Approximating in exponential time identical parallel machine scheduling with minimum number of tardy jobs

Federico Della Croce¹, Vincent T’kindt²,¹

¹ DIGEP, Politecnico di Torino, CNR, IEIIT
Corso Duca degli Abruzzi 24, 10129 Torino, Italy,
federico.dellacroce@polito.it

² Université de Tours, Laboratoire d’Informatique Fondamentale et Appliquée (EA 6300), ERL CNRS 7002 ROOT, Tours, France,
tkindt@univ-tours.fr

Mots-clés : Scheduling, Parallel machines, Approximation, Exponential-time algorithms.

1 Introduction

This work is at the intersection of the fields of exact exponential-time algorithms and approximation algorithms. We cope with the following question: How can we approach optimal solutions for \( \mathcal{NP} \)-hard problems that do not admit an approximation algorithm? Recent works on optimization problems have shown the opportunity of providing, in such a case, approximation algorithms with a moderately exponential worst-case time complexity (see for instance [4]). In this paper, we present first exponential-time approximation algorithms for scheduling problems.

We consider the problem of scheduling \( n \) jobs on \( m \) identical parallel machines. Each job \( j \) is defined by a processing time \( p_j \), a due date \( d_j \) and it has to be processed non preemptively by one of the machines. Each machine can process one job at a time. The aim is to compute a schedule so that the number of tardy jobs, denoted by \( \sum_j U_j \), is minimized. For any given schedule \( s \), let \( C_j(s) \) be the completion time of job \( j \) : whenever \( C_j(s) > d_j \) we set \( U_j = 1 \) (job is tardy), and \( U_j = 0 \) otherwise (job is early). Without loss of generality, we assume that \( d_1 \leq d_2 \leq \ldots \leq d_n \). Using the standard three-field notation [2] this problem is denoted by \( P|d_j|\sum_j U_j \). It is \( \mathcal{NP} \)-hard even in the case of two machines [1].

We first provide a result that rules out the existence of polynomial time approximation algorithms for the \( P|d_j|\sum_j U_j \) problem, when the ratio is defined by:

\[
\rho = \frac{\sum_j U_j}{\sum_j U_j^*}.
\]

**Theorem 1** The \( P2|d_j|\sum_j U_j \) problem does not admit a polynomial time approximation algorithm with a bounded approximation ration unless \( \mathcal{P} = \mathcal{NP} \).

Theorem 1 implies that providing an approximation algorithm requires an exponential time. Notice that Lenté et al. [3] proposed an exact exponential-time algorithm for solving the \( P|d_j|\sum_j U_j \) problem based on the Sort & Search technique. This algorithm requires \( O^*(\max(m + 1\cdot5)) \) time and space in the worst case. For information, when \( m = 2 \), this complexity becomes \( O^*(3^7) = O(1.7321^n) \). The existence of this EETA creates the challenge of finding an exponential time approximation algorithm which worst-case time complexity is lower than that bound.
2 A branching based approximation algorithm

We focus on a first exponential-time approximation algorithm, referred to as $\text{Bapprox}$, that relies on a branching scheme. Let $k > 0$ be an integer parameter and jobs be grouped into $\lceil \frac{n}{k} \rceil$ batches. Each batch $B_{\ell}$ contains jobs $\{((\ell - 1) \cdot k + 1, \ldots, \ell k\}$, $1 \leq \ell \leq \lceil \frac{n}{k} \rceil$, and there exists a last batch $B_{\lceil \frac{n}{k} \rceil}$ containing the last $(n - k \cdot \lceil \frac{x}{k} \rceil)$ jobs if $\frac{n}{k}$ is not integral. Algorithm $\text{Bapprox}$ builds a binary search tree by branching at each level $k$ on batch $B_k$ and scheduling all its jobs either early or tardy (Figure 1).

Then, for each leaf node we have a set of tardy jobs and a set of early jobs. The existence of a feasible schedule for the latter is tested in $O^*(m n^2)$ time by solving the corresponding $P|\bar{d}_j = d_j|-\text{problem}$. If such a schedule exists then all the tardy jobs are scheduled on any machine after the early jobs.

**Proposition 1** Algorithm $\text{Bapprox}$ admits a worst-case ratio $\rho \leq k$.

**Proposition 2** Algorithm $\text{Bapprox}$ requires $O^*(1 + \frac{k}{m} n^2)$ time and $O^*(m n^2)$ space.

For illustration, in the case of 2 machines, algorithm $\text{Bapprox}$ requires $O(1.5643^n)$ time when $k = 3$, $O(1.4953^n)$ time when $k = 4$, ... Interestingly, this approximation algorithm can be further improved by means of a preprocessing step or even generalized to the case of the weighted number of tardy jobs criterion. This will be shown at the time of the conference.

Références


