Online Multistage Subset Maximization Problems

Evripidis Bampis
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
evripidis.bampis@lip6.fr

Bruno Escoffier
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
bruno.escoffier@lip6.fr

Kevin Schewior
Fakultät für Informatik, Technische Universität München, Germany
Départment d'Informatique, École Normale Supérieure Paris, PSL University, France
kschewior@gmail.com

Alexandre Teiller
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
alexandre.teiller@lip6.fr

Abstract

Numerous combinatorial optimization problems (knapsack, maximum-weight matching, etc.) can be expressed as *subset maximization problems*: One is given a ground set $N = \{1, \ldots, n\}$, a collection $\mathcal{F} \subseteq 2^N$ of subsets thereof such that $\emptyset \in \mathcal{F}$, and an objective (profit) function $p : \mathcal{F} \to \mathbb{R}_+$. The task is to choose a set $S \in \mathcal{F}$ that maximizes $p(S)$. We consider the *multistage* version (Eisenstat et al., Gupta et al., both ICALP 2014) of such problems: The profit function p_t (and possibly the set of feasible solutions $\mathcal{F}_t$) may change over time. Since in many applications changing the solution is costly, the task becomes to find a sequence of solutions that optimizes the trade-off between good per-time solutions and stable solutions taking into account an additional similarity bonus. As similarity measure for two consecutive solutions, we consider either the size of the intersection of the two solutions or the difference of n and the Hamming distance between the two characteristic vectors.

We study multistage subset maximization problems in the *online* setting, that is, p_t (along with possibly $\mathcal{F}_t$) only arrive one by one and, upon such an arrival, the online algorithm has to output the corresponding solution without knowledge of the future.

We develop general techniques for online multistage subset maximization and thereby characterize those models (given by the type of data evolution and the type of similarity measure) that admit a constant-competitive online algorithm. When no constant competitive ratio is possible, we employ lookahead to circumvent this issue. When a constant competitive ratio is possible, we provide almost matching lower and upper bounds on the best achievable one.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Online algorithms

Keywords and phrases Multistage optimization, Online algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2019.11

Related Version A full version of the paper is available at [7], `http://arxiv.org/abs/1905.04162`.

Funding *Kevin Schewior*: Supported by the DAAD within the PRIME program using funds of BMBF and the EU Marie Curie Actions.

Acknowledgements This research benefited from the support of FMJH program PGMO and from the support of EDF-Thalès-Orange.

1 Introduction

In a classical combinatorial optimization setting, given an instance of a problem one needs to find a good feasible solution. However, in many situations, the data may evolve over time and one has to solve a sequence of instances. The natural approach of solving every instance independently may induce a significant transition cost, for instance for moving a system from one state to another. This cost may represent, e.g., the cost of turning on/off the servers in a data center [17, 8, 4, 1], the cost of changing the quality level in video streaming [16], or the cost for turning on/off nuclear plants [24]. Gupta et al. [15] and Eisenstat et al. [13] proposed a *multistage* model where given a time horizon $t = 1, 2, \dots, T$, the input is a sequence of instances $I_1, I_2, \dots, I_T$, (one for each time step), and the goal is to find a sequence of solutions $S_1, S_2, \dots, S_T$ (one for each time step) reaching a trade-off between the quality of the solutions in each time step and the stability/similarity of the solutions in consecutive time steps. The addition of the transition cost makes some classic combinatorial optimization problems much harder. This is the case for instance for the minimum weighted perfect matching problem in the *off-line* case where the whole sequence of instances is known in advance. While the one-step problem is polynomially-time solvable, the multistage problem becomes hard to approximate even for bipartite graphs and for only two time steps [5, 15].

In this work, we focus on the *online* case, where at time t no knowledge is available for instances at times $t+1, \dots, T$. When it is not possible to handle the online case, we turn our attention to the *k-lookahead* case, where at time t the instances at times $t+1, \dots, t+k$ are also known. This case is of interest since in some applications like in dynamic capacity planning in data centers, the forecasts of future demands may be very helpful [18, 19]. Our goal is to measure the impact of the lack of knowledge of the future on the quality and the stability of the returned solutions. Indeed, our algorithms are limited in their knowledge of the sequence of instances. Given that the number of time steps is given, we compute the competitive ratio of the algorithm after time step T: As we focus on maximization problems, we say that a (deterministic) algorithm is (strictly) α-competitive (with competitive ratio α) if its value is at least $\frac{1}{\alpha}$ times the optimal value on all instances.

As it is usual in the online setting, we consider no limitations in the computational resources available. This implies that at every time step t, where instance I_t is known, we assume the existence of an oracle able to compute the optimal solution for that time step. Notice also that our lower bounds do not rely on any complexity assumptions. Some recent results are already known for the online multistage model [6, 15], however all these results are obtained for specific problems. In this work, we study multistage variants of a broad family of maximization problems. The family of optimization problems that we consider is the following.

▶ **Definition 1** (Subset Maximization Problems). *A Subset Maximization problem $\mathcal{P}$ is a combinatorial optimization problem whose instances $I = (N, p, \mathcal{F})$ consist of*

- *A ground set N;*
- *A set $\mathcal{F} \subseteq 2^N$ of feasible solutions such that $\emptyset \in \mathcal{F}$;*
- *A non-negative weight $p(S)$ for every $S \in \mathcal{F}$.*

The goal is to find $S^* \in \mathcal{F}$ such that $p(S^*) = \max\{p(S) : S \in \mathcal{F}\}$.

We will consider that the empty set is always feasible, ensuring that the feasible set of solutions is non empty. This is a very general class of problems, including the maximization *Subset Selection* problems studied by Pruhs and Woeginger in [23] (they only considered linear objective functions). It contains for instance graph problems where N is the set of vertices (as in any maximization induced subgraph problem verifying some property) or the set of edges (as in matching problems). It also contains classical set problems (knapsack, maximum 3-dimensional matching,...), and more generally 0-1 linear programs (with non negative profits in the objective function).

Given a problem in the previous class, we are interested in its multistage version [15, 13]. The stability over time of a solution sequence is classically captured by considering a transition cost when a modification is made in the solution. Here, dealing with maximization problems, we will consider a transition *bonus* B for taking into account the similarity of two consecutive solutions. In what follows, we will use the term object to denote an element of N (so an object can be a vertex of a graph, or an edge,..., depending on the underlying problem).

▶ **Definition 2** (Multistage Subset Maximization Problems). *In a Multistage Subset Maximization problem $\mathcal{P}$, we are given*

- *a number of steps $T \in \mathbb{N}$, a set N of n objects;*
- *for any $t \in T$, an instance I_t of the optimization problem. We will denote:*
  - *p_t the objective (profit) function at time t*
  - *$\mathcal{F}_t \in 2^N$ the set of feasible solutions at time t*
- *$B \in \mathbb{R}^+$ a given transition profit.*
- *the value of a solution sequence $\mathcal{S} = (S_1, \ldots, S_T)$ is*

$$f(\mathcal{S}) = \sum_{t=1}^{T} p_t(S_t) + \sum_{t=1}^{T-1} b(S_t, S_{t+1})$$

  where $b(S_t, S_{t+1})$ is the transition bonus for the solution between time steps t and $t+1$. We will use the term profit for $p_t(S_t)$, bonus for the transition bonus $b(S_t, S_{t+1})$, and value of a solution $\mathcal{S}$ for $f(\mathcal{S})$;
- *the goal is to determine a solution sequence of maximum value.*

The fact that T is known may be regarded as rather uncommon the field of online algorithms. At the end of Subsection 1.2, we relate it to our results and justify it.

There are two natural ways to define the transition bonus. We will see that these two ways of measuring the stability induce some differences in the competitive ratios one can get.

▶ **Definition 3** (Types of transition bonus). *If S_t and S_{t+1} denote, respectively, the solutions for time steps t and $t+1$, then we can define the transition bonus as:*

- Intersection Bonus: *B times $|S_t \cap S_{t+1}|$: in this case the bonus is proportional to the number of objects in the solution at time t that remain in it at time $t+1$.*
- Hamming Bonus: *B times $|S_t \cap S_{t+1}| + |\overline{S_t} \cap \overline{S_{t+1}}|$. Here we get the bonus for each object for which the decision (to be in the solution or not) is the same between time steps t and $t+1$. In other words, the bonus is proportional to $|N|$ minus the number of modifications (Hamming distance) in the solutions.*

Note that by scaling profits (dividing them by B), we can arbitrarily fix $B = 1$. So from now on, we assume $B = 1$.

In this article, we will consider two possible ways for the data to evolve.

▶ **Definition 4** (Types of data evolution).
- Static Set of Feasible Solutions (SSFS): *only profits may change over time, so the structure of feasible solutions remains the same: $\mathcal{F}_t = \mathcal{F}$ for all t.*
- General Evolution (GE): *any modification in the input sequence is possible. Both the profits and the set of feasible solutions may change over time. In this latter model, for knapsack, profits and weights of object (and the capacity of the bag) may change over time; for maximum independent set edges in the graph may change,....*

1.1 Related Work

A series of papers consider the online or semi-online settings, where the input changes over time and the algorithm has to modify (re-optimize) the solution by making as few changes as possible (see [3, 9, 12, 14, 20, 21] and the references therein). The multistage model considered in this paper has been introduced in Eisenstat et al. [13] and Gupta et al. [15]. Eisenstat et al. [13] studied the multistage version of facility location problems. They proposed a logarithmic approximation algorithm. An et al. [2] obtained constant factor approximation algorithms for some related problems. Gupta et al. [15] studied the Multistage Maintenance Matroid problem for both the offline and the online settings. They presented a logarithmic approximation algorithm for this problem, which includes as a special case a natural multistage version of Spanning Tree. They also considered the online version of the problem and they provide an efficient randomized competitive algorithm against any oblivious adversary. The same paper also introduced the study of the Multistage Minimum Perfect Matching problem for which they proved that it is hard to approximate even for a constant number of stages. Bampis et al. [5] improved this negative result by showing that the problem is hard to approximate even for bipartite graphs and for the case of two time steps. When the edge costs are metric within every time step they proved that the problem remains APX-hard even for two time steps. They also showed that the maximization version of the problem admits a constant factor approximation algorithm, but is APX-hard. Olver et al. [22] studied a multistage version of the Minimum Linear Arrangement problem, which is related to a variant of the List Update problem [25], and provided a logarithmic lower bound for the online version and a polylogarithmic upper bound for the offline version.

The Multistage Max-Min Fair Allocation problem has been studied in the offline and the online settings in [6]. This problem corresponds to a multistage variant of the Santa Klaus problem. For the off-line setting, the authors showed that the multistage version of the problem is much harder than the static one. They provided constant factor approximation algorithms for the off-line setting. For the online setting they proposed a constant competitive ratio for SSFS-type evolving instances and they proved that it is not possible to find an online algorithm with bounded competitive ratio for GE-type evolving instances. Finally, they showed that in the 1-lookahead case, where at time step t we know the instance of time step $t + 1$, it is possible to get a constant approximation ratio.

Buchbinder et al. [11] and Buchbinder, Chen and Naor [10] considered a multistage model and they studied the relation between the online learning and competitive analysis frameworks, mostly for fractional optimization problems.

1.2 Summary of Results and Overview

The contribution of our paper is a framework for online multistage maximization problems (comprising different models), a characterization of those models in which a constant competitive ratio is achievable, and almost tight upper and lower bounds on the best-possible competitive ratio for these models. The focus here is on deterministic algorithms.

We increase the complexity of the considered models over the course of the paper. We start with the arguably simplest model: Considering a static set of feasible solutions clearly restricts the general model of evolution; while such a straightforward comparison between the Hamming and intersection bonus is not possible, the Hamming bonus seems simpler in that, compared to the intersection model, there are (somewhat comparable) extra terms added on the profit of both the algorithm and the optimum. As we show in Subsection 2.1,

Table 1 Our bounds on the best-possible competitive ratio $c^\star$ for the different models. The Landau symbol is with respect to $T \to \infty$.

	static set of feasible solutions	general evolution
Hamming bonus	$2 - o(1) \leq c^\star \leq 2$ Theorems 6 and 5	$1 + \sqrt{2} \leq c^\star \leq 3 + o(1)$ Theorems 12 and 10
Intersection bonus	$2 \leq c^\star \leq 2 + o(1)$ Theorems 9 and 8	$c^\star = \infty$ $c^\star = 4$ for 1-lookahead Theorems 14, 16, and 15

there is indeed a simple 2-competitive algorithm: At each time t, it greedily chooses the set S_t that either maximizes the transition bonus w.r.t. S_{t-1} (that is, choosing $S_t = S_{t-1}$, which is possible in this model) or maximizes the value $p_t(S_t)$. We complement this observation with a matching lower bound only involving two time steps.

We then toggle the transition-bonus model and the data-evolution model separately and show that constant competitive ratios can still be achieved. First, in Subsection 2.2, we consider intersection bonus. We show that, *after* modifying the profits (internally) to make larger solutions more profitable, a $(2 + 1/(T-1))$-competitive algorithm can be achieved by a greedy approach again. We also give an (almost matching) lower bound of 2 again. Next, we toggle the evolution model. In Subsection 3.1, we adapt the greedy algorithm from Subsection 2.1 by reweighting to obtain a $(3 + 1/(T-1))$-competitive algorithm using a more complicated analysis. We complement this result with a lower bound of $1 + \sqrt{2}$.

In Subsection 3.2, we finally consider the general-evolution model with intersection bonus, where we give a simple lower bound showing that a constant-competitive ratio is not achievable. This lower bound relies on forbidding to choose any item in the second step that the algorithm chose in the first step. We circumnavigate such issues by allowing the algorithm a lookahead of one step and present a 4-competitive algorithm for that setting. A similar phase transition has been observed for a related problem [6], but our algorithm, based on a doubling approach, is different. We also give a matching lower bound of 4 on the competitive ratio of any algorithm in the same setting. We summarize all results described thus far in Table 1.

We note that the lower bounds mentioned for the Hamming model are only shown for a specific fixed number of time steps, and that in general there is no trivial way of extending these bounds to a larger number of time steps. One may however argue that the large-T regime is in fact the interesting one for both practical applications and in theory, the latter because the effect of having a first time step without bonus vanishes. At the end of the respective sections, we therefore give asymptotical lower bounds of $3/2$ and roughly 1.696 for the cases of a static set of feasible solutions and general evolutions, respectively. These bounds are non-trivial, but we do not know if they are tight.

It is plausible that the aforementioned upper bounds can be improved if extra assumptions on characteristics of the objective function and the sets of feasible solutions are made. In Subsubsection 3.1.2, we show that already very natural assumptions suffice: Assuming that at each time the feasible solutions are closed under taking subsets and the objective function is subadditive, we give a $(21/8 + o(1))$-competitive algorithm for the model with a general evolution and Hamming bonus, improving the previous $(3 + o(1))$-competitive ratio. Our lower bounds for general evolution and Hamming bonus in fact fulfill the extra assumptions.

We observe that all our algorithms except for the one discussed in Subsection 2.1 require that T is known in that their behavior in the last step is different from the behavior in the steps before. This assumption is crucial: In all these models, there are examples in which one can in the first time step choose either a small profit or a potentially large bonus not knowing if there is another timestep to realize the bonus. Such examples imply a superconstant lower bound on the competitive ratio in these models. This justifies our assumption that T is known.

In Section 4, we summarize our results and mention directions for future research that we consider interesting.

Due to space constraints, some proofs only appear in the full version [7].

2 Model of a Static Set of Feasible Solutions

We consider here the model of evolution where only profits change over time: $\mathcal{F}_t = \mathcal{F}$ for any t. We first consider the Hamming bonus model and show a simple 2-competitive algorithm. We will then show that a (asymptotic) competitive ratio of 2 can also be achieved in the intersection bonus model using a more involved algorithm. In both cases, this ratio 2 is shown to be (asymptotically) optimal.

2.1 Hamming-Bonus Model

▶ **Theorem 5.** *In the SSFS model with Hamming bonus, there is a 2-competitive algorithm.*

Proof. We consider the very simple following algorithm. At each time step t, the algorithm computes an optimal solution S_t^* with associated profit $p_t(S_t^*)$. At $t = 1$ we fix $S_1 = S_1^*$. For $t > 1$, if $p_t(S_t^*) > n$ then fix $S_t = S_t^*$, otherwise fix $S_t = S_{t-1}^*$ (which is possible thanks to the fact that the set of feasible solutions does not change).

Let f^* be the optimal value. Since any solution sequence gets profit at most $p_t(S_t^*)$ at time t, and bonus at most n between two consecutive time steps, we get $f^* \leq \sum_{t=1}^{T} p(S_t^*) + n(T-1)$.

By construction, at time $t > 1$, either the algorithm gets profit $p_t(S_t^*)$ when $p_t(S_t^*) > n$, or bonus (from $t-1$) n when $n \geq p_t(S_t^*)$. So in any case the algorithm gets profit plus bonus at least $\frac{p_t(S_t^*)+n}{2}$. At time 1 it gets profit at least $p_1(S_1^*)$. So

$$f(S_1 \ldots, S_T) \geq p_1(S_1^*) + \sum_{t=2}^{T} \frac{p_t(S_t^*)}{2} + \frac{n(T-1)}{2} \geq \frac{f^*}{2},$$

which completes the proof. ◀

▶ **Theorem 6.** *Consider the SSFS model with Hamming bonus. For any $\epsilon > 0$, there is no $(2-\epsilon)$-competitive algorithm, even if there are only 2 time steps.*

Proof. We consider a set $N = \{1, 2, \ldots, n\}$ of $n = 1 + \lceil \frac{1}{\epsilon} \rceil$ objects, $T = 2$ time steps, and an additive profit function. There are three feasible solutions: $S^0 = \emptyset$, $S^1 = \{1\}$ and $S^2 = \{2, \ldots, n\}$. At $t = 1$, all the profits are 0. Let us consider an online algorithm A. We consider the three possibilities for the algorithm at time 1:

- At time 1, A chooses S^0: at time 2 we give profit 1 to all objects. If A takes no object at time 2, it gets profit 0 and bonus n. If it takes S^1, it gets profit 1 and bonus $n-1$. If it takes S^2, it gets profit $n-1$ and bonus 1, so in any case the computed solution has value n. The solution consisting of taking S^2 at both time steps has profit $n-1$ and bonus n, so value $2n-1$.

- At time 1, A chooses S^1: at time 2 we give profit 0 to object 1, and profit 1 to all other objects. Then, if the algorithm takes S^0 (resp, S^1, S^2), at time 2 its gets value $n-1$ (resp, n, $n-1$) while the solution consisting of taking S^2 at both time steps has value $2n-1$.
- At time 1, A chooses S^2: at time 2 we give profit n to object 1, and 0 to all other objects. Then if the algorithm takes S^0 (resp, S^1, S^2) at time 2 its gets value 1 (resp, n, n), while the solution consisting of taking S^1 at both time steps has value $2n$.

In any case, the ratio is at least $\frac{2n-1}{n} = 2 - \frac{1}{n} > 2 - \epsilon$. ◀

We complement this lower bound with an asymptotical result for large T; the proof is provided in the full version.

▶ **Theorem 7.** *Consider the SSFS model with Hamming bonus. For every $\epsilon > 0$, there is a T_ϵ such that, for each number of time steps $T \geq T_\epsilon$, there is no $(3/2 - \epsilon)$-competitive algorithm.*

2.2 Intersection-Bonus Model

In the intersection-bonus model things get harder since an optimal solution S_t^* may be of small size and then gives very small (potential) bonus for the next step. As a matter of fact, the algorithm of the previous section has unbounded competitive ratio in this case: take a large number n of objects, $\mathcal{F} = 2^N$, and at time 1 all objects have profit 0 up to one which has profit ϵ. The algorithm will take this object (instead of taking $n-1$ objects of profit 0) and then potentially get bonus at most 1 instead of $n-1$.

Thus we shall put an incentive for the algorithm to take solutions of large size, in order to have a chance to get a large bonus. We define the following algorithm called `MP-Algo` (for Modified Profit algorithm). Informally, at each time step t, the algorithm computes an optimal solution with a modified objective function p'_t. These modifications take into account (1) the objects taken at time $t-1$ (2) an incentive to take a lot of objects. Formally, `MP-Algo` works as follows:

1. At $t = 1$: let $p'_1(S) = p_1(S) + |S|$. Choose S_1 as an optimal solution for the problem with modified profits p'_1.
2. For t from 2 to $T-1$: let $p'_t(S) = p_t(S) + |S \cap S_{t-1}| + |S|$. Choose S_t as an optimal solution for the problem with modified profit function p'_t.
3. At $t = T$: let $p'_T(S) = p_T(S) + |S \cap S_{T-1}|$. Choose S_T as an optimal solution with modified profit function p'_T.

The cases $t = 1$ and $t = T$ are specific since there is no previous solution for $t = 1$, and no future solution for $t = T$.

▶ **Theorem 8.** *In the SSFS model with intersection bonus, `MP-Algo` is $\left(\frac{2}{1-1/(T-1)}\right)$-competitive.*

Proof. Let $(\hat{S}_1, \ldots, \hat{S}_T)$ be an optimal sequence. Since S_t is optimal with respect to p'_t, for $t = 2, \ldots, T-1$ we have:

$$p'_t(S_t) = p_t(S_t) + |S_t \cap S_{t-1}| + |S_t| \geq p'_t(\hat{S}_t) \geq p_t(\hat{S}_t) + |\hat{S}_t|. \tag{1}$$

Since S_{t-1} is also a feasible solution at time t, we have:

$$p'_t(S_t) = p_t(S_t) + |S_t \cap S_{t-1}| + |S_t| \geq p_t(S_{t-1}) \geq 2|S_{t-1}|. \tag{2}$$

Similarly, at $t = T$ $p'_T(S) = p_T(S) + |S \cap S_{t-1}|$ so

$$\begin{aligned} p_T(S_T) + |S_T \cap S_{T-1}| &\geq p_T(\hat{S}_T), & (3)\\ p_T(S_T) + |S_T \cap S_{T-1}| &\geq |S_{T-1}|. & (4) \end{aligned}$$

At $t = 1$, $p'_t(S) = p_t(S) + |S|$, so

$$p_1(S_1) + |S_1| \geq p_1(\hat{S}_1) + |\hat{S}_1|. \tag{5}$$

Now, note that $|S_t \cap S_{t-1}|$ is the transition bonus of the computed solution between $t-1$ and t. By summing Equation (1) for $t = 2, \ldots, T-1$, Equation (3) and Equation (5), we deduce:

$$f(S_1, \ldots, S_T) + \sum_{t=1}^{T-1} |S_t| \geq \sum_{t=1}^{T} p_t(\hat{S}_t) + \sum_{t=1}^{T-1} |\hat{S}_t|.$$

Since in the optimal sequence the transition bonus between time t and $t+1$ is at most $|\hat{S}_t|$, we get:

$$f(S_1, \ldots, S_T) + \sum_{t=1}^{T-1} |S_t| \geq f(\hat{S}_1, \ldots, \hat{S}_T). \tag{6}$$

Now we sum Equation (2) for $t = 2, \ldots, T-1$ and Equation (4):

$$f(S_1, \ldots, S_T) + \sum_{t=2}^{T-1} |S_t| \geq 2 \sum_{t=2}^{T-1} |S_{t-1}| + |S_{T-1}|.$$

From this we easily derive:

$$f(S_1, \ldots, S_T) \geq \sum_{t=2}^{T-2} |S_t|. \tag{7}$$

By summing Equations (6) and (7) we have $2f(S_1, \ldots, S_T) \geq f(\hat{S}_1, \ldots, \hat{S}_T) - |S_{T-1}|$. The competitive ratio follows from the fact that $f(\hat{S}_1, \ldots, \hat{S}_T) \geq (T-1)|S_{T-1}|$ (since S_{T-1} is feasible for all time steps). ◀

We note that competitive ratio 2 can be derived with a similar analysis when the number of time steps is 2 or 3. In the full version, we show a matching asymptotical lower bound.

▶ **Theorem 9.** *Consider the SSFS model with intersection bonus. For any $\epsilon > 0$ and number of time steps $T = \lceil 1/\epsilon \rceil$, there is no $(2-\epsilon)$-competitive algorithm.*

3 Model of General Evolution

We consider in this section that the set of feasible solutions may evolve over time. We will show that in the Hamming bonus model, we can still get constant competitive ratios, though ratios slightly worse than in the case where only profits could change over time. Then, we will tackle the intersection bonus model, showing that no constant competitive ratio can be achieved. However, with only 1-lookahead we can get a constant competitive ratio.

3.1 Hamming-Bonus Model

In this section we consider the Hamming bonus model. We first show in Section 3.1.1 that there exists a $\left(3 + \frac{1}{T-1}\right)$-competitive algorithm. Interestingly, we then show in Section 3.1.2 that a slight assumption on the problem structure allows to improve the competitive ratio. More precisely, we achieve a 21/8 (asymptotic) competitive ratio if we assume that the objective function is subadditive (so including the additive case) and that a subset of a feasible solution is feasible. These assumptions are satisfied by all the problems mentioned in introduction. We finally consider lower bounds in Section 3.1.3.

3.1.1 General Case

We adapt the idea of the 2-competitive algorithm working for the Hamming bonus model for a static set of feasible solutions (Section 2.1) to the current setting where the set of feasible solutions may change. Let us consider the following algorithm `BestOrNothing`: at each time step t, `BestOrNothing` computes an optimal solution S_t^* with associated profit $p_t(S_t^*)$ and compares it to 2 times the maximum potential bonus, i.e to $2n$. It chooses S_t^* if the associated profit is at least $2n$, otherwise it chooses $S_t = \emptyset$. A slight modification is applied for the last step T.

In the full version, we define the algorithm formally and prove an upper bound on the competitive ratio achieved by this algorithm.

▶ **Theorem 10.** *In the GE model with Hamming bonus,* `BestOrNothing` *is* $\left(3 + \frac{1}{T-1}\right)$*-competitive.*

3.1.2 Improvement for Sub-additivity and Subset Feasibility

In this section we assume that the problem have the following two properties:

- *subset feasibility*: at any time step, every subset of a feasible solution is feasible.
- *sub-additivity*: for any disjoint S, S', any t, $p_t(S \cup S') \leq p_t(S) + p_t(S')$.

Note that this implies that, if a feasible set X is partitioned into (disjoint) subsets $X_1, \ldots, X_h$, then $X_1, \ldots, X_h$ are feasible and $p_t(X) \leq \sum_i p_t(X_i)$.

We exploit this property to devise algorithms where we partition the set of objects and solve the problems on subinstances. As a first idea, let us partition the set of objects into into 3 sets A, B, C of size (roughly) $n/3$; consider the algorithm which at every time step t computes the best solutions S_t^A, S_t^B, S_t^C on each subinstance on A, B and C, and chooses S_t as the one of maximum profit between these 3 solutions. By sub-additivity and subset feasibility, the algorithm gets profit at least $1/3$ of the optimal profit at each time step. Dealing with bonuses, at each time step the algorithm chooses a solution included either in A, or in B, or in C so, for any $t < T$, at least one set among A, B and C is not chosen neither at time t nor at time $t+1$, and the algorithm gets transition bonus at least $n/3$. Hence, the algorithm is 3-competitive.

We now improve the previous algorithm. The basic idea is to remark that if for two consecutive time steps $t, t+1$ the solution S_t and S_{t+1} are taken in the same subset, say A, then the bonus is (at least) $2n/3$ instead of $n/3$. Roughly speaking, we can hope for a ratio better than $1/3$ for the bonus. Then the algorithm makes a trade-off at every time step: if the profit is very high then it will take a solution maximizing the profit, otherwise it will do (nearly) the same as previously. More formally, let us consider the algorithm `3-Part`. We first assume that n is a multiple of 3. A parameter $x \in \mathbb{R}_+$ will be defined later.

```
1. Partition $N$ in three subsets $A, B, C$ of size $n/3$.
2. For $t \in \{1, \ldots, T\}$: compute a solution $S_t^*$ maximizing $p_t(S)$
   - Case (1): If $p_t(S_t^*) \geq xn$: define $S_t = S_t^*$
   - Otherwise ($p_t(S_t^*) \leq xn$): compute solutions with optimal profit $S_t^A$, $S_t^B$, $S_t^C$ included
     in $A$, $B$ and $C$. Let $a_t$, $b_t$ and $c_t$ the respective profits.
     - Case (2): if $t \geq 2$ and Case (1) did not occur at $t-1$, do:
       If $S_{t-1} \subseteq A$ (resp. $S_{t-1} \subseteq B$, $S_{t-1} \subseteq C$), compute $\max\{a_t + 2n/3, b_t + n/3, c_t + n/3\}$
       (resp. $\max\{a_t + n/3, b_t + 2n/3, c_t + n/3\}$, $\max\{a_t + n/3, b_t + n/3, c_t + 2n/3\}$) and
       define $S_t$ as $S_t^A$, $S_t^B$ or $S_t^C$ accordingly.
     - Case (3) ($t = 1$ or Case (1) occurred at $t-1$) do:
       * Define $S_t$ as the solution with maximum profit among $S_t^A$, $S_t^B$, $S_t^C$.
```

If N is not a multiple of 3, we add one or two dummy objects that are in no feasible solutions (at any step).

In the full version, we prove an upper bound on the competitive ratio of this algorithm.

▶ **Theorem 11.** *Consider the GE model with Hamming bonus. Under the assumption of subset feasibility and sub-additivity,* `3-Part` *is* $(21/8 + O(1/T + 1/n))$*-competitive.*

3.1.3 Lower Bounds

We complement the algorithmic results with a lower bound for two time steps and an asymptotical one, which we both prove in the full version. Interestingly, these bounds are also valid for the latter restricted setting with subset feasibility and sub-additivity.

▶ **Theorem 12.** *Consider the GE model with Hamming bonus and* $T = 2$ *time steps. For any* $\epsilon > 0$*, there is no* $(1 + \sqrt{2} - \epsilon)$*-competitive algorithm.*

▶ **Theorem 13.** *Consider the GE model with Hamming bonus. For every* $\epsilon > 0$*, there is a* T_ϵ *such that, for each number of time steps* $T \geq T_\epsilon$*, there is no* $(\alpha - \epsilon)$*-competitive algorithm where* $\alpha = \frac{6\cdot\sqrt[3]{9+\sqrt{87}}}{\sqrt[3]{6\cdot(9+\sqrt{87})^2} - \sqrt[3]{36}} \approx 1.696$*.*

3.2 Intersection-Bonus Model

We now look at the general-evolution model with intersection bonus. This model is different from the ones considered before: We first give a simple lower bound showing that there is no constant-competitive algorithm.

▶ **Theorem 14.** *In the GE model with intersection bonus, there is no c-competitive algorithm for any constant c.*

Proof. We consider an instance with no profit. Let $T = 2$, $N = \{1, 2\}$, and $\mathcal{F}_1 = \{\emptyset, \{1\}, \{2\}\}$, that is, there are two items, and at time 1 it is only forbidden to take both of them. Assume w.l.o.g. that the algorithm does not pick item 2 at time 1. Then picking item 1 becomes infeasible at time 2 while picking item 2 remains feasible. Then the algorithm achieves 0 profit and bonus while the optimum can achieve a bonus of 1. ◀

Note that in this model, by adding dummy time steps giving no bonus and no profit, the previous lower bound extends to any number of time steps. This lower bound motivates considering the 1-lookahead model: at time t, besides I_t, the algorithm knows the instance I_{t+1}. It shall decide the feasible solution chosen at time t. We consider an algorithm based on the following idea: at some time step t, the algorithm computes an optimal sequence of 2 solutions $(S^*_{t,1}, S^*_{t,2})$ of value z^*_t for the subproblem defined on time steps t and $t + 1$. Suppose it fixes $S_t = S^*_{t,1}$. Then, at time $t + 1$, it computes $(S^*_{t+1,1}, S^*_{t+1,2})$ of value z^*_{t+1}. Depending on the values z^*_t and z^*_{t+1}, it will either choose to set $S_{t+1} = S^*_{t,2}$, confirming its choice at t (getting in this case value z^*_t for sure between time t and $t + 1$), or change its mind and set $S_{t+1} = S^*_{t+1,1}$ (possibly no value got yet, but a value z^*_{t+1} if it confirms this choice at $t + 2$). When a choice is confirmed ($S_t = S^*_{t,1}$ and $S_{t+1} = S^*_{t,2}$), then the algorithm starts a new sequence (fix $S_{t+2} = S^*_{t+2,1}, \ldots$).

More formally, let $(S^*_{t,1}, S^*_{t,2})$ be an optimal solution of the subproblem defined on time steps t and $t + 1$, and denote z^*_t its value (including profits and bonus between time t and $t + 1$). To avoid unnecessary subcases, we consider at time T $(S^*_{T,1}, S^*_{T,2})$ where $S_{T,2} = \emptyset$ and z^*_T is the profit of the optimal solution for the single time step T, $S^*_{T,1}$. Then consider the algorithm `Balance` which:

1. At time $t = 1$ compute $(S^*_{1,1}, S^*_{1,2})$ and fix $S_1 = S^*_{1,1}$.
2. For $t = 2$ to T: compute $(S^*_{t,1}, S^*_{t,2})$.
   - Case (1): If at $t-1$ the algorithm chose S_{t-1} equal to $S^*_{t-2,2}$ (i.e., Case (3) occurred), then fix $S_t = S^*_{t,1}$.
   - Case (2): Otherwise, if $z^*_t > 2z^*_{t-1}$, then fix $S_t = S^*_{t,1}$.
   - Case (3): Otherwise fix $S_t = S^*_{t-1,2}$.

▶ **Theorem 15.** *In the GE model with intersection bonus and 1-lookahead,* `Balance` *is a 4-competitive algorithm.*

Proof. Let V be the set of time steps in which Case (3) occurred. In the proof, intuitively we partition the time period into periods which end at some time $t \in V$, and prove the claimed ratio in each of these sub-periods.

Formally, let u, v $(u < v)$ be two time steps in V such that $w \notin V$ for any $u < w < v$. Note that since Case (3) occurred at time u, Case (1) occurred at $u+1$, so $u \neq v-1$, and Case (2) occurred at time $u+2, \ldots, v-1$. So $z^*_t > 2z^*_{t-1}$ for $t = u+2, \ldots, v-1$. By an easy recurrence, this means that, for all $t \in \{u+1, \ldots, v-1\}$, we have $z^*_t < z^*_{v-1}/2^{v-1-t}$. By taking the sum, we get $\sum_{t=u+1}^{v-1} z^*_t < 2z^*_{v-1}$. Since Case (3) occurred at v, $z^*_v \leq 2z^*_{v-1}$. Finally:

$$\sum_{t=u+1}^{v} z^*_t \leq 4z^*_{v-1}.$$

Now, at each time v for which case (3) occurred, we choose $S_v = S^*_{v-1,2}$. As previously said, Case (3) did not occur at $v-1$, so we choose $S_{v-1} = S^*_{v-1,1}$. Then the algorithm gets value at least z^*_{v-1} for these two time steps. In other words $f(S_1, \ldots, S_T) \geq \sum_{v \in V} z^*_{v-1}$. Consider first the case where $T \in V$ (case (3) occurred at time T). Then we get a partition of the time steps into subintervals ending in $v \in V$. So

$$\sum_{t=1}^{T} z^*_t \leq 4 \sum_{v \in V} z^*_{v-1} \leq 4f(S_1, \ldots, S_T).$$

Let $(\hat{S}_1, \ldots, \hat{S}_T)$ be an optimal solution. We have $p_t(\hat{S}_t) + p_{t+1}(\hat{S}_{t+1}) + b(\hat{S}_t, \hat{S}_{t+1}) \leq z^*_t$. So $f(\hat{S}_1, \ldots, \hat{S}_T) \leq \sum_{t=1}^{T-1} z^*_t$, and:

$$f(\hat{S}_1, \ldots, \hat{S}_T) \leq \sum_{t=1}^{T} z^*_t \leq 4f(S_1, \ldots, S_T).$$

Note that this is overestimated, each $p_t(\hat{S}_t)$ appears two times in the sum.

Now, if $T \notin V$, then $T-1 \in V$: indeed, Case (2) cannot occur at time T (since $z^*_t \leq z^*_{t-1}$). So we have in this case:

$$\sum_{t=1}^{T-1} z^*_t \leq 4 \sum_{v \in V} z^*_{v-1} \leq 4f(S_1, \ldots, S_T).$$

But again since $f(\hat{S}_1, \ldots, \hat{S}_T) \leq \sum_{t=1}^{T-1} z^*_t$, we have

$$f(\hat{S}_1, \ldots, \hat{S}_T) \leq \sum_{t=1}^{T-1} z^*_t \leq 4f(S_1, \ldots, S_T).$$

This completes the proof. ◀

In the full version, we prove a matching lower bound. The idea of the proof of the matching lower bound is as follows: As can be seen from the proof of Theorem 15, the estimate on the profit has slack for the 4-competitive algorithm. We give a construction in which there is no profit and in which the bonus when not “committing” to the solution from the previous time step is geometrically increasing over time; otherwise the bonus is 0. As it turns out, however, when the factor is 2 in each time step, we cannot show a lower bound of 4 in case the algorithm does not commit until the last time step. Interestingly, if we use the minimum factor to show a lower bound of $4-\epsilon$ in case the algorithm commits at any time step but the last, we can find a large-enough time horizon such that, in case the algorithm commit only in the last time step, we can also show a lower bound of $4-\epsilon$.

▶ **Theorem 16.** *Consider the GE model with intersection bonus. For any $\epsilon > 0$, there is a T_ϵ such that, for each number of time steps $T \geq T_\epsilon$, there is no $4-\epsilon$ competitive ratio.*

4 Conclusion

In this paper, we have developed techniques for online multistage subset maximization problems and thereby settled the achievable competitive ratios in the various settings almost exactly. Disregarding asymptotically vanishing terms in the upper bounds, what remains open is the exact ratio in the general-evolution setting with Hamming bonus (shown to be between $1+\sqrt{2}$ and 3 in this paper) and exact bounds for the models with Hamming bonus when $T \to \infty$. Furthermore, it is plausible that the ratios can be improved for (classes of) more specific problems.

We emphasize that we have focussed on deterministic algorithms in this work. Indeed, some of our bounds can be improved by randomization (assuming an oblivious adversary):

- In the general-evolution model with Hamming bonus assuming sub-additivity and subset feasibility, there is a simple randomized $(2+o(1))$-competitive algorithm (along the lines of the algorithms in Subsubsection 3.1.2): Initially partition N uniformly at random into two equal-sized sets (up to possibly one item) A and B. At each time, select the optimal solution restricted to A. Again, the algorithm is $(2+o(1))$-competitive separately on both profit and bonus.
- While the strong lower bound without lookahead in the general-evolution model with intersection bonus still holds, we can get a simple 2-competitive algorithm for lookahead 1: Initially flip a coin to interpret the instance as a sequence of length-2 instances either starting at time 1 or 2. Thanks to lookahead 1, the length-2 instances can all be solved optimally. The total value of all these length-2 instances adds up to at least the optimal value, and the expected value obtained by the algorithm is half of that.

While we believe that we have treated various of the most natural ways of defining value in multistage subset maximization problems, other ways can be thought of, to some of which our results extend. For instance, Theorem 5 also works for time-dependent or object-dependent bonus without major modifications (whereas, e.g., Theorem 8 does not).

We have not worried about computational complexity in this work (and therefore neither about the representation of the set of feasible solutions); indeed, often we use an oracle providing the optimal solution to instances of a potentially hard problem. However, we mention that, if only an approximation algorithm to the problem at hand was known, we would be able to obtain similar online algorithms whose competitive ratio would depend on the approximation guarantee of the approximation algorithm.

References

1 Susanne Albers and Jens Quedenfeld. Optimal Algorithms for Right-Sizing Data Centers. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 363–372, 2018.

2 Hyung-Chan An, Ashkan Norouzi-Fard, and Ola Svensson. Dynamic Facility Location via Exponential Clocks. *ACM Trans. Algorithms*, 13(2):21:1–21:20, 2017.

3 Barbara M. Anthony and Anupam Gupta. Infrastructure Leasing Problems. In *Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 424–438, 2007.

4 Antonios Antoniadis and Kevin Schewior. A Tight Lower Bound for Online Convex Optimization with Switching Costs. In *Workshop on Approximation and Online Algorithms (WAOA)*, pages 164–175, 2017.

5 Evripidis Bampis, Bruno Escoffier, Michael Lampis, and Vangelis Th. Paschos. Multistage Matchings. In *Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 7:1–7:13, 2018.

6 Evripidis Bampis, Bruno Escoffier, and Sasa Mladenovic. Fair Resource Allocation Over Time. In *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 766–773, 2018.

7 Evripidis Bampis, Bruno Escoffier, Kevin Schewior, and Alexandre Teiller. Online Multistage Subset Maximization Problems. *CoRR*, abs/1905.04162, 2019. `arXiv:1905.04162`.

8 Nikhil Bansal, Anupam Gupta, Ravishankar Krishnaswamy, Kirk Pruhs, Kevin Schewior, and Clifford Stein. A 2-Competitive Algorithm For Online Convex Optimization With Switching Costs. In *Workshop on Approximation, Randomization, and Combinatorial Optimization Algorithms and Techniques (APPROX/RANDOM)*, pages 96–109, 2015.

9 Nicolas K. Blanchard and Nicolas Schabanel. Dynamic Sum-Radii Clustering. In *International Conference and Workshops on Algorithms and Computation (WALCOM)*, pages 30–41, 2017.

10 Niv Buchbinder, Shahar Chen, and Joseph Naor. Competitive Analysis via Regularization. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 436–444, 2014.

11 Niv Buchbinder, Shahar Chen, Joseph Naor, and Ohad Shamir. Unified Algorithms for Online Learning and Competitive Analysis. *Math. Oper. Res.*, 41(2):612–625, 2016. `doi:10.1287/moor.2015.0742`.

12 Edith Cohen, Graham Cormode, Nick G. Duffield, and Carsten Lund. On the Tradeoff between Stability and Fit. *ACM Trans. Algorithms*, 13(1):7:1–7:24, 2016. `doi:10.1145/2963103`.

13 David Eisenstat, Claire Mathieu, and Nicolas Schabanel. Facility Location in Evolving Metrics. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 459–470, 2014.

14 Albert Gu, Anupam Gupta, and Amit Kumar. The Power of Deferral: Maintaining a Constant-Competitive Steiner Tree Online. *SIAM J. Comput.*, 45(1):1–28, 2016. `doi:10.1137/140955276`.

15 Anupam Gupta, Kunal Talwar, and Udi Wieder. Changing Bases: Multistage Optimization for Matroids and Matchings. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 563–575, 2014.

16 Vinay Joseph and Gustavo de Veciana. Jointly optimizing multi-user rate adaptation for video transport over wireless systems: Mean-fairness-variability tradeoffs. In *IEEE International Conference on Computer Communications (INFOCOM)*, pages 567–575, 2012.

17 Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. Dynamic Right-Sizing for Power-Proportional Data Centers. *IEEE/ACM Trans. Netw.*, 21(5):1378–1391, 2013.

18 Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. Dynamic Right-Sizing for Power-Proportional Data Centers. *IEEE/ACM Trans. Netw.*, 21(5):1378–1391, 2013.

19 Zhenhua Liu, Iris Liu, Steven H. Low, and Adam Wierman. Pricing data center demand response. In *ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 111–123, 2014.

20 Nicole Megow, Martin Skutella, José Verschae, and Andreas Wiese. The Power of Recourse for Online MST and TSP. *SIAM J. Comput.*, 45(3):859–880, 2016. doi:10.1137/130917703.

21 Chandrashekhar Nagarajan and David P Williamson. Offline and online facility leasing. *Discrete Optimization*, 10(4):361–370, 2013.

22 Neil Olver, Kirk Pruhs, Rene Sitters, Kevin Schewior, and Leen Stougie. The Itinerant List-Update Problem. In *Workshop on Approximation and Online Algorithms (WAOA)*, pages 310–326, 2018.

23 Kirk Pruhs and Gerhard J. Woeginger. Approximation schemes for a class of subset selection problems. *Theor. Comput. Sci.*, 382(2):151–156, 2007.

24 Cécile Rottner. *Combinatorial Aspects of the Unit Commitment Problem.* PhD thesis, Sorbonne Université, 2018.

25 Daniel Dominic Sleator and Robert Endre Tarjan. Amortized Efficiency of List Update and Paging Rules. *Commun. ACM*, 28(2):202–208, 1985.