On the pagination problem: scheduling jobs sharing common parts on identical machines

Aristide Grange, Imed Kacem, Sébastien Martin, Sarah Minich
Université de Lorraine, LCOMS EA 7306, F-57070 Metz, France
{aristide.grange, imed.kacem, sebastien.martin, sarah.minich}@univ-lorraine.fr

Keywords: optimization, scheduling.

1 Introduction

1.1 Initial problem description

The pagination problem can be described as follows. We are given a set of symbols, a collection of subsets of these symbols (the tiles) and an integer capacity ($C$). We have to find the minimal number of pages which can contain all tiles without exceeding a maximal number of $C$ symbols per page. A full description can be found in [2]. This description comes from the work of Sindelar, Sitaraman and Shenoy in [5] where the objective is to assign a set of virtual machines (VM) to physical servers while using the fewest possible servers. The VM can share resources (memory pages) if they are assigned to the same server. The NP-hardness proof of this problem is developed in [3].

1.2 Particular case

We modified our initial problem into a new one. In this new problem, we only have two pages, each with an infinity capacity. We want to partition the set of tiles into two sets while minimizing the maximum capacity used on each page. If we rename the pages and call them machines and if we schedule jobs instead of tiles, it is easy to draw a parallel with scheduling. To complete the analogy, the number of symbols in a tile can be called the processing time. Two jobs can overlap if they are assigned to the same machine and if their corresponding tiles share at least one symbol.

2 Our work

First, it is easy to observe that a 2-approximation algorithm exists for the studied problem. Thus, any heuristic with a worst-case ratio greater or equal to 2 is not interesting. The existence of a better approximation ratio is a challenging question, in particular, the existence of a polynomial-time approximation scheme (PTAS).

First special case for the modified problem: Complexity To get started, we worked on the smallest situation which exists: each tile contains precisely two symbols. We proved that this special case is NP-hard using a reduction from the Balanced Complete Bipartite Subgraph problem (BCBS problem). The full description of the BCBS problem can be found in [GT 24] in [1] and its NP-hardness proof is fully developed at the end of [4].
First special case: Label-based approximation algorithm We are currently exploring a lead based on the labelling of the symbols to design an approximation algorithm for our two-machine scheduling problem under sharing symbols’ constraints. Each time a tile is assigned to a machine, its symbols are labelled with the colour $m_i$ corresponding to the machine $M_i$. Before trying to assign the next tile, we compute a pre-treatment. It goes through every tile $t$ we still have to schedule and checks if the number of $m_i$-labelled symbols (i.e., the symbols labelled with the colour $m_i$) is at least $\beta \times |t|$ (with $|t|$ the size of the tile). If this condition is verified, we assume that it is a good choice to assign the tile $t$ to the machine $M_i$, as this assignment would not increase much the makespan.

Second special case: tree-based work We considered the design of a fully PTAS (or FPTAS for short) through the modification of a dynamic programming algorithm for a second type of the two-processor problem, where the structure of tiles is based on a (hyper)tree representation. This problem is close to the one developed in [5]. As a perspective of this work, we will try to find a good upper-bound to make the FPTAS more effective.

3 Conclusions and perspectives

The NP-hardness of the problems (the initial and the modified one) is a good reason to work on approximation algorithms.

In the near future, we have two ideas to design other heuristics for the studied problems. First, we will design an efficient mathematical model to use a primal-dual method. We are also thinking about a heuristic based on scoring the tiles. During the progress of the algorithm, we will give a weight to the tiles left depending on the symbols not assigned yet to the machines. The order in which the tiles are processed will change according to the different weights.

Références