Contents lists available at ScienceDirect
Discrete Applied Mathematics
journal homepage: www.elsevier.com/locate/dam

Note

Interval scheduling maximizing minimum coverage

Veli Mäkinen, Valeria Staneva [1], Alexandru I. Tomescu, Daniel Valenzuela [*], Sebastian Wilzbach [2]

Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki, Finland

ARTICLE INFO

Article history:
Received 29 October 2015
Received in revised form 27 July 2016
Accepted 26 August 2016
Available online xxxx

Keywords:
Interval scheduling
Read pruning
Haplotype assembly
Max-flows

ABSTRACT

In the classical interval scheduling type of problems, a set of n jobs, characterized by their start and end time, needs to be executed by a set of machines, under various constraints. In this paper we study a new variant in which the jobs need to be assigned to at most k identical machines, such that the minimum number of machines that are busy at the same time is maximized. This is relevant in the context of genome sequencing and haplotyping, specifically when a set of DNA reads aligned to a genome needs to be pruned so that no more than k reads overlap, while maintaining as much read coverage as possible across the entire genome. We show that the problem can be solved in time $\min\left(O(n^2 \log k/\log n), O(nk \log k)\right)$ by using max-flows. We also give an $O(n \log n)$-time approximation algorithm with approximation ratio $\rho = \frac{k}{\lfloor k/2 \rfloor}$.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

Interval scheduling is a classical problem in combinatorial optimization. The input usually consists of n jobs, such that each job j needs to be executed in the time interval $[s_j, f_j)$, by any available machine. In the most basic variant of this problem, each machine is always available, can process at most one job at a time, and once it starts executing a job it does so until it is finished. The task is to process all jobs using the minimum number of machines [10]. This is solvable in time $O(n \log n)$ [8]. In another problem variant, known as *interval scheduling with given machines*, there are only k available machines, and the execution of each job brings a specified profit. The task is to schedule a maximum-profit subset of jobs. This is also solvable in polynomial time, for example by min-cost flows [1,2]. Some problem variants are NP-hard, for example if each job can be executed only by a given subset of machines [1], or if each machine is available during a specific period of time [3]. See the surveys [10,11] for further references.

Most previous work has focused on either maximizing the profit obtained from executing the jobs, or on minimizing the resources used by the jobs. In this paper we study a new problem variant with a rather different objective function, motivated by a new application of interval scheduling in genome haplotyping with high-throughput DNA sequencing. In this variant, which we call *interval scheduling maximizing minimum coverage*, we need to select a subset of jobs to be executed by a given number k of machines, such that the minimum, over the number of machines that are busy at any given time, is as large as possible. Fig. 1 gives an example. To the best of our knowledge this variant has not been addressed before.

The rest of the paper is structured as follows. In Section 2 we discuss our original motivation in the context of high-throughput DNA sequencing and give the precise problem formulation. In Section 3 we present a reduction to

[*] Corresponding author.
E-mail address: dvalenzu@cs.helsinki.fi (D. Valenzuela).
[1] Currently studying at MIT, Massachusetts. Work conducted as summer intern at UH.
[2] Currently studying at LMU, Munich. Work conducted as exchange student at UH.

http://dx.doi.org/10.1016/j.dam.2016.08.021
0166-218X/© 2017 Elsevier B.V. All rights reserved.

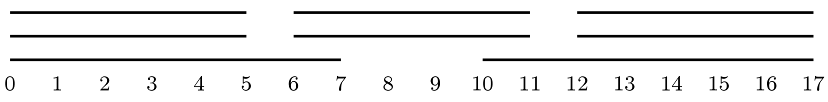

Fig. 1. An instance of interval scheduling maximizing minimum coverage in which 8 intervals are given and $k = 2$ machines are available to execute them. Observe that in all solutions to this problem three disjoint intervals of length 5 need to be removed, leading to a solution that executes 5 jobs and the number of idle machines is never greater than 1. However, in the solutions to the classical interval scheduling with given machines problem, the two intervals of length 7 are removed both in the case when all intervals have the same profit, and in the case when the profit of an interval equals its length.

a max-flow problem, which leads to an $O(n^2 \log k/\log n)$ solution for our problem. In Section 4 we present a tailored max-flow algorithm that runs in $O(nk \log k)$ time, which is faster than the previous when $k = o(n/\log n)$. Since for large k the best complexity is almost quadratic, we also study a way to find approximate solutions: In Section 5 we present an $O(n \log n)$-time $\frac{k}{\lfloor k/2 \rfloor}$-approximation algorithm.

2. Haplotype phasing, read pruning and interval scheduling

High-throughput sequencing is a technique developed over the last decade that can produce millions of DNA fragments, called *reads*, from random positions across the genome of an individual. Depending on the technology, their length can be from hundreds to thousands of characters. Many analyses are carried out by first aligning the reads to a reference genome sequence of the species, and studying, for example, the genetic variations of the individual with respect to the reference (see e.g. [14]). A more detailed analysis, called *haplotype phasing*, also takes into account the fact that in some species, such as humans, each chromosome is present in two copies, inherited from each parent. In this context it is also desirable to assign the genetic variations to the copy of the chromosome where they are present.

Since real data has sequencing and alignment errors, a well-known problem formulation asks for the minimum number of corrections that enables a consistent partitioning of the input set of reads into the two copies of the chromosome they were sequenced from. This problem is called *minimum error correction* and was introduced by Lippert et al. in [13] and proved NP-hard in [4]. A practical algorithm for this problem was proposed in [16], having a time complexity of $O(2^{k-1}m)$, where m is the proportional to the length of the genome, and k is the maximum number of reads covering any position of the genome. This algorithm is particularly useful because its runtime is independent of the read length.

The higher the number of reads, and the more uniform they are distributed across the genome, the more accurate the solution to the minimum error correction problem is in practice. However, the $O(2^{k-1}m)$ time complexity makes this algorithm feasible only for small values of k. In its implementation [16], for every genomic position with too high read coverage, some reads are removed at random. However, this may arbitrarily lead to some other positions having a too low coverage for accurate results. In this paper we study the problem of pruning the read set such that the maximum read coverage is less than a given integer k, and the minimum coverage across all genomic positions is as high as possible.

Our formal definition is as follows. We will represent each read i as an interval $[s_i, f_i)$. We will assume that $0 \leqslant s_i < f_i < N$. Given an interval $[s_i, f_i)$ and a point $p \in [s_i, f_i)$, we say that $[s_i, f_i)$ *covers* p. If $p \in (s_i, f_i)$ we say that $[s_i, f_i)$ *strictly covers* p. Given a set $S = \{[s_i, f_i) \ : \ i \in \{1, \dots, n\} \mid s_i < f_i\}$ of intervals, and a point p we define the *coverage of* p as $\mathsf{cov}_S(p) = |\{[s_i, f_i) \in S | [s_i, f_i) \text{ covers } p\}|$. When clear from the context, the subscript S will be omitted. We also define the *maximum coverage of* S as $\mathsf{maxcov}(S) = max_{p\in[0,N)} \mathsf{cov}_S(p)$. Likewise, we define the *minimum coverage of* S as $\mathsf{mincov}(S) = min_{p\in[0,N)} \mathsf{cov}_S(p)$. Our problem is the following one.

Problem 1 (*Interval scheduling maximizing minimum coverage*).

INPUT. A set $S = \{[s_i, f_i) \ : \ i \in \{1, \dots, n\} \mid s_i < f_i\}$ of intervals and an integer k.
TASK. Find an $S' \subseteq S$ such that $\mathsf{maxcov}(S') \leqslant k$ and maximizing $\mathsf{mincov}(S')$.

Note that if we only keep the first condition, namely $S' \subseteq S$ such that $\mathsf{maxcov}(S') \leqslant k$ our problem would be exactly the one of finding a feasible set of jobs to be scheduled on k machines.

3. The reduction to max-flows

In this section we show that the problem is solvable in time $O(n^2 \log k/\log n)$ by max-flows. First, we consider the decision version of the maximization problem, as follows.

Problem 2 (*Interval scheduling with bounded coverage*).

INPUT. A set $S = \{[s_i, f_i) \ : \ i \in \{1, \dots, n\} \mid s_i < f_i\}$ of intervals, and integers k, t.
TASK. Decide if there exists an $S' \subseteq S$ such that $\mathsf{maxcov}(S') \leqslant k$ and $\mathsf{mincov}(S') \geqslant t$, and if yes, output such S'.