \in M \cup U$. Let $C_j = \sum_{i \in M \cup U} C_{A_j}(i)$ for every j. Below we first show that the utility gained by the algorithm is an $\Omega(\frac{1}{\log \mu})$ fraction of C_n, and then we show that the additional utility gained by any assignment is at most C_n.

Lemma 5.2. *Let A be the assignment that is computed by the algorithm. Then $C_n \leq 2\log \mu \cdot w(A)$.*

Proof. We prove that $C_j \leq 2 \log \mu \cdot w(A_j)$ by induction on j. The base case of $j = 0$ is trivial. For the inductive step, let S_j be a stream that was assigned to the users in U_j. Using induction, suffices to prove that

$$C_j - C_{j-1} \leq 2 \log \mu \cdot \sum_{u \in U_j} w_u(S_j) \ .$$

Clearly,

$$C_j - C_{j-1} = \sum_{i \in M \cup U_j} \left(C_{A_j}(i) - C_{A_{j-1}}(i) \right) \ .$$

Now, if the normalized load of budget i was not changed due to S_j, then $C_{A_j}(i) = C_{A_{j-1}}(i)$. Otherwise,

$$C_{A_j}(i) - C_{A_{j-1}}(i) = B_i \cdot \left(\mu^{L_{A_j}(i)} - \mu^{L_{A_{j-1}}(i)} \right) = B_i \cdot \mu^{L_{A_{j-1}}(i)} \left(2^{\log \mu \cdot c_i(S_j)/B_i} - 1 \right) \ .$$

Since $c_i(S_j) \leq \frac{B_i}{\log \mu}$ and $2^x - 1 \leq x$ for $x \in [0,1]$, it follows that

$$\begin{aligned} C_{A_j}(i) - C_{A_{j-1}}(i) &\leq B_i \cdot \mu^{L_{A_{j-1}}(i)} \left(\log \mu \cdot \frac{c_i(S_j)}{B_i} \right) \\ &= \log \mu \cdot \mu^{L_{A_{j-1}}(i)} c_i(S_j) \\ &= \log \mu \left(C_{A_{j-1}(i)} \cdot \frac{c_i(S_j)}{B_i} + c_i(S_j) \right) \ . \end{aligned}$$

Combined with the condition of Line 4 of the algorithm and the LHS of (1), we obtain

$$\begin{aligned} C_j - C_{j-1} &\leq \log \mu \cdot \sum_{i \in M \cup U_j} \left(\frac{c_i(S_j)}{B_i} \cdot C_{A_{j-1}(i)} + c_i(S_j) \right) \\ &\leq \log \mu \cdot \left(\sum_{u \in U_j} w_u(S_j) + \sum_{i \in M \cup U_j} c_i(S_j) \right) \\ &\leq 2 \log \mu \cdot \sum_{u \in U_j} w_u(S_j) \ , \end{aligned}$$

as required. □

Lemma 5.3. *Let A^* be an optimal assignment. Then $w(A^*) - w(A) \leq C_n$.*

Proof. Consider the stream S_j and let U_j^* be the set of users u such that $S_j \in A^*(u) \setminus A_{j-1}(u)$. Observe that if the algorithm did not assign S_j to any user ($U_j = \emptyset$), then $U_j^* = \{u : S_j \in A^*(u)\}$. By the maximality of U_j,

$$\sum_{u \in U_j \cup U_j^*} w_u(S_j) \quad < \sum_{i \in M \cup U_j \cup U_j^*} \frac{c_i(S_j)}{B_i} \cdot C_{A_{j-1}}(i) \ .$$

If $U_j \neq \emptyset$, then by the condition of Line 4 we know that $\sum_{u \in U_j} w_u(S_j) \geq \sum_{i \in M \cup U_j} \frac{c_i(S_j)}{B_i} \cdot C_{A_{j-1}}(i)$. Hence,

$$\sum_{u \in U_j^*} w_u(S_j) \ < \sum_{i \in U_j^*} \frac{c_i(S_j)}{B_i} \cdot C_{A_{j-1}}(i) \ .$$

Otherwise, if S_j was disregarded by the algorithm, then

$$\sum_{u \in U_j^*} w_u(S_j) \quad < \sum_{i \in M \cup U_j^*} \frac{c_i(S_j)}{B_i} \cdot C_{A_{j-1}}(i) \ .$$

It follows that

$$\begin{aligned}
w(A^*) - w(A) &\leq \sum_j \sum_{i \in M \cup U_j^*} \frac{c_i(S_j)}{B_i} \cdot C_{A_{j-1}}(i) \\
&\leq \sum_j \sum_{i \in M \cup U_j^*} \frac{c_i(S_j)}{B_i} \cdot C_{A_n}(i) \\
&\leq \sum_{i \in M} C_{A_n}(i) \sum_{S_j \in \mathcal{S}(A^*)} \frac{c_i(S_j)}{B_i} \ + \ \sum_{i \in U} C_{A_n}(i) \sum_{S_j \in A^*(i)} \frac{c_i(S_j)}{B_i} \\
&\leq \sum_{i \in M \cup U} C_{A_n}(i) \\
&= C_n \ ,
\end{aligned}$$

where the last inequality is due to the fact that the optimal solution A^* must satisfy the budget constraints. □

Theorem 5.4. *Algorithm **Allocate** computes $(1 + 2\log\mu)$-approximate solutions in the case where $c_i(S) \leq \frac{B_i}{\log \mu}$ for every stream S and cost measure i.*

Proof. By the previous two lemmas is follows that $w(A^*) - w(A) \leq 2\log\mu \cdot w(A)$. Hence, $w(A^*) \leq (1 + 2\log\mu) \cdot w(A)$. □

Note that γ is polynomial in n if all numbers in the input are polynomial in n. In this case the approximation ratio is $O(\log n)$.

Acknowledgments

The authors would like to thank Baruch Awerbuch for very useful discussions, and the anonymous reviewers who intercepted a few errors in the manuscript.

References

[1] A. A. Ageev and M. Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328, 2004.

[2] N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms*, 2(2):153–177, 2006.

[3] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *34th IEEE Symp. on Foundations of Computer Science*, pages 32–40, 1993.

[4] N. Benameur, S. Ben Fredj, F. Delcoigne, S. Oueslati-Boulahia, and J. W. Roberts. Integrated admission control for streaming and elastic traffic. In *Quality of Future Internet Services*, volume 2156 of *LNCS*, pages 69–81. Springer, 2001.

[5] C. Bewick, R. Pereira, and M. Merabti. Admission control and routing of smoothed video streams. *International Journal of Simulation, Systems, Science and Technology*, 6(3-4):61–68, 2005.

[6] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications. In *7th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, volume 3122 of *LNCS*, pages 72–83, 2004.

[7] Cisco Systems, Inc. Integrated video admission control for the delivery of quality video experience. Cisco Systems White Paper, 2006.

[8] Cisco Systems, Inc. The evolving IPTV service architecture. Cisco Systems White Paper, 2007.

[9] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[10] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. Analysis of approximation algorithms for maximizing submodular set function II. *Mathematical Programming Study*, 8:73–87, 1978.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[12] D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problem*. PWS Publishing Company, 1997.

[13] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[14] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, Inc., 1988.

[15] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions I. *Mathematical Programming*, 14(1):265–294, 1978.

[16] M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[17] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin Heidelberg New York, 2001.