Theoretical Computer Science 613 (2016) 79–93

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Capacitated Max-Batching with interval graph compatibilities [☆]

Tim Nonner

IBM Research - Zurich, Switzerland

ARTICLE INFO

Article history:
Received 16 December 2013
Received in revised form 20 October 2015
Accepted 26 October 2015
Available online 26 November 2015
Communicated by P.G. Spirakis

Keywords:
Graph algorithms
Approximation algorithms
Scheduling
Approximation schemes
NP-hardness

ABSTRACT

We consider the problem of partitioning interval graphs into cliques of bounded size. Each interval has a weight, and the weight of a clique is the maximum weight of any interval in the clique. The goal is then to find such a partition of minimum total weight. This graph problem can also be interpreted as a batch scheduling problem. Solving a long-standing open question, we show NP-hardness, even if the bound on the clique sizes is constant. Moreover, we give a PTAS for this case.

1. Introduction

We consider the problem of partitioning interval graphs into cliques of bounded size. Each interval has a weight, and the cost of a clique is the maximum weight of any interval in the clique. Specifically, let V be the intervals of the graph, where each interval $I \in V$ has a weight $w_I \in \mathbb{Q}^+$, and let k be the bound on the clique size. The objective is hence to partition the intervals V into cliques $C_1, C_2, \ldots, C_s$ of at most size k such that $\sum_{i=1}^{s} w_{C_i}$ is minimized, where $w_C := \max_{I \in C} w_I$ for each clique C. We refer to this problem as *Max-Batching*, because we can think of an interval as a *job*, and of a clique as a *batch* of jobs which satisfy the compatibility constraint implied by the interval graph structure. The weight of a job can then be interpreted as the time required to process this job, which implies that the weight of a batch is exactly its processing time if all contained jobs can be processed in parallel. This scenario occurs for example in semiconductor burn-in operations [12, 9,5], where the intervals model valid temperature ranges such that two semiconductors can be burned together in an oven if their temperature ranges overlap. Another application is to interpret the intervals as time intervals during which the respective job needs to be processed, as done by Bechetti et al. [14] in the context of data aggregation. Max-Batching can be generalized to arbitrary graphs instead of interval graphs, as done in [11,9,6]. In this case, the problem is clearly NP-hard, since it contains graph coloring [10].

Previous work. Finke et al. [9] showed that Max-Batching can be solved via a DP (Dynamic Program) in polynomial time for $k = \infty$. A similar result was independently obtained by Becchetti et al. [3] in the context of data aggregation. Moreover, this result was extended by Gijswijt, Jost, and Queyranne [11] to value-polymatroidal cost functions, a subset of the well-known submodular cost functions. Using this result as a relaxation, Correa et al. [6] presented a 2-approximation

[☆] An extended abstract of this paper has appeared at SWAT'10.
E-mail address: tim@nonner.de.

http://dx.doi.org/10.1016/j.tcs.2015.10.047
0304-3975/© 2015 Elsevier B.V. All rights reserved.

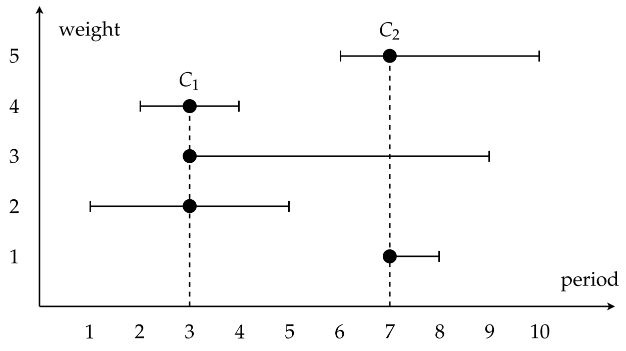

Fig. 1. Geometric interpretation.

algorithm for Max-Batching for arbitrary k. However, it was raised as an open problem in [9,5,6] whether this problem is NP-hard or not.

Note that if the weights on the intervals are uniform, for example $w_I = 1$ for each interval I, then Max-Batching simplifies to finding a clique partition of minimum cardinality where each clique has at most size k [4]. We can also think of this problem as a hitting set problem with uniform capacity k where we want to hit or *stab* intervals with vertical lines which correspond to cliques as illustrated in Fig. 1. Since the natural greedy algorithm solves this problem in polynomial time [9], Even et al. [8] addressed the more complicated case of non-uniform capacities. They presented a polynomial time algorithm based on a general DP approach introduced by Baptiste [2] for the problem of scheduling jobs such that the number of gaps is minimized.

Contributions. We settle the complexity of Max-Batching by proving NP-hardness in Section 5 for all $k \geq 3$, which solves an open problem from [9,5,6]. This is tight, since Max-Batching can be solved in polynomial time for $k = 2$ by using an algorithm for weighted matching. Moreover, we present a DP based PTAS for any constant k in Section 4. It is worth mentioning that this DP differs significantly from the DPs introduced before for the related problems discussed above, but shares some similarities with the DP used by Bansal et al. [1] for unsplittable flow on line graphs. Using the natural geometric interpretation of Max-Batching used in [9], we briefly discuss these approaches in Section 3.

Related work. Also the complementary problem, where we want to partition a graph into independent sets instead of cliques, called *Max-Coloring*, has raised a considerable amount of attention [17,16]. Note that, for $k = \infty$, finding such a partition is equivalent to coloring a graph. Pemmaraju, Raman, and Varadarajan [17] showed that this complementary problem is NP-hard, even for interval graphs and $k = \infty$. Moreover, Pemmaraju, and Raman [16] proved that, for $k = \infty$, a graph class admits a $4c$-approximation algorithm if there is a c-approximation algorithm for the simpler coloring problem. The factor 4 has been improved to e by Epstein and Levin [7]. Using a completely different approach from the one presented in this paper, the author showed that Max-Coloring in interval graphs also admits a PTAS [15]. Finally, note that our results for Max-Batching can be applied to this complementary problem for the graph class of co-interval graphs and constant k.

2. Preliminaries

Let n be the number of intervals, and assume without loss of generality that all endpoints of intervals are distinct and elements in the range $\{1, \ldots, T\}$ for $T := 2n$. Following the temporal interpretation of Bechetti et al. [3] in the context of data aggregation, we refer to an element in $\{1, \ldots, T\}$ as a *period* and assume that each interval contains at least two periods. We also refer to a clique-partition as a *schedule*.

For a schedule σ, define $\text{cost}(\sigma) := \sum_{C \in \sigma} w_C$, and let $\text{OPT} := \text{cost}(\sigma^*)$ denote the cost of an optimal schedule σ^*. We refer to each maximal set of intervals which share the same weight as a *weight class*. Let then $m := |\{w_I \mid I \in V\}|$ denote the number of weight classes. For each batch $C \in \sigma$, there is some period t_C such that $t_C \in I$ for each interval $I \in C$. If t_C is ambiguous, simply choose t_C to be the leftmost such period. Using this we can think of a schedule σ as a function $\sigma : V \to \{1, \ldots, T\}$ that *assigns* each interval I to the period $\sigma(I) = t_C$, where $C \in \sigma$ is the batch with $I \in C$. We briefly explain in the following paragraph why this is a valid description of a schedule.

Consider the set of intervals C assigned by σ to some period t. If $|C| \leq k$, then we can keep C as a batch. Otherwise, we need to further decompose C. The natural way to do this is to first order the intervals in C as $I_1, I_2, \ldots$ such that $w_{I_1} \geq w_{I_2} \geq \ldots$, and then form batches of size k starting with $I_1, I_2, \ldots, I_k$, and so on, which will result in a sequence of batches $C_1, C_2, \ldots$. Each batch in this sequence except the last one has exactly size k. Since this is the optimal way to partition C into batches, a schedule is hence sufficiently defined by how it assigns intervals to periods.

3. Geometric interpretation

If we interpret the weight w_I of each interval I as a vertical height, then we can also think of a batch C as a vertical line with x-coordinate t_C starting at y-coordinate w_C and ending at y-coordinate 0. We say that each interval contained in