Dynamic Combinatorial Optimization: a complexity and approximability study

David Adjiashvili
Institute for Operations Research (IFOR)
Eidgenössischen Technischen Hochschule (ETH) Zürich
Rämistrasse 101, 8092 Zürich, Switzerland
Email: david.adjiashvili@ifor.math.ethz.ch

Sandro Bosio
Institute for Operations Research (IFOR)
Eidgenössischen Technischen Hochschule (ETH) Zürich
Rämistrasse 101, 8092 Zürich, Switzerland
Email: sandro.bosio@ifor.math.ethz.ch

Robert Weismantel
Institute for Operations Research (IFOR)
Eidgenössischen Technischen Hochschule (ETH) Zürich
Rämistrasse 101, 8092 Zürich, Switzerland
Email: robert.weismantel@ifor.math.ethz.ch

Abstract

Dynamic optimization problems are widely studied in many areas of mathematics and control theory. However, relatively little is known about dynamic counterparts of purely combinatorial problems, except for specific structures such as network flows and scheduling problems. This paper introduces a general setting for studying and analyzing over-time versions of combinatorial optimization problems. We provide an extensive complexity study, including a widely applicable oracle-based approximation algorithm.

In a combinatorial problem $\mathcal{P}$ we are given a set system $(A, \mathcal{X})$ and a linear weight function w, and the goal is to find a solution $S \in \mathcal{X}$ with maximum weight $w(S)$. Given a time window $W = \{1, \ldots, T\}$ and durations $\tau_a \in W$ for each $a \in A$, in the temporal extension of $\mathcal{P}$ we search for a *solution over time* with maximum *weight over time*. A solution over time is a collection of static solutions $S_t \in \mathcal{X}$, one for each $t \in W$, such that for each $a \in A$ the set of time points $\{t \in W : a \in S_t\}$ at which a is selected is a disjoint union of intervals of length τ_a. Informally, each ground-set element can be selected any number of times within the time window W, but whenever it gets *activated* it must remain selected for its whole duration. The weight over time is simply defined as $\sum_{t \in W} w(S_t)$.

Our main result is an $\alpha\beta$-approximation algorithm, with $\alpha \approx 1.691030$, for the rather general case in which $(A, \mathcal{X})$ is an independence system and $\mathcal{P}$ admits a β-approximation algorithm. This algorithm considerably generalizes the α-approximation total-value heuristic by Kohli and Krishnamurti (1992) for integer knapsack. We discuss the connection to the integer knapsack and mention applications to scheduling and orthogonal rectangle packing.

1 Introduction

The need to model the temporal dimension in real-world applications has sprung the development of several new branches in combinatorial optimization. While such extensions have been studied in depth in some fields, scheduling and network flow theory being notable examples, other areas of combinatorial optimization did not receive a similar attention. This paper aims at reducing the gap, presenting a new model for extending combinatorial optimization problems over time that has natural applications in scheduling, packing, and knapsack problems.

Let $(A, \mathcal{X})$ be a *set system* defined by a finite *ground set* A and a family $\mathcal{X}$ of *feasible subsets*[1] of the ground set. Given weights $w_a \in \mathbb{Q}$ for each $a \in A$, the standard linear maximization problem associated with $(A, \mathcal{X})$ is to find an element $S \in \mathcal{X}$ with maximum weight $w(S) = \sum_{a \in S} w_a$. Let $[n]$ denote the set $\{1, \dots, n\}$, for $n \in \mathbb{N}_+$. Given a *total duration* $T \in \mathbb{N}_+$ and a *duration* $\tau_a \in [T]$ for each $a \in A$, we define the corresponding *linear maximization problem over time* as the problem of finding a collection $\mathcal{S} = (S_1, \dots, S_T)$ such that

1) $S_t \in \mathcal{X}$ for every $t \in [T]$, and

2) for each $a \in A$ the set $\{t : a \in S_t\}$ is the disjoint union of intervals of length τ_a, i.e., the disjoint union of sets $\{\delta_i^a, \dots, \delta_i^a + \tau_a - 1\}$, where δ_i^a are called *activation times*,

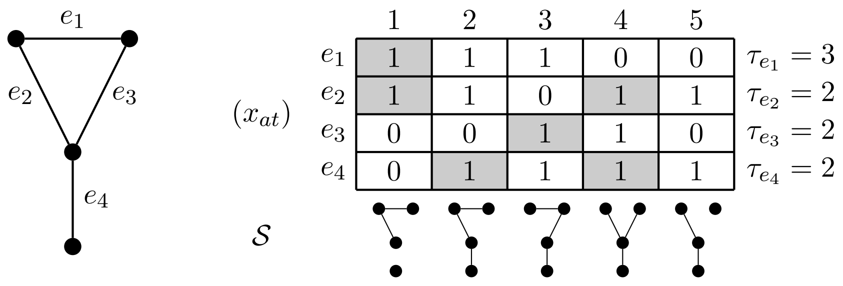
so as to maximize the *weight over time* $w(\mathcal{S}) = \sum_{t \in [T]} w(S_t)$ of the solution. For brevity we will refer to the time extension of a problem $\mathcal{P}$ by "$\mathcal{P}$ over time". We will use uppercase letters to denote solutions of $\mathcal{P}$ (sometime denoted static solutions), and use calligraphic uppercase letters to refer to "solutions over time", that is, solutions satisfying conditions 1) and 2). To avoid any misunderstanding, we warn the reader that the network flow over time problems that can be derived from this definition are *not* the standard network flow over times problems studied in the literature, which use a temporal extension that is unique to flows and not directly applicable to other problems. We will remark more on flows over time later in the paper.

A solution over time can be seen as an $|A| \times T$ binary matrix (x_{at}), where each column $x_{\cdot t}$ is the incidence vector of some $S \in \mathcal{X}$, and in each row $x_{a \cdot}$ any maximal consecutive-one sequence has length multiple of τ_a (see Figure 1). An informal phrasing of condition 2) is that whenever an element is *activated* it remains in the solution for exactly τ_a time units, after which it becomes inactive and can be reactivated again at any later time. A more formal equivalent definition is that each row $x_{a \cdot}$ corresponds to a solution of an integer knapsack with a single item of size τ_a and knapsack capacity T, where $x_{at} = 1$ if the knapsack position t is occupied and 0 otherwise.

As T is an integer parameter in the input, an explicit description of a solution over time would in general be exponential in the input size. To avoid confusion with output-polynomial algorithms, we stress here that all algorithms described in this paper run in *input*-polynomial time, as the solutions over time they provide can be encoded efficiently. The study of output-polynomial algorithms is an interesting direction for future work.

Deciding if a solution over time exists for a generic set system is NP-hard. In this paper we focus our attention mostly on independence systems, for which existence of a solution is always guaranteed. Let us recall that $(A, \mathcal{X})$ is an *independence system* (IS) if $X \in \mathcal{X}$ and $Y \subseteq X$ imply $Y \in \mathcal{X}$. Since optimizing over an IS is NP-hard in general (the set of all cliques of a graph is an IS), the over-time case will be just as hard, as for $T = 1$ it reduces to the static case. The following stronger negative result can be stated.

[1]For the sake of simplicity we consider simple subsets, but all results can be directly extended to consider multisets. Note also that multisets can be directly modeled into the current notation by suitably expanding A with additional elements, if multiplicity upper bounds are known.



$$(x_{at})\quad \begin{array}{c|ccccc|l} & 1 & 2 & 3 & 4 & 5 & \\ \hline e_1 & 1 & 1 & 1 & 0 & 0 & \tau_{e_1} = 3 \\ e_2 & 1 & 1 & 0 & 1 & 1 & \tau_{e_2} = 2 \\ e_3 & 0 & 0 & 1 & 1 & 0 & \tau_{e_3} = 2 \\ e_4 & 0 & 1 & 1 & 1 & 1 & \tau_{e_4} = 2 \end{array}$$

Figure 1: A graph (left) and a corresponding forest over time (right) for $T = 5$. Edge durations are shown in the figure. The matrix elements corresponding to activations are shaded in gray. Assuming uniform edge weight 1, this forest over time has total weight 13, which is also maximum.

Theorem 1. *The linear maximization problem over time is* APX*-hard, even if $T = 3$, $(A, \mathcal{X})$ is an IS, and the linear maximization problem on $(A, \mathcal{X})$ is in* P*.*

The assumption $T = 3$ cannot be improved, as even for a generic set system for $T < 3$ the problem over time is not harder than the original problem.

Theorem 2. *Given a set system $(A, \mathcal{X})$ for which the linear maximization problem admits a polynomial time β-approximation, if $T < 3$ the corresponding linear maximization problem over time admits a polynomial time β-approximation.*

Moreover, if the linear maximization problem admits a constant factor approximation, then APX-hardness cannot be strengthened either. This is a consequence of the main result of this paper, which we state hereafter.

Theorem 3. *Given an IS $(A, \mathcal{X})$ admitting a polynomial time β-approximation algorithm for linear maximization, with $\beta = \beta(A, \mathcal{X})$, the corresponding linear maximization problem over time admits a polynomial time $\alpha\beta$-approximation algorithm, with $\alpha \approx 1.691030$.*

The reason for stating Theorem 3 with respect to a β-approximation algorithm is twofold. On the one hand it allows to state a result also for an IS for which linear maximization is NP-hard, but that admits an approximation algorithm. On the other hand, we can consider multi-dimensional extensions without extra effort. Indeed, consider the set of feasible solutions over time corresponding to some given IS. It is easy to see that this feasible set is again an IS, as removing an occurrence of an element $a \in A$ from a solution over time (which means removing a from some activation time t up to $t + \tau_a - 1$) still gives a feasible solution over time. We can hence treat the time extended version of the problem as an IS with an approximate linear optimization oracle and extend it using the same algorithm along another dimension.

The exact value of α in Theorem 3 is $\alpha = \sum_{i=1}^{\infty} 1/s_i$, where $\{s_i\}_{i \in \mathbb{N}_+}$ is the sequence defined recursively by $s_1 = 1$ and $s_{n+1} = s_n(s_n + 1)$ for $n \geqslant 2$. The sequence $\{s_i + 1\}_{i \in \mathbb{N}_+}$ is known as Sylvester's sequence (see [23] for references regarding both sequences). The constant α appears as the approximation guarantee of several well-known algorithms, including the total-value heuristic for integer knapsack [15] and the Harmonic algorithm for Bin Packing [19]. As will be discussed in more detail later, the link with integer knapsack is more than just casual. The algorithm on which Theorem 3 is based is a generalization of the total-value heuristic. Moreover, choosing as IS the uniform matroid of rank one, the resulting linear maximization problem over time is exactly the integer knapsack, so that the result in [15] can be seen a corollary of Theorem 3.

The integer knapsack can be seen as the 1-dimensional version of the d-dimensional orthogonal integer knapsack, which consists in filling a d-dimensional container with d-dimensional boxes without rotation (note that this problem is different from the so called multi-dimensional integer knapsack, where one has to satisfy several simultaneous knapsack criteria). This problem is also known as the unbounded orthogonal rectangle packing problem. The binary case, in which each box can be used at most once, has been studied in several contexts (see e.g. [7, 8] and the references therein). The 2-dimensional binary case was studied by Jansen et al. [13], who proposed a $(2+\epsilon)$-approximation algorithm. Recursively applying Theorem 3 (extended to ISs defined on multisets) we directly obtain an α^d-approximation algorithm for d-dimensional orthogonal integer knapsack. Moreover, as the integer knapsack admits an FPTAS (see e.g. [12, 18]), we can state the following stronger corollary of Theorem 3, which to the best of our knowledge is the first nontrivial approximation algorithm for this problem:

Corollary 1. *The d-dimensional orthogonal integer knapsack admits an $\alpha^{d-1}(1+\epsilon)$-approximation algorithm for any $\epsilon > 0$.*

We mention here a second application of Theorem 3. Consider a factory with a set M of machines, where a set A of goods can be produced. Each good $a \in A$ has a profit w_a, and the production of a unit of good a requires the cooperation (or joint operation) of a certain subset $M_a \subseteq M$ of the machines, and occupies these machines for a certain production time τ_a (without preemption). Goods can be produced in any quantity, and each machine can work on at most one good at any given time. Assume that the factory is rented for a certain period of T time units. It is required to find a production schedule that maximizes the total profit within the time window $[T]$. We call this problem Cooperative Scheduling (CS).

It is easy to see that, by identifying M with a vertex set and each M_a with an hyperedge, CS can be easily seen to be the maximum weight hypergraph matching problem over time, as we want to find a collection of hypergraph matchings (subsets of goods that require no common machine), for each time $t \in [T]$, that satisfy conditions 2). Using the $(\frac{k+1}{2}+\epsilon)$-approximation algorithm for k-uniform weighted hypergraph matching [4], and using a polynomial time algorithm for the classical weighted matching problem, we can state the following two corollaries of Theorem 3.

Corollary 2. *When $|M_a| \leqslant k$ for all goods $a \in A$, CS admits an $(\alpha\frac{k+1}{2}+\epsilon)$-approximation algorithm.*

Corollary 3. *When $|M_a| \leqslant 2$ for all goods $a \in A$, CS becomes the weighted matching problem over time, and hence admits an α-approximation algorithm.*

We remark here that the weighted matching problem over time is APX-hard. This result, which implies Theorem 1, is stated hereafter and proved in the appendix.

Theorem 4. *Finding a maximum weighted matching over time is* APX*-complete, even if $T = 3$, G is a bipartite graph, $\Delta(G) = 4$, and $w_a = 1$ for all $a \in A$.*

Algorithms for many generalizations of CS can be directly derived as well. For example, Corollary 3 can be extended to the case in which each machine can process at most $b_i \in \mathbb{N}_+$ goods at the same time, which is the weighted b-matching problem over time and hence admits an α-approximation algorithm as well.

The rest of the paper is organized as follows. Section 2 reviews related work. In Section 3, after stating some preliminary results for time expansions of both general set systems and ISs, we describe the algorithm on which Theorem 3 is based and prove the $\alpha\beta$ approximation guarantee, leaving the proofs of some technical claims in the appendix. Finally, Section 4 summarizes the contribution and discusses open questions and further research directions.

2 Related Work

We start reviewing some related work on the integer knapsack. The integer knapsack is weakly NP-hard, and admits a pseudo-polynomial time algorithm as well as an FPTAS [12, 18]. More relevant for this paper, however, are the approximation guarantees obtained by greedy approaches. The classic density-based greedy algorithm is a 2-approximation algorithm, and the total-value heuristic improves this to an α-approximation algorithm [15]. These heuristics will be reviewed in more detail in the next section, where they will be generalized to IS linear maximization problems over time. For both these algorithms the approximation factor is guaranteed by a single greedy iteration, which selects one specific item and fills as much left capacity as possible. Iterating the process with respect to the remaining capacity can improve the solution, but does not improve the worst case analysis. Iterative versions are however required for the joint analysis of these two algorithms, for which an approximation guarantee of 1.5 was proved in [16]. The average-case behavior of these algorithms was studied in [17].

Flows over time are a well-known temporal extension of network flows. The field originated from a seminal paper by Ford and Fulkerson [9] which extended the standard maximum flow problem, showing that computing a single minimum cost static flow and constructing a corresponding "temporally repeated" flow gives an optimal flow over time (referred to as *dynamic flow* in the paper). In network flows the classic temporal extension makes the rather natural assumption that flow needs a given time τ_e to move from the source to the sink of an edge (think about vehicles moving along a transportation network for the discrete case, or liquids/gases moving into pipes for the continuous case). As such, discrete-time dynamic flow problems can be stated as standard flow problems on a temporally expanded network, and hence typically admit a pseudo-polynomial time algorithm. Following our definition, the temporal extension of network flows would have instantaneous flow transmission and additional temporal conditions on flow activation. This makes the problem closer to a path scheduling problem than to a flow problem. As such, the definition "flows over time" should be reserved for the classical temporal extension of flows. For an extensive review of flows over time we refer to [22]. We refer to [14] for a recent development combining discrete and continuous flows over time in a unique framework.

In scheduling theory the dimension of time is inherent in most problems. We refer the reader to the book of Pinedo [21] for a thorough treatment of this topic. The scheduling problems which are most relevant to this work are profit maximization scheduling problems [3, 2, 5], in which not all given jobs need to be processed. Instead, each job is associated with a certain profit that is gained in case it is completed before its deadline. Jobs can also have different release times and time-dependent profits. Further variants of the problem are achieved by specifying the number of available machines, whether the machines are identical or unrelated, etc. To the best of our knowledge the problem CS was not studied previously. CS can be viewed as a throughput maximization problem with identical release times and deadlines for all jobs, with time-independent profits and processing times. On the other hand the resource constraint is significantly more general than the one considered in most scheduling problems: while most scheduling problems specify a knapsack-type constraint on the total work performed in each time step (typically given by a number of available machines), CS requires that the set of jobs active at any given time is an element of an IS. The latter type of constraint is relevant in applications in which the set of resources is highly non-homogeneous, and every resource has a very specific utilization. In such applications resources are not interchangeable, hence a given job requires a predetermined subset of the resources to be processed.

A time extension of the matroid intersection problem was considered in [10]. Given two matroids $M_1 = (A, \mathcal{I}_1)$ and $M_2 = (A, \mathcal{I}_2)$, installation times $\tau : A \to \mathbb{N}_+$ and a total duration

$T \in \mathbb{N}_+$ the goal is to find T elements $X_1, \ldots, X_T \in \mathcal{I}_1$ satisfying the following additional conditions. Let $Y_t \subseteq A$ be defined as the set of elements $a \in A$ such that $a \in X_{t-\tau_a}$. Then $Y_1, \ldots, Y_T \in \mathcal{I}_2$. Furthermore, if $a \in X_t$ it must hold that $t + \tau_a \leqslant T$. A polynomial algorithm is obtained via a reduction to an ordinary matroid intersection problem on certain time-expanded matroids, and some applications are mentioned. This work was generalized in [11].

3 Results

Consider a set system $(A, \mathcal{X})$ corresponding to some combinatorial problem $\mathcal{P}$, and let weights w, durations τ, and a total duration T be given. While deciding if a solution over time exists for a general set system is NP-hard, even if $\mathcal{P}$ is polynomial time solvable, we can nevertheless state some simple positive conditional results. For $T = 1$ the problem $\mathcal{P}$ over time is simply the original problem $\mathcal{P}$. The following proposition asserts that, in general, if all durations divide T, then the complexity of $\mathcal{P}$ over time is the same as the complexity of $\mathcal{P}$ for any problem $\mathcal{P}$.

Proposition 1. *If all durations τ_a divide T, then a maximum weight solution over time can be derived by computing a single static maximum weight solution.*

Proof. Let $S^* \in \mathcal{X}$ be a maximum weight solution. For any solution over time $\mathcal{S}$ we have the trivial upper bound $w(\mathcal{S}) \leqslant Tw(S^*)$, as $w(S^t) \leqslant w(S^*)$ for all $t \in [T]$. As τ_a divides T for very $a \in A$, it is easy to see that $\mathcal{S}^* = \{S^t = S^*, t \in [T]\}$ is a feasible solution over time of value $w(\mathcal{S}^*) = Tw(S^*)$, hence it is also optimal. □

A direct corollary of this is the following.

Corollary 4. *Let S^* be an optimal solution for $\mathcal{P}$, and G^* be an optimal solution for $\mathcal{P}$ when only elements $a \in A$ whose duration divide T are allowed (if such a solution exists). Then $\mathcal{S}^* = \{S^t = G^*, t \in [T]\}$ is a solution over time providing an approximation guarantee of $w(S^*)/w(G^*)$.*

Moreover, since for $T = 2$ all durations are either 1 or 2, we obtain as a corollary Theorem 2, which we state again hereafter for convenience.

Theorem 2. *Given a set system $(A, \mathcal{X})$ for which the linear maximization problem admits a polynomial time β-approximation, if $T < 3$ the corresponding linear maximization problem over time admits a polynomial time β-approximation.*

In contrast, for $T = 3$ deciding if a solution over time exists is NP-complete. This is stated hereafter, and proved in the appendix.

Corollary 5. *For a generic set system $(A, \mathcal{X})$, deciding if a solution over time exists is* NP*-hard, even if linear optimization on $(A, \mathcal{X})$ is in* P *and $T = 3$.*

Note that when T is polynomial in the input size, or more generally if $\frac{T}{\tau_a}$ is polynomial in the input size for every $a \in A$, any solution over time admits a compact representation.

3.1 Approximation algorithms for IS linear optimization over time

If $(A, \mathcal{X})$ is an independence system, existence of a feasible solution over time can always be guaranteed. On the other hand, the problem over time can be harder than its static version, as stated in Theorem 1. In this section we prove that the hardness jump cannot be too large. That is, if the static problem admits a constant factor approximation, then the over time version also admit a constant factor approximation (albeit not the same one).

As the relation to the integer knapsack is fundamental, let us recall it here. We will use slightly different notation from the classical one, so as to underline the connection with our over-time setting. Given a knapsack capacity T and a set A of items, where for each item $a \in A$ a volume $\tau_a \in [T]$ and a value p_a are given, the integer knapsack consists of finding nonnegative integers x_a for each $a \in A$ that satisfy the knapsack condition $\sum_{a \in A} \tau_a x_a \leqslant T$ and that maximize the total gain $\sum_{a \in A} p_a x_a$.

Proposition 2. *Integer knapsack is a linear optimization problem over time.*

Proof. The transformation can be done by defining the set system $(A, \mathcal{X})$ to be the uniform matroid of rank one $X = \{S \subseteq A : |S| \leqslant 1\}$, and defining item weights to be the unitary weight (or item density, in the integer knapsack terminology) $w_a = p_a/\tau_a$. □

Let $T_a = \lfloor T/\tau_a \rfloor$ be the number of times an item $a \in A$ fits into the knapsack capacity, and note that $T_a\tau_a > T/2$ for any $a \in A$. A classic heuristic for the integer knapsack is the density-based greedy algorithm, which provides the approximation guarantee 2. This greedy algorithm chooses the item $g \in A$ with highest density w_g and uses g as many times as possible, that is, sets $x_g = T_g$. As this solution has value $T_g p_g = T_g \tau_g w_g \geqslant \frac{T}{2} w_g$, and $T w_g$ is an upper bound on the optimum, the guarantee is easily verified. Note also that iterating the procedure so as to fill any remaining capacity does not improve the worst case guarantee. The following algorithm is a simple extension of the density-based heuristic to IS linear optimization over time.

Algorithm 1 Repeat Best$(A, \mathcal{X}, [T], \tau, w)$

Requires: A β-approximation for linear optimization on the IS $(A, \mathcal{X})$
Output: A solution over time $\mathcal{G}$

```
1: find $G \in \mathcal{X}$ such that $\beta w(G) \geqslant w(S)$ for all $S \in \mathcal{X}$
2: set $G_t \leftarrow \{a \in G : t \leqslant T_a\tau_a\}$ for all $t \in [T]$
3: return $\mathcal{G} = \{G_t : t \in [T]\}$
```

When $(A, \mathcal{X})$ is an IS, the solution $\mathcal{G}$ returned by Repeat Best is a feasible solution over time. It is also easy to see that $\mathcal{G}$ provides an approximation guarantee of 2β. Indeed, let $\mathcal{S}^*$ be an optimal solution over time. As $T\beta w(G)$ is an upper bound on $w(\mathcal{S}^*)$, we have

$$w(\mathcal{G}) = \sum_{a \in G} T_a \tau_a w_a \geqslant \sum_{a \in G} \frac{T}{2} w_a = \frac{T}{2} w(G) \geqslant \frac{w(\mathcal{S}^*)}{2\beta}.$$

Note also that Repeat Best makes a single call to the β-approximation oracle, and that the solution over time returned can be encoded efficiently due to its "repeated" nature.

A better worst-case greedy algorithm for the integer knapsack is the total-value heuristic [15]. Instead of repeating the item with highest density, this heuristic chooses the item that gives the highest total repeated value. Namely, it performs the same greedy choice as in the density heuristic, but using the total-value weights $w'_a = T_a p_a$ instead of the unitary weight w_a. As this choice dominates the density-based one, the total-value heuristic guarantee is at least a factor 2. The analysis in [15] proves that this greedy choice provides the approximation guarantee α, which is asymptotically tight and cannot be improved by iterating the greedy choice on the remaining capacity. The following algorithm is the natural extension of the total-value heuristic to IS linear optimization over time.

Algorithm 2 Best Repeat$(A, \mathcal{X}, [T], \tau, w)$

```
Requires: A $\beta$-approximation for linear optimization on the IS $(A, \mathcal{X})$
Output:   A solution over time $\mathcal{G}$
1: set $w'_a \leftarrow T_a \tau_a w_a$ for all $a \in A$
2: return Repeat Best$(A, \mathcal{X}, [T], \tau, w')$
```

The algorithm performs a single call to Repeat Best, and hence a single call to the β-approximation oracle. Note that the solution can again be encoded efficiently. As for the total-value heuristic, Repeat Best dominates Best Repeat, as it provides the best (up to the β factor) among all repeated solutions over time, and hence provides a guarantee of at least 2β. Moreover, as integer knapsack is a special case of IS linear optimization over time, it certainly cannot provide a guarantee better than $\alpha\beta$, as α is asymptotically tight for the integer knapsack. The main result of this paper is that Best Repeat actually guarantees the factor $\alpha\beta$.

Theorem 3. *Given an IS $(A, \mathcal{X})$ admitting a polynomial time β-approximation algorithm for linear maximization, with $\beta = \beta(A, \mathcal{X})$, the corresponding linear maximization problem over time admits a polynomial time $\alpha\beta$-approximation algorithm.*

The proof in [15] is based on the fact that g has highest total-value, that is, that $w'_g \geqslant w'_a$ for every $a \in A$, allows to rather directly bound the guarantee by an infinite dimensional *integer* program, which is then shown to have optimum α. In our case we have the condition $w'(G) \geqslant \beta w'(S)$ for every $S \in \mathcal{X}$, which is an aggregate bound only. The proving techniques we employ are substantially different, but will nonetheless allow us to draw a remarkable parallel with the proof in [15]. Indeed, we will eventually show that the guarantee can be bounded by an infinite dimensional *linear* program whose optimum is α. The rest of the paper is devoted to the proof of Theorem 3. The proofs of some technical claims are given in the appendix.

Proof. Let $G \in \mathcal{X}$ be the static solution chosen by Best Repeat, so that $\beta w'(G) \geqslant w'(S)$ for all $S \in \mathcal{X}$, and let $\text{Alg} = w(\mathcal{G}) = \sum_{a \in G} T_a \tau_a w_a$ be the value of the solution over time returned by Best Repeat. Let moreover $\mathcal{S}^* = (S_1, \ldots, S_T)$ be an optimal solution over time, with value $\text{Opt} = w(\mathcal{S}^*)$, and let $\delta = \frac{1}{\beta}\frac{\text{Opt}}{\text{Alg}}$. Our goal is to show that $\delta \leqslant \alpha$.

For each $k \in [T]$, we define the subset of elements that can be repeated k times by

$$A^k = \left\{ a \in A : \tau_a \in \left(\frac{T}{k+1}, \frac{T}{k} \right] \right\}. \tag{1}$$

As the durations τ_a are integers in $[T]$, the sets A^k partition A. By construction, for each $k \in [T]$ one has $T_a = k$ and $\tau_a \geqslant \Gamma_k$ for all $a \in A^k$, where $\Gamma_k = \left\lfloor \frac{T}{k+1} \right\rfloor + 1$. We then similarly partition each solution set S_t into the sets $S_t^k = S_t \cap A^k$, for $k, t \in [T]$.

For a fixed value $q \in [T]$, we define sets $M_1, \ldots, M_q \in \mathcal{X}$ by choosing $M_j = S_{t_j}$ for $t_j = \left\lceil j \frac{T}{q+1} \right\rceil$. For each set M_j we have that

$$\beta \text{Alg} = \beta \sum_{a \in G} T_a \tau_a w_a \geqslant \sum_{a \in M_j} T_a \tau_a w_a \geqslant \sum_{k=1}^{q} \sum_{a \in M_j \cap A^k} T_a \tau_a w_a = \sum_{k=1}^{q} \sum_{a \in M_j \cap A^k} k \tau_a w_a,$$

so that summing over all sets M_j for $j \in [q]$ we get

$$q\beta \text{Alg} \geqslant \sum_{j=1}^{q} \sum_{k=1}^{q} \sum_{a \in M_j \cap A^k} k \tau_a w_a = \sum_{k=1}^{q} k \sum_{j=1}^{q} \sum_{a \in M_j \cap A^k} \tau_a w_a. \tag{2}$$

In the last term, when we sum over $j \in [q]$ we are counting the elements $a \in S_t^k$ for all t with some repetitions. In the following counting argument we assume that $T > 2$, as for $T = 2$ and $T = 1$ the argument does not hold (these are the only cases in which $\lfloor \frac{T}{2} \rfloor + 1 \not< T$). Note that for $T \leqslant 2$ Corollary 4 guarantees that $\delta = 1$.

Consider a fixed $k \leqslant q$ and a $t \in [T]$. Since elements in S_t^k have duration at least Γ_k, and the (fractional) distance in time between two solutions M_j is $T/(q+1)$, each element $a \in S_t^k$ will belong to at least $\lfloor \Gamma_k (q+1)/T \rfloor$ different solutions M_j. Note that we stop at $k = q$ because, due to the choice of the solutions M_j, for $k > q$ some (or all) of the elements in S_t^k might never appear in the sets M_j. Let z_k be the fraction of OPT due to elements in A^k. Using the counting argument outlined above, for every $k \leqslant q$ we can write

$$\sum_{j=1}^{q} \sum_{a \in M_j \cap A^k} \tau_a w_a \geqslant \left\lfloor \Gamma_k \frac{q+1}{T} \right\rfloor \sum_{t=1}^{T} w(S_t^k) = \left\lfloor \Gamma_k \frac{q+1}{T} \right\rfloor z_k OPT. \tag{3}$$

Finally, combining (2) and (3) we get the bound

$$\sum_{k=1}^{q} \frac{1}{q} \left\lfloor \Gamma_k \frac{q+1}{T} \right\rfloor k z_k \leqslant \frac{1}{\delta}. \tag{4}$$

By collecting all bounds (4) for $q \in [T]$ together with the convexity constraint $\sum_{k=1}^{T} z_k = 1$, and applying the variable change $y_k = \delta k z_k$, we can bound δ by means of the following LP:

$$\begin{aligned} & \delta_T = \max && \sum_{k=1}^{T} \frac{1}{k} y_k && \\ (P_T) \qquad & s.t. && \sum_{k=1}^{q} \frac{1}{q} \left\lfloor \Gamma_k \frac{q+1}{T} \right\rfloor y_k \leqslant 1 && q \in [T] \\ & && y_k \geqslant 0 && k \in [T]. \end{aligned} \tag{5}$$

This derivation is shown in detail in the appendix in the proof of the following claim.

Claim 1. *$\delta \leqslant \delta_T$ for any $T > 2$.*

The value δ_T can be numerically computed, and provides an upper bound on the approximation guarantee for an instance with total duration T. Let us remark that given a specific instance (or instance class) the bound can be further specialized by setting $y_k = 0$ whenever A^k is empty. This can improve the guarantee, in particular if $A^k = \emptyset$ for small values of k. To obtain a global approximation guarantee we need to bound δ_T for every possible $T \in \mathbb{N}_+$. To do so, we use the following infinite-dimensional linear program:

$$\begin{aligned} & \delta_\infty = \max && \sum_{k \in \mathbb{N}_+} \frac{1}{k} y_k && \\ (P_\infty) \qquad & s.t. && \sum_{k=1}^{q} \frac{1}{q} \left\lfloor \frac{q+1}{k+1} \right\rfloor y_k \leqslant 1 && q \in \mathbb{N}_+ \qquad (6) \\ & && \sum_{k \in \mathbb{N}_+} \frac{1}{k+1} y_k \leqslant 1 && \qquad (7) \\ & && y_k \geqslant 0 && k \in \mathbb{N}_+. \end{aligned}$$

P_∞ can be understood intuitively as follows. Constraint (6) is obtained from constraint (5) by fixing q and taking T to infinity. Constraint (7) is obtained by taking the asymptotic version of constraint (5) for $q = T$, which can be equivalently stated in the simpler form

$$\sum_{k=1}^{T} \frac{\Gamma_k}{T} y_k \leqslant 1. \tag{8}$$

Regardless of how P_∞ is derived, considering also the fact that the sequence δ_T is not monotonic, one has to formally prove that it provides an upper bound to each problem P_T. This is stated in the following claim, and proved in the appendix.

Claim 2. *$\delta_T \leqslant \delta_\infty$ for every $T \in \mathbb{N}_+$.*

To show that $\delta_\infty \leqslant \alpha$, we make use of duality for infinite-dimensional linear programming. Let v_q be the dual variable to the q-th constraint (6), and let w be the dual variable corresponding to constraint (7). The standard linear programming dual of P_∞ reads:

$$\begin{array}{llll} & \delta'_\infty = \min & w + \sum_{q \in \mathbb{N}_+} v_q & \\ (D_\infty) & s.t. & \frac{1}{k+1} w + \sum_{q \geqslant k} \frac{1}{q} \left\lfloor \frac{q+1}{k+1} \right\rfloor v_q \geqslant \frac{1}{k} & k \in \mathbb{N}_+ \\ & & w \geqslant 0, v_q \geqslant 0 & q \in \mathbb{N}_+. \end{array} \tag{9}$$

In infinite-dimensional linear programming there are situations in which strong duality does not hold, and remarkably there are also situations in which weak duality fails as well. Moreover, while optimal primal and dual solutions exist in our case, in general one should speak about suprema (infima) instead than maxima (minima). For problems P_∞ and D_∞ strong duality can be claimed due to a classical result in [6]. This is stated hereafter, and proved in the appendix.

Claim 3. *$\delta_\infty = \delta'_\infty$.*

To prove that $\delta'_\infty \leqslant \alpha$, and hence to conclude the proof, we determine analytically a feasible dual solution of value α. To this end, recall the sequence $\{s_n\}_{n \in \mathbb{N}_+}$ used to define α. Our dual solution $(\tilde{w}, \tilde{v})$ is defined by setting $\tilde{w} = 1$ and

$$\tilde{v}_{s_j - 1} = \frac{1}{s_{j-1}} - \frac{1}{s_{j-1}^2} + \frac{1}{s_j^2}$$

for $j \geqslant 2$, and 0 elsewhere. That is, it assigns nonzero values only to variables v_q where $q = s_j - 1$ for some integer $j \geqslant 2$. The following claims, proved in the appendix, conclude the proof.

Claim 4. *The solution $(\tilde{w}, \tilde{v})$ has objective function value α.*

Claim 5. *The solution $(\tilde{w}, \tilde{v})$ is feasible for D_∞.*

□

We can also easily show that indeed $\delta_\infty = \alpha$. This can be done in two different ways. The first, informally speaking, is that α is asymptotically tight for the integer knapsack, and hence Repeat Best cannot provide a better guarantee. The second approach is to compare P_∞

with the infinite-dimensional integer programming problem resulting from the analysis of the total-value heuristic in [15], which reads as follows:

$$
\begin{array}{llll}
 & \alpha = \max & \displaystyle\sum_{k\in\mathbb{N}_+} \frac{1}{k} y_k & \\
(P_{IK}) & \quad s.t. & \displaystyle\sum_{k=1}^{q} \frac{1}{k+1} y_k < 1 & q \in \mathbb{N}_+ \\
 & & y_k \in \mathbb{N} & k \in \mathbb{N}_+.
\end{array}
\tag{10}
$$

The following claim, proved in the appendix, shows that P_∞ is a relaxation of P_{IK}, and hence that $\delta_\infty \geqslant \alpha$. In particular, the primal solution defined by setting $y_{s_j} = 1$ for $j \in \mathbb{N}_+$ and 0 elsewhere is feasible for both problems and attains the value α.

Claim 6. *Any solution y feasible for P_{IK} is also feasible for P_∞.*

4 Conclusion and Future Work

This paper introduces a novel model for combinatorial optimization over time. The main contribution is a polynomial time $\alpha\beta$-approximation algorithm when the static combinatorial problem $\mathcal{P}$ admits a β-approximation and its solution set is an independence system (IS). This algorithm is the analog of the total-value heuristic for the integer knapsack, which attains the approximation guarantee α. As the integer knapsack is a special case of IS problem over time for which $\beta = 1$, our analysis does indeed generalize the result for the total-value heuristic. The result is somewhat surprising, as while the integer knapsack admits an FPTAS, for other IS problems with $\beta = 1$ the over time version is APX-hard. In other words, our algorithm is oblivious to this complexity increase. We conclude the paper by stating some of many promising directions for ongoing and future work.

Complexity thresholds for IS problems over time. It would be interesting to narrow the gap between approximability and inapproximability result for IS problems over time in which the static problem is polynomial time solvable ($\beta = 1$). Improving the gap for matching over time is particularly interesting, due to is application in cooperative scheduling.

Matroid optimization over time. Study the special case in which the set system $(A, \mathcal{X})$ on which the problem is defined is a matroid. In particular, either prove the existence of a PTAS (an FPTAS), or prove APX-hardness (strong NP-hardness).

General set systems over time. Identify cases, which do not correspond to independent systems, for which it is possible to decide the existence of a solution over time in polynomial time. For those cases devise approximation algorithms for the maximization and minimization versions.

Joint analysis. A combined analysis of the iterated versions of the density-based heuristic and of the total-value heuristic provides an approximation guarantee of 1.5 for the integer knapsack [16]. It would be interesting to perform a similar study for a general IS over time. Note that in this case there is not a unique natural definition of "iterative" version.

Upper bounds on element multiplicity. It is of particular relevance, both from the viewpoint of theory as well as for applications, to study the over-time extension in which each element $a \in A$ can be used at most u_a times over the time window. This extends the standard binary (or bounded) knapsack, and it would be interesting for example to extend the classical 2-approximation algorithm for binary knapsack to this case, or to prove hardness of approximation.

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. V. Kann, M. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation.* Springer, 1999.

[2] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. (S.) Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. In *Journal of the ACM*, pages 735–744, 2000.

[3] A. Bar-Noy and S. Guha. Approximating the throughput of multiple machines in real-time scheduling. *SIAM J. Comput.*, 31:331–352, February 2002.

[4] P. Berman. A d/2 approximation for maximum weight independent set in d-claw free graphs. *Nordic J. of Computing*, 7:178–184, September 2000.

[5] C. Chekuri, A. Gal, S. Im, S. Khuller, J. Li, R. McCutchen, B. Moseley, and L. Raschid. New models and algorithms for throughput maximization in broadcast scheduling. In *Proceedings of the 8th international conference on Approximation and online algorithms*, WAOA'10, pages 71–82, Berlin, Heidelberg, 2011. Springer-Verlag.

[6] R. J. Duffin and L. A. Karlovitz. An infinite linear program with a duality gap. *Management Science. Journal of the Institute of Management Science. Application and Theory Series*, 12(1):122–134, 1965.

[7] S. P. Fekete and J. Schepers. A new exact algorithm for general orthogonal *d*-dimensional knapsack problems. In Rainer E. Burkard and Gerhard J. Woeginger, editors, *ESA*, volume 1284 of *Lecture Notes in Computer Science*, pages 144–156. Springer, 1997.

[8] S. P. Fekete, J. Schepers, and J. C. van der Veen. An Exact Algorithm for Higher-Dimensional Orthogonal Packing. *Operations Research*, 55(3):569–587, 2007.

[9] L. R. Ford and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6(3):419–433, 1958.

[10] H. W. Hamacher. A time expanded matroid algorithm for finding optimal dynamic matroid intersections. *Zeitschrift für Operations Research. Serie A. Serie B*, 29(5):203–215, 1985.

[11] H. W. Hamacher. Maximal dynamic polymatroid flows and applications. *Discrete Applied Mathematics. The Journal of Combinatorial Algorithms, Informatics and Computational Sciences*, 15(1):41–54, 1986.

[12] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22:463–468, October 1975.

[13] K. Jansen and G. Zhang. On rectangle packing: maximizing benefits. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '04, pages 204–213, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[14] R. Koch, E. Nasrabadi, and M. Skutella. Continuous and discrete flows over time: A general model based on measure theory. *Mathematical Methods Of Operations Research*, 73(3):301–337, 2011.

[15] R. Kohli and R. Krishnamurti. A total-value greedy heuristic for the integer knapsack problem. *Operations Research Letters*, 12(2):65–71, 1992.

[16] R. Kohli and R. Krishnamurti. Joint performance of greedy heuristics for the integer knapsack-problem. *Discrete Applied Mathematics*, 56(1):37–48, 1995.

[17] R. Kohli, R. Krishnamurti, and P. Mirchandani. Average performance of greedy heuristics for the integer knapsack problem. *European Journal Of Operational Research*, 154(1):36–45, 2004.

[18] E. L. Lawler. Fast approximation algorithms for knapsack problems. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 206 –213, 31 1977-nov. 2 1977.

[19] C. C. Lee and D. T. Lee. A simple on-line bin-packing algorithm. *J. ACM*, 32:562–572, July 1985.

[20] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[21] M. L. Pinedo. *Scheduling: Theory, Algorithms and Systems.* Springer, third edition, 2008.

[22] M. Skutella. An introduction to network flows over time. In William Cook, L. Lovász, and J. Vygen, editors, *Research Trends in Combinatorial Optimization*, pages 451–482. Springer Berlin Heidelberg, 2009.

[23] N. J. A. Sloane. Sequences A000058 and A007018. The On-Line Encyclopedia of Integer Sequences, `http://oeis.org`.

Appendices

A Complexity proofs

Theorem 4. *Finding a maximum weighted matching over time is* APX*-complete, even if* $T = 3$*, G is a bipartite graph with* $\Delta(G) = 4$*, and* $w_e = 1$ *for all* $e \in E$.

Proof. We prove Theorem 4 with a simple reduction from 3-SAT. In 3-SAT the input consists of a set X of variables and a set C of clauses, each being a disjunction of at most three literals, and the goal is to find an assignment of truth values to the variables that satisfies all the clauses. 3-SAT is NP-complete even if each variable appears in at most three clauses [20]. Note that this is not true if each clause is required to be a disjunction of *exactly* three (distinct) literals.

Let us prove first NP-hardness of our problem. The reduction generates a graph $G = (V, E)$ as follows. For each variable $x_i \in X$, two *literal nodes* v_i^T, v_i^F and one *variable node* v_i are created, linked by the *variable edges* $\{v_i, v_i^F\}$ and $\{v_i, v_i^T\}$. Then, for each clause $c_j \in C$, two *clause nodes* ω_j and ω'_j are added, together with the *clause edge* $\{\omega_j, \omega'_j\}$. Finally, for each clause $c_j \in C$, the clause node ω_j is connected with a *clause-literal edge* to the (at most three) literal nodes corresponding to the literals appearing in the clause. Figure 2 gives an example of our graph construction. It is easy to see that G is bipartite and has maximum degree $\Delta(G) = 4$. The total duration is $T = 3$. Edge durations τ_e are set to 1 for clause-literal edges, 2 for clause edges, and 3 for variable edges. Finally, the edge weights are set to $w_e = 1$ for all edges.

Given a satisfying assignment $\bar{x}$, a matching over time $\mathcal{S} = \{S_1, S_2, S_3\}$ with value $3(|X|+|C|)$ can be derived as follows. For each variable x_i, we add to S_t for all $t \in [T]$ the variable edge $\{v_i, v_i^F\}$ if $\bar{x}_i = true$, and the variable edge $\{v_i, v_i^T\}$ otherwise. As each variable appears in at

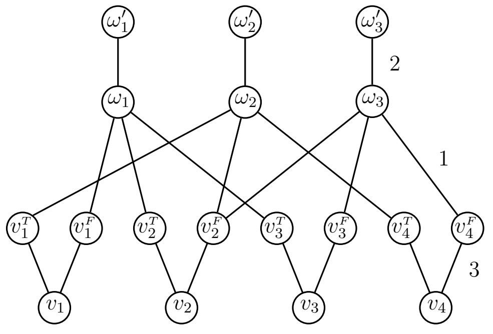


Figure 2: Example of the reduction for a 3-SAT instance with four variables $x_1 \dots x_4$ and the three constraints $c_1 = \overline{x}_1 \vee x_2 \vee x_3$, $c_2 = x_1 \vee \overline{x}_2 \vee x_4$, and $c_3 = \overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4$. The edge duration is depicted for one clause edge, one clause-literal edge, and one variable edge.

most three clauses, we can assume that each literal node is connected to at least one and at most two clause nodes, as otherwise the corresponding variable can be fixed. For each literal we then add one clause-literal edge to S_1 and the other (if any) to S_3. As $\bar{x}$ is a satisfying assignment, for each clause node ω_j one or more incident clause-literal edges will be selected within $[T]$. We then arbitrarily keep one of them in $\mathcal{S}$ and remove the others. This allows to add all clause edges $\{\omega_j, \omega'_j\}$ to S_2 as well as either S_1 or S_3, depending on which clause-literal edge has been chosen for ω_j. The resulting collection $\mathcal{S}$ is a matching over time with value $3(|X| + |C|)$.

Conversely, a matching over time of value $3(|X| + |C|)$ induces a satisfying assignment for the 3-SAT instance. The maximum joint contribution given by clause edges and clause literal edges to any matching over time is $3|C|$, and the maximum contribution for variable edges is $3|X|$. Therefore, $3(|X| + |C|)$ is an upper bound on the value of any matching over time. It is also clear that this value can only be achieved if all variable nodes are matched, which induces a truth assignment $\bar{x}$, and if all clause nodes are matched at all times $t \in [T]$, which due to the edge durations implies that $\bar{x}$ is a satisfying assignment.

Let us now show APX-completeness. As stated in Corollary 3, maximum weighted matching over time admits an α-approximation algorithm, so the problem is in APX. Indeed, for $T = 3$ the approximation factor guarantee obtained from P_T is actually $1.\bar{3}$, as we have $A^2 = \emptyset$. We now prove that maximum weighted matching over time is APX-hard, i.e., there exist some approximation guarantee that cannot be achieved unless P = NP. To do so, we consider the maximization version of the 3-SAT variant we consider, which is denoted MAX-3-SAT(3). This problem is APX-complete (see e.g. [1], Corollary 8.15), and therefore there is some constant approximation factor $\beta > 1$ that cannot be achieved in polynomial time unless P = NP.

Let $\textsc{Opt}$ be the optimum of an instance of MAX-3-SAT(3). As stated above, there exists some $\beta > 1$ such that finding a truth assignment satisfying at least $\frac{1}{\beta}\textsc{Opt}$ clauses is NP-hard. Let $\mathcal{G}$ be a matching over time obtained by some algorithm for the instance constructed in the above reduction. We assume without loss of generality that in $\mathcal{G}$ all variable nodes v_i and all clause nodes ω'_j are matched at $t = 2$ (at least), as otherwise one can construct a matching over time $\mathcal{G}'$ satisfying this condition for which $w(\mathcal{G}') \geqslant w(\mathcal{G})$. Let then $\mathcal{S}^*$ be a maximum weight matching over time satisfying the same condition as $\mathcal{G}$. Then both $\mathcal{G}$ and $\mathcal{S}^*$ induce a truth assignment, and it is easy to see that $w(\mathcal{S}^*) = 3|X| + 2|C| + \textsc{Opt}$ and that $w(\mathcal{G}) = 3|X| + 2|C| + \textsc{Alg}$, where $\textsc{Alg}$ is the number of clauses satisfied by the assignment induced by $\mathcal{G}$.

Assume that $\mathcal{G}$ provides an approximation guarantee $\gamma > 1$, so that $w(\mathcal{G}) \geqslant \frac{1}{\gamma} w(\mathcal{S}^*)$. This gives $\textsc{Alg} \geqslant \frac{1}{\gamma}\textsc{Opt} - 3|X|(1 - \frac{1}{\gamma}) - 2|C|(1 - \frac{1}{\gamma})$. As the best among any random assignment and its complement satisfy at least half of the clauses, we have $\textsc{Opt} \geqslant \frac{1}{2}|C|$. Moreover, assuming each variable appears in at least two clauses, the number of clause-literal edges is at least $2|X|$ and at most $3|C|$, which gives $|X| \leqslant \frac{3}{2}|C|$. Using these bounds we obtain $\textsc{Alg} \geqslant (14\frac{1}{\gamma} - 13)\textsc{Opt}$, which allows to conclude that achieving a factor $\gamma = 14\beta/(13\beta+1)$ is NP-hard. □

Corollary 5. *For a generic set system $(A, \mathcal{X})$, deciding if a solution over time exists is* NP*-complete, even if $T = 3$ and linear optimization on $(A, \mathcal{X})$ is in* P*.*

Proof. A direct consequence of Theorem 5. □

Theorem 5. *Finding a perfect matching over time is NP-complete, even if $|T| = 3$ and G is a bipartite graph with $\Delta(G) = 4$.*

Proof. We modify the reduction in the proof of Theorem 4 as follows. Take two copies of the graph G described in the reduction, and add edges of duration one between the two copies of nodes v_i^T, v_i^F, and w'_j. The graph remains bipartite, and as these new edges are incident to nodes that previously had degree at most three, the maximum degree does not increase.

Now, a perfect matching can be obtained if and only if each node v_i is matched to one of the two nodes v_i^T or v_i^F, and each node ω_j is matched for (at least) one time unit to some literal node v_i^T or v_i^F. The inter-copy edges added above guarantee that any such partial matching over time can be extended to a perfect matching over time. It is then easy to see that a perfect matching induces a satisfying assignment (actually, one for each copy of G, as the assignments corresponding to the two copies do not need to be the same). □

B Proof of Claims for Theorem 3

Claim 1. *$\delta \leqslant \delta_T$ for any $T > 2$.*

Proof. By collecting all bounds (4) for $q \in [T]$ into a single problem, for any given $T > 2$ we can bound δ by the optimum of the following LP:

$$\begin{aligned} \max \quad & \delta \\ s.t. \quad & \sum_{k=1}^{q} \frac{1}{q} \left\lfloor \Gamma_k \frac{q+1}{T} \right\rfloor k z_k \leqslant \frac{1}{\delta} && q \in [T] \\ & \sum_{k=1}^{T} z_k = 1 \\ & z_k \geqslant 0 && k \in [T]. \end{aligned}$$

Applying the variable change $y_k = \delta k z_k$ we obtain

$$\begin{aligned} \max \quad & \delta \\ s.t. \quad & \sum_{k=1}^{q} \frac{1}{q} \left\lfloor \Gamma_k \frac{q+1}{T} \right\rfloor y_k \leqslant 1 && q \in [T] \\ & \sum_{k=1}^{T} \frac{1}{k} y_k = \delta && (11) \\ & \frac{1}{\delta k} y_k \geqslant 0 && k \in [T]. \end{aligned}$$

Noting that $\frac{1}{\delta k} y_k \geqslant 0$ if and only if $y_k \geqslant 0$, as $\delta k \geqslant 1$, and removing δ due to constraint (11), we obtain problem P_T, and hence the guarantee $\delta \leqslant \delta_T$. □

Claim 2. *$\delta_T \leqslant \delta_\infty$ for every $T \in \mathbb{N}_+$.*

Proof. For $T \leqslant 2$ we have $\delta_T = 1$, and we will show that $\delta_\infty > 1$. Therefore, let $T > 2$, let $\bar{y}$ be an optimal solution for P_T, and consider the solution y for P_∞ defined by

$$y_k = \begin{cases} \bar{y}_k & \text{if } k \leqslant T \\ 0 & \text{otherwise.} \end{cases}$$

As the objective function value of y is δ_T, to prove the claim one only needs to show that y is feasible for P_∞. Consider first constraints (6). As $\Gamma_k \geqslant \frac{T}{k+1}$, for $q \leqslant T$ we have

$$1 \geqslant \sum_{k=1}^{q} \frac{1}{q} \left\lfloor \Gamma_k \frac{q+1}{T} \right\rfloor \bar{y}_k \geqslant \sum_{k=1}^{q} \frac{1}{q} \left\lfloor \frac{T}{k+1} \frac{q+1}{T} \right\rfloor \bar{y}_k = \sum_{k=1}^{q} \frac{1}{q} \left\lfloor \frac{q+1}{k+1} \right\rfloor y_k$$

We now show that, for the solution y, each q-th constraint (6) for $q > T$ is implied by the T-th constraint (5), which as $T > 2$ is equivalent to constraint (8). Let $q = T + S$, and, given $k \in [q]$, let $\rho_k, \rho'_k \geqslant 0$ and $r_k, r'_k \leqslant k$ be integers such that $T = \rho_k(k+1) + r_k$ and $S = \rho'_k(k+1) + r'_k$. As $r_k \leqslant k$, we have that $\left\lfloor \frac{T}{k+1} \right\rfloor = \left\lfloor \rho_k + \frac{r_k}{k+1} \right\rfloor = \rho_k$, which applied to constraint (8) gives

$$1 \geqslant \sum_{k=1}^{T} \frac{\Gamma_k}{T} \bar{y}_k = \sum_{k=1}^{T} \frac{\rho_k + 1}{T} \bar{y}_k$$

We can similarly bound constraint (6) by

$$\begin{aligned} \sum_{k=1}^{q} \frac{1}{q} \left\lfloor \frac{q+1}{k+1} \right\rfloor y_k &= \sum_{k=1}^{T} \frac{1}{T+S} \left\lfloor \frac{T+S+1}{k+1} \right\rfloor \bar{y}_k \\ &= \sum_{k=1}^{T} \frac{1}{T+S} \left\lfloor \rho'_k + \rho_k + \frac{r_k + r'_k + 1}{k+1} \right\rfloor \bar{y}_k \leqslant \sum_{k=1}^{T} \frac{\rho'_k + \rho_k + 1}{T+S} \bar{y}_k \end{aligned}$$

where the last inequality is due to $r_k + r'_k + 1 < 2(k+1)$. It remains to show that $\frac{\rho_k+1}{T} \geqslant \frac{\rho'_k+\rho_k+1}{T+S}$ for any $S > 0$, i.e., that $S(\rho_k + 1) - T\rho'_k \geqslant 0$. By substituting T and S, and using the fact that $r_k \leqslant k$, we obtain

$$\begin{aligned} S(\rho_k + 1) - T\rho'_k &= \left(\rho'_k(k+1) + r'_k\right)(\rho_k + 1) - \left(\rho_k(k+1) + r_k\right)\rho'_k \\ &= r'_k(\rho_k + 1) + \rho'_k(k + 1 - r_k) \geqslant r'_k(\rho_k + 1) + \rho'_k \geqslant 0. \end{aligned}$$

We finish the proof of the claim by observing that constraint (7) is also satisfied, as applying $r_k \leqslant k < k+1$ to constraint (8) we obtain

$$1 \geqslant \sum_{k=1}^{T} \frac{\Gamma_k}{T} \bar{y}_k = \sum_{k=1}^{T} \frac{\rho_k + 1}{\rho_k(k+1) + r_k} \bar{y}_k > \sum_{k=1}^{T} \frac{\rho_k + 1}{(k+1)(\rho_k + 1)} \bar{y}_k = \sum_{k=1}^{T} \frac{1}{k+1} y_k.$$

□

Claim 3. *$\delta_\infty = \delta'_\infty$.*

Proof. We apply a classical result in [6], which states that if D_∞ is weakly consistent and has a finite weak value M then P_∞ is strongly consistent and has finite strong value M.

Weak consistence of D_∞ means that the problem $D(r)$ defined by removing all constraints for $k > r$ is feasible for any $r \in \mathbb{N}_+$, which in our case is easily seen as setting $w = 2$ and $v = 0$ gives a dual solution of value 2 that is feasible for all $D(r)$. The weak value M is defined as the limit for $r \to \infty$ of the optimum δ'_r of $D(r)$. The sequence δ'_r is monotonically non-increasing by construction, and as we have seen we have $\delta'_r \leqslant 2$ for all r, which means that M is finite.

We remark that in our case weak duality suffices, as we show explicit primal and dual feasible solutions attaining the same value. Although we did not find explicit references in the literature, weak duality in our case is a consequence of the fact that all variables and all constraint coefficients are nonnegative, which allows to perform the index-reordering on which weak duality in linear programming is based without caring about convergence of the corresponding series (series with positive coefficients can always be rearranged as desired).

□

Claim 4. *The solution $(\tilde{w}, \tilde{v})$ has objective function value α.*

Proof. The objective function value of this dual solution is

$$1 + \sum_{j>2} v_{s_j - 1} = 1 + \sum_{j\geqslant 2} \left(\frac{1}{s_j^2} - \frac{1}{s_{j-1}^2} + \frac{1}{s_{j-1}} \right).$$

As all terms converge to zero for $j \to \infty$, we can reorder the series and get

$$= 1 + \sum_{j\geqslant 2} \left(\frac{1}{s_j^2} - \frac{1}{s_{j-1}^2} \right) + \sum_{j\geqslant 2} \frac{1}{s_{j-1}} = 1 - 1 + \alpha = \alpha.$$

□

Claim 5. *The solution $(\tilde{w}, \tilde{v})$ is feasible for D_∞.*

Proof. Using $s_j = s_{j-1}(s_{j-1}+1)$, let us start by writing $\tilde{v}_{s_{j-1}}$ in the following equivalent form:

$$\begin{aligned}
\tilde{v}_{s_j-1} &= \frac{1}{s_{j-1}} - \frac{1}{s_{j-1}^2} + \frac{1}{s_j^2} &&= \frac{s_j(s_{j-1}+1) - (s_{j-1}+1)^2 + 1}{s_j^2} \\
&= \frac{s_j s_{j-1} - s_{j-1}^2 - 2s_{j-1} + s_j}{s_j^2} &&= \frac{s_j - (s_{j-1}+1) - 1}{s_j(s_{j-1}+1)} + \frac{1}{s_j} \\
&= \frac{1}{s_{j-1}+1} - \frac{1}{s_j(s_{j-1}+1)} &&= \frac{s_j - 1}{s_j} \frac{1}{s_{j-1}+1},
\end{aligned}$$

so as to define

$$\begin{aligned}
\sigma(\tilde{v}, k) &= \sum_{q\geqslant k} \frac{1}{q} \left\lfloor \frac{q+1}{k+1} \right\rfloor \tilde{v}_q &&= \sum_{j: s_j - 1 \geqslant k} \frac{1}{s_j - 1} \left\lfloor \frac{s_j - 1 + 1}{k+1} \right\rfloor \tilde{v}_{s_j - 1} \\
&= \sum_{j: s_j > k} \frac{1}{s_j - 1} \left\lfloor \frac{s_j}{k+1} \right\rfloor \tilde{v}_{s_j-1} &&= \sum_{j: s_j > k} \frac{1}{s_j} \frac{1}{s_{j-1}+1} \left\lfloor \frac{s_j}{k+1} \right\rfloor.
\end{aligned}$$

We want to show that for every $k \in \mathbb{N}_+$ we have

$$\sigma(\tilde{v}, k) \geqslant \frac{1}{k} - \frac{1}{k+1}\tilde{w} = \frac{1}{k(k+1)}.$$

Indeed, we claim that for this dual solution all constraints for $k = s_\ell$, $\ell \in \mathbb{N}_+$ are tight, while the remaining ones are redundant.

Let us first prove that all constraints for $k = s_\ell$ are tight. Note that by definition s_ℓ and $s_\ell+1$ divide s_j for any $j > \ell$. Moreover, one has $\sum_{j \geqslant \ell} \frac{1}{s_j+1} = \frac{1}{s_\ell}$, and in particular $\sum_{\ell \in \mathbb{N}_+} \frac{1}{s_\ell+1} = 1$. We can then write

$$\sigma(\tilde{v}, k) = \frac{1}{s_\ell+1} \sum_{j>\ell} \frac{1}{s_{j-1}+1} = \frac{1}{s_\ell(s_\ell+1)} = \frac{1}{k(k+1)}.$$

It remains to show that the other constraints are satisfied with strict inequality. First note that the nonzero variables appearing in all constraints for $s_{\ell-1} < k < s_\ell$ are the same, that is:

$$\sigma(\tilde{v}, k) = \sum_{j \geqslant \ell} \frac{1}{s_j} \frac{1}{s_{j-1}+1} \left\lfloor \frac{s_j}{k+1} \right\rfloor.$$

Also, we only need to consider $\ell > 2$, as there are no constraints between $s_1 = 1$ and $s_2 = 2$. Using the first term of the summation we bound $\sigma(\tilde{v}, k)$ for for all $s_{\ell-1} < k < s_\ell$ as follows:

$$\sigma(\tilde{v}, k) > \frac{1}{s_\ell} \frac{1}{s_{\ell-1}+1} \left\lfloor \frac{s_\ell}{k+1} \right\rfloor \geqslant \frac{1}{s_\ell} \frac{1}{s_{\ell-1}+1} \left(\frac{s_\ell - k}{k+1} \right) = \frac{s_\ell - k}{s_\ell(s_{\ell-1}+1)} \frac{1}{k+1},$$

where we used the fact that $\left\lfloor \frac{a}{b} \right\rfloor \geqslant \frac{a-b+1}{b}$, for a, b integers. We thus want to get

$$\frac{s_\ell - k}{s_\ell(s_{\ell-1}+1)} \geqslant \frac{1}{k},$$

which is the same as to say that we want to satisfy the quadratic inequality

$$k^2 - ks_\ell + s_\ell(s_{\ell-1}+1) \leqslant 0.$$

This allows to conclude that for $\ell \geqslant 4$ the constraints are satisfied for all k between $s_{\ell-1}+3$ and $s_\ell - s_{\ell-1} - 3$. We verify this by showing that in both extremes the quadratic inequality is satisfied, so that it must be satisfied also for the intermediate values. For $k = s_{\ell-1}+3$ we have

$$\begin{aligned} k^2 - ks_\ell + s_\ell(s_{\ell-1}+1) &= s_{\ell-1}^2 + 6s_{\ell-1} - 2s_\ell + 9 \\ &= s_{\ell-1}^2 + 6s_{\ell-1} - 2s_{\ell-1}(s_{\ell-1}+1) + 9 \\ &= -s_{\ell-1}^2 + 4s_{\ell-1} + 9, \end{aligned}$$

which is negative for $\ell \geqslant 4$. For $k = s_\ell - s_{\ell-1} - 3$ we also get

$$k^2 - ks_\ell + s_\ell(s_{\ell-1}+1) = s_{\ell-1}^2 + 6s_{\ell-1} - 2s_\ell + 9.$$

We are now left to prove that the constraints are satisfied for $k \in [s_{\ell-1}+1, s_{\ell-1}+2]$ and $k \in [s_\ell - s_{\ell-1} - 2, s_\ell - 1]$ for $\ell \geqslant 4$, and for $k \in [3, 5]$, which are the values of k between s_2 and s_3. For $k = s_\ell - 1$ this is easily verified. Indeed, as $k+1 = s_\ell$ divides s_j for all $j \geqslant \ell$, we obtain

$$\sigma(\tilde{v}, k) = \frac{1}{s_\ell} \frac{1}{s_{\ell-1}} > \frac{1}{(s_\ell - 1)s_\ell} = \frac{1}{k(k+1)}.$$

Moreover, noting that $\sigma(\tilde{v}, k)$ is nonincreasing between $s_{\ell-1}$ and s_ℓ, we can say that all constraints for which $s_{\ell-1} < k < s_\ell$ that satisfy

$$\frac{1}{s_\ell} \frac{1}{s_{\ell-1}} \geqslant \frac{1}{k(k+1)}$$

are also satisfied with strict inequality, which is true for $k^2 + k - s_\ell s_{\ell-1} \geqslant 0$. Therefore for $k \geqslant k_\ell^0$, where $k_\ell^0 = \frac{\sqrt{4s_\ell s_{\ell-1}+1}-1}{2}$ is the larger root of the quadratic equation $k^2+k-s_\ell s_{\ell-1} = 0$, the constraints will be satisfied. As we have that

$$k_\ell^0 = \frac{\sqrt{1+4s_\ell s_{\ell-1}}-1}{2} \leqslant \frac{\sqrt{4s_\ell s_{\ell-1}}}{2} = s_{\ell-1}\sqrt{s_{\ell-1}+1},$$

and as $x\sqrt{x+1} \leqslant x^2 - 3$ for $x \geqslant 3$, we have that $k_\ell^0 \leqslant s_{\ell-1}^2 - 3 = s_\ell - s_{\ell-1} - 3$ for $\ell \geqslant 4$. Moreover, as one can verify that $k_3^0 = 3$, we are only left to prove that the constraints are satisfied for $k \in \{s_{\ell-1}+1, s_{\ell-1}+2\}$ for all $\ell \geqslant 4$. For these two cases we bound $\sigma(\tilde{v}, k)$ by the first two terms of the summation:

$$\sigma(\tilde{v}, k) > \frac{1}{s_\ell}\frac{1}{s_{\ell-1}+1}\left\lfloor\frac{s_\ell}{k+1}\right\rfloor + \frac{1}{s_{\ell+1}}\frac{1}{s_\ell+1}\left\lfloor\frac{s_{\ell+1}}{k+1}\right\rfloor,$$

which, defining $r_j = s_j \bmod (k+1)$, gives

$$(k+1)\sigma(\tilde{v}, k) > \frac{s_\ell - r_\ell}{s_\ell}\frac{1}{s_{\ell-1}+1} + \frac{s_{\ell+1} - r_{\ell+1}}{s_{\ell+1}}\frac{1}{s_\ell+1}.$$

We now find explicit values for r_ℓ and $r_{\ell+1}$ for the two values of k that remain to be proved. We will use the fact that $s_j = s_{j-1}(s_{j-1}+1)$ and that $ab \bmod c = (a+\lambda c)(b+\lambda c) \bmod c$ for any $a, b, c \in \mathbb{N}_+$ and $\lambda \in \mathbb{Z}$. Let $\bar{k} = k+1$. For $k = s_{\ell-1}+1$, and hence $\bar{k} = s_{\ell-1}+2$, we can write

$$r_\ell = s_{\ell-1}(s_{\ell-1}+1) \bmod \bar{k} = (s_{\ell-1} - \bar{k})(s_{\ell-1}+1-\bar{k}) \bmod \bar{k} = 2$$

and

$$r_{\ell+1} = s_\ell(s_\ell+1) \bmod \bar{k} = (s_\ell - \bar{k}(s_{\ell-1}-1))(s_\ell + 1 - \bar{k}(s_{\ell-1}-1)) \bmod \bar{k},$$

which as $\bar{k}(s_{\ell-1}-1) = s_{\ell-1}^2 + s_{\ell-1} - 2 = s_\ell - 2$ gives

$$r_{\ell+1} = 6.$$

Let us now consider $k = s_{\ell-1}+2$, for which $\bar{k} = s_{\ell-1}+3$. Then we similarly obtain

$$r_\ell = (s_{\ell-1} - \bar{k})(s_{\ell-1}+1-\bar{k}) \bmod \bar{k} = 6$$

and

$$r_{\ell+1} = (s_\ell - \bar{k}(s_{\ell-1}-2))(s_\ell + 1 - \bar{k}(s_{\ell-1}-2)) \bmod \bar{k},$$

which as $\bar{k}(s_{\ell-1}-2) = s_{\ell-1}^2 + s_{\ell-1} - 6 = s_\ell - 6$ gives

$$r_{\ell+1} = 6 \cdot 7 \bmod \bar{k},$$

which gives $r_{\ell+1} = 6$ for $\ell = 4$ and $r_{\ell+1} = 42$ for $\ell > 4$.

Using the values for r_ℓ and $r_{\ell+1}$, we now show that $(k+1)\sigma(\tilde{v},k) > \frac{1}{k}$. To simplify the proof, as $k \geqslant s_{\ell-1}+1$, let us use the following bound:

$$\begin{aligned}(k+1)\sigma(\tilde{v},k) &> \frac{s_\ell - r_\ell}{s_\ell}\frac{1}{s_{\ell-1}+1} + \frac{s_{\ell+1}-r_{\ell+1}}{s_{\ell+1}}\frac{1}{s_\ell+1} \\ &> \frac{s_\ell - r_\ell}{s_\ell+1}\frac{1}{s_{\ell-1}+1} + \frac{s_{\ell+1}-r_{\ell+1}}{s_{\ell+1}}\frac{1}{s_\ell+1} \\ &> \frac{1}{s_\ell+1}\left(\frac{s_\ell - r_\ell}{k} + 1 - \frac{r_{\ell+1}}{s_{\ell+1}}\right)\end{aligned}$$

We can thus prove that $(k+1)\sigma(\tilde{v},k) > \frac{1}{k}$ by showing that

$$\frac{1}{s_\ell+1}\left(\frac{s_\ell - r_\ell}{k} + 1 - \frac{r_{\ell+1}}{s_{\ell+1}}\right) \geqslant \frac{1}{k},$$

namely, that

$$1 - \frac{r_{\ell+1}}{s_{\ell+1}} - \frac{r_\ell+1}{k} \geqslant 0.$$

For $k = s_{\ell-1}+1$ and $\ell \geqslant 4$ this gives

$$1 - \frac{6}{s_{\ell+1}} - \frac{3}{s_{\ell-1}+1} \geqslant 1 - \frac{6}{s_5} - \frac{3}{s_3+1} > 0.$$

Similarly, for $k = s_{\ell-1}+2$ and $\ell = 4$ we get

$$1 - \frac{6}{s_5} - \frac{7}{s_3+2} > 0,$$

while for $k = s_{\ell-1}+2$ and $\ell \geqslant 5$ we get

$$1 - \frac{42}{s_{\ell+1}} - \frac{7}{s_{\ell-1}+2} \geqslant 1 - \frac{42}{s_6} - \frac{7}{s_4+2} > 0,$$

which concludes the proof. □

Claim 6. *Any solution y feasible for P_{IK} is feasible also for P_∞.*

Proof. Let y_l be a feasible solution to P_{IK}. Constraint (7) is clearly satisfied. Then, assume by absurd that for some $q \in \mathbb{N}_+$ we have that the corresponding constraint (6) is violated, i.e., that

$$\sum_{k=1}^{q} \left\lfloor \frac{q+1}{k+1} \right\rfloor y_k > q.$$

As all coefficients are integer, and y is integer as well, this implies that

$$\sum_{k=1}^{q} \left\lfloor \frac{q+1}{k+1} \right\rfloor y_k \geqslant q+1.$$

On the other hand, we have

$$\sum_{k=1}^{q} \left\lfloor \frac{q+1}{k+1} \right\rfloor y_k \leqslant \sum_{k=1}^{q} \frac{q+1}{k+1} y_k = (q+1)\sum_{k=1}^{q}\frac{1}{k+1}y_k,$$

which leads to the contradiction

$$\sum_{k=1}^{q}\frac{1}{k+1}y_k \geqslant 1.$$

□