Available online at www.sciencedirect.com

SCIENCE DIRECT®

Discrete Optimization 2 (2005) 269–272

DISCRETE OPTIMIZATION

www.elsevier.com/locate/disopt

Note

A comment on scheduling two parallel machines with capacity constraints

Gerhard J. Woeginger

Department of Mathematics and Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Received 23 August 2004; received in revised form 28 June 2005; accepted 30 June 2005
Available online 12 September 2005

Abstract

In a recent paper [Discrete Appl. Math. 130 (2003) 449–467], Yang, Ye, and Zhang investigate the problem of scheduling independent jobs on two identical machines, with a limit on the number of jobs that can be assigned to each single machine, so as to minimize the total weighted completion time of the jobs. They provide a semidefinite programming based, polynomial time approximation algorithm with a worst case ratio of 1.1626.

In this short technical note, we establish the existence of an FPTAS for this problem, and for the corresponding makespan minimization problem.
© 2005 Published by Elsevier B.V.

Keywords: Parallel machine scheduling; Approximation algorithm; Approximation scheme

1. Introduction

Yang et al. [4] consider a scheduling problem with n independent jobs $J_1, \ldots, J_n$. Every job J_j $(j = 1, \ldots, n)$ is specified by its positive integer processing time p_j and by its positive integer weight w_j. Every job is available for processing at time 0, and it must be run without interruption for p_j time units on some machine. There are two identical parallel machines M_1 and M_2, and a given machine capacity bound q. To each single machine, at most q $(n/2 \leqslant q \leqslant n)$ of the jobs may be assigned. Let C_j denote the completion time of job J_j in a given schedule. The goal is to find a schedule that obeys the capacity constraint and minimizes the total weighted job completion time $\sum_{j=1}^{n} w_j C_j$; this problem is denoted by $P2 \,|\, q \,|\, \sum w_j C_j$.

Yang et al. [4] observe that problem $P2 \,|\, q \,|\, \sum w_j C_j$ is NP-hard, and they develop a non-trivial, semidefinite programming based, polynomial time approximation algorithm with a worst case ratio of 1.1626 for this problem. (An approximation algorithm for a minimization problem has worst case ratio ρ, if it always returns a solution with objective value at most a factor ρ above the optimal objective value. A fully polynomial time approximation scheme (FPTAS) is a family of approximation algorithms A_ε over all $\varepsilon > 0$, with worst case guarantees $1 + \varepsilon$ and with running times polynomially bounded in the input size and in $1/\varepsilon$.)

In this short technical note, we observe that problem $P2 \,|\, q \,|\, \sum w_j C_j$ possesses a simple dynamic programming formulation that satisfies certain nice properties. Therefore, this problem belongs to the class of so-called *DP-benevolent* optimization problems, and by a general result of Woeginger [3] it possesses an FPTAS.

E-mail address: gwoegi@win.tue.nl.

1572-5286/$ - see front matter © 2005 Published by Elsevier B.V.
doi:10.1016/j.disopt.2005.06.005

Theorem 1. *Problem $P2 \mid q \mid \sum w_j C_j$ has an FPTAS.*

Yang et al. [4] also mention a manuscript [2] by Bramel et al. that deals with problem $P2 \mid q \mid C_{\max}$, that is, with makespan minimization on two identical machines with a limit on the number of jobs that can be assigned to each single machine. The makespan $C_{\max}$ is the maximum job completion time in some schedule. The manuscript [2] presents a polynomial time approximation algorithm with worst case ratio $\frac{7}{6}$ for $P2 \mid q \mid C_{\max}$. A minor modification of the proof for Theorem 1 yields an FPTAS for $P2 \mid q \mid C_{\max}$:

Theorem 2. *Problem $P2 \mid q \mid C_{\max}$ has an FPTAS.*

The proof method also yields an FPTAS for the two variants $Pm \mid q_1, \ldots, q_m \mid \sum w_j C_j$ and $Pm \mid q_1, \ldots, q_m \mid C_{\max}$: here the number m of machines is a fixed constant that is not part of the input. A schedule is feasible if machine M_i receives at most q_i jobs for $i = 1, \ldots, m$. Finally, we remark that Ageev and Sviridenko [1] provide a polynomial time approximation algorithm with worst case ratio $\frac{3}{2}$ for the (considerably more complex) variant of the problem $P \mid q_1, \ldots, q_m \mid \sum w_j C_j$ where the number of machines is part of the input.

2. The proofs

We start with the proof of Theorem 1. In a preprocessing step, we renumber the jobs such that $p_1/w_1 \leqslant p_2/w_2 \leqslant \cdots \leqslant p_n/w_n$ holds. A straightforward job interchange argument shows that under this numbering, there always exists an optimal schedule in which both machines process the jobs in increasing order of index. We denote by $P = \sum_{j=1}^{n} p_j$ the total job processing time and by $W = \sum_{j=1}^{n} w_j$ the total job weight.

In our dynamic program, we will store certain informations for certain schedules for the first k jobs ($0 \leqslant k \leqslant n$): Every such schedule is encoded by a five-dimensional vector $[n_1, n_2, P_1, P_2, z]$:

- n_i ($i = 1, 2$) specifies the number of jobs assigned to machine M_i.
- P_i ($i = 1, 2$) specifies the overall processing time of all jobs assigned to machine M_i.
- z specifies the weighted sum of completion times of the scheduled jobs.

The state space $\mathscr{S}_k$ consists of all five-dimensional vectors for schedules for the first k jobs. For $k = 0$, the state space $\mathscr{S}_0$ contains $[0, 0, 0, 0, 0]$ as its only element. For $k \geqslant 1$, every schedule $[n_1, n_2, P_1, P_2, z]$ in state space $\mathscr{S}_{k-1}$ can be extended by either placing job J_k at the end of machine M_1 (with a completion time $C_k = P_1 + p_k$) or by placing it at the end of machine M_2 (with a completion time $C_k = P_2 + p_k$). This yields the two schedules

$$[n_1 + 1, n_2, P_1 + p_k, P_2, z + w_k P_1 + w_k p_k]$$

and

$$[n_1, n_2 + 1, P_1, P_2 + p_k, z + w_k P_2 + w_k p_k].$$

We put both schedules into the state space $\mathscr{S}_k$, regardless whether they are feasible (that is: satisfy $n_1 + 1 \leqslant q$ respectively $n_2 + 1 \leqslant q$), or infeasible. In the end, the optimal objective value is extracted from $\mathscr{S}_n$ as

$$\min\{z : [n_1, n_2, P_1, P_2, z] \in \mathscr{S}_n \text{ with } n_1 \leqslant q \text{ and } n_2 \leqslant q\}.$$

The running time of this dynamic programming formulation is pseudo-polynomial, and bounded by $O(nP^2W)$. In the framework of Woeginger [3], problem $P2 \mid q \mid \sum w_j C_j$ belongs to the class of DP-benevolent problems, and thus automatically has an FPTAS. We briefly sketch how the dynamic program can be translated into an FPTAS (this sketch is considerably simpler than working through the exposition in [3]).

The main idea is to iteratively thin out the state space of the dynamic program, to collapse solutions that are 'close' to each other, and to bring the size of the state space down to polynomial. We define a so-called *trimming* parameter Δ as

$$\Delta = 1 + \frac{\varepsilon}{2n}. \tag{1}$$

The trimming of the state spaces is based on the notion of *Δ-domination*: A state $s'=[n_1, n_2, P_1', P_2', z']$ is Δ-dominated by another state $s=[n_1, n_2, P_1, P_2, z]$, if and only if

$$P_1/\Delta \leqslant P_1' \leqslant P_1\Delta \quad \text{and} \quad P_2/\Delta \leqslant P_2' \leqslant P_2\Delta \quad \text{and} \quad z/\Delta \leqslant z' \leqslant z\Delta. \tag{2}$$

As long as the state space $\mathscr{S}_k$ contains a state s' that is Δ-dominated by another state s, we remove this dominated state s'. This removal procedure eventually terminates, and thus yields the trimmed state space $\mathscr{S}_k^*$. Whenever we compute a new state space $\mathscr{S}_k^*$ in the trimmed dynamic program, then we start from the trimmed state space $\mathscr{S}_{k-1}^*$ instead of the original state space $\mathscr{S}_{k-1}$ in the original dynamic program.

Lemma 3. *The cardinality of every trimmed state space $\mathscr{S}_k^*$ is polynomially bounded in the input size and in $1/\varepsilon$.*

Proof. The P_1-coordinate and P_2-coordinate of every state are integers between 0 and P; the z-coordinate is an integer between 0 and WP. For any quintuple (n_1, n_2, f, g, h) of integers, the trimmed state space $\mathscr{S}_k^*$ contains at most one state $[n_1, n_2, P_1, P_2, z]$ with $\Delta^f \leqslant P_1 \leqslant \Delta^{f+1}$, $\Delta^g \leqslant P_2 \leqslant \Delta^{g+1}$, and $\Delta^h \leqslant z \leqslant \Delta^{h+1}$. Since $f, g \leqslant \lceil \log_\Delta(P) \rceil$ and $h \leqslant \lceil \log_\Delta(WP) \rceil$, up to a constant factor the cardinality of $\mathscr{S}_k^*$ is bounded from above by

$$\log_\Delta^3(WP) = \log_2^3(WP) \,/\, \log_2^3(\Delta) \leqslant \log_2^3(WP) \,/\, (\Delta-1)^3 = \log_2^3(WP) \cdot 8n^3/\varepsilon^3.$$

Here we first used the inequality $\log_2(1+x) \geqslant x$ for $0 \leqslant x \leqslant 1$, and then definition (1). Since $\log_2(WP)$ is the number of bits needed to write down the input, the cardinality of $\mathscr{S}_k^*$ is indeed polynomially bounded in the input size and in $1/\varepsilon$. $\square$

Lemma 4. *Every state $s' = [n_1, n_2, P_1', P_2', z']$ in the (un-trimmed) state space $\mathscr{S}_k$ is Δ^k-dominated by some state $s = [n_1, n_2, P_1, P_2, z]$ in the trimmed state space $\mathscr{S}_k^*$.*

Proof. By a straightforward induction on k. $\square$

Lemma 5. *The best feasible solution in the final trimmed state space $\mathscr{S}_n^*$ has an objective value that is at most a factor $1+\varepsilon$ above the objective value of the best feasible solution in the final un-trimmed state space $\mathscr{S}_n$.*

Proof. Let $s'=[n_1, n_2, P_1', P_2', z']$ be the state in $\mathscr{S}_n$ that yields the optimal objective value. By Lemma 4, the trimmed state space $\mathscr{S}_n^*$ contains a state $s = [n_1, n_2, P_1, P_2, z]$ that Δ^n-dominates s'. The cost z is at most a factor

$$\Delta^n = \left(1 + \frac{\varepsilon}{2n}\right)^n \leqslant 1 + \varepsilon$$

above the optimal cost z'. Here we used the inequality $(1+x/m)^m \leqslant 1+2x$ for real numbers x with $0 \leqslant x \leqslant 1$ and for integers $m \geqslant 1$. $\square$

By Lemma 3, the cardinality of every trimmed state space $\mathscr{S}_k^*$ is polynomially bounded in the input size and in $1/\varepsilon$. Therefore, the computation of the trimmed state spaces in the dynamic program can be done within a time complexity polynomially bounded in the input size and in $1/\varepsilon$. Lemma 5 proves the desired worst case guarantee. This completes the proof of Theorem 1.

The proof of Theorem 2 proceeds along the same lines as the proof above, but is slightly simpler. Schedules are encoded by four-dimensional vectors $[n_1, n_2, P_1, P_2]$ (instead of five-dimensional vectors $[n_1, n_2, P_1, P_2, z]$). The computation of the un-trimmed and trimmed state spaces remains the same (except for the missing fifth coordinate). In the end, the optimal objective value is extracted from $\mathscr{S}_n$ as

$$\min\{\max\{P_1, P_2\} : [n_1, n_2, P_1, P_2] \in \mathscr{S}_n \text{ with } n_1 \leqslant q \text{ and } n_2 \leqslant q\}.$$

This completes the proof of Theorem 2.

References

[1] A.A. Ageev, M.I. Sviridenko, Pipage rounding: A new method of constructing algorithms with proven performance guarantee, J. Combin. Optim. 8 (2004) 307–328.

[2] J. Bramel, L. Chan, D. Simchi-Levi, Worst-case analysis of the modified LPT heuristic for the two-machine makespan problem with capacity constraints, Manuscript.
[3] G.J. Woeginger, When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS) ?, INFORMS J. Comput. 12 (2000) 57–75.
[4] H. Yang, Y. Ye, J. Zhang, An approximation algorithm for scheduling two parallel machines with capacity constraints, Discrete Appl. Math. 130 (2003) 449–467.