RC23639 (W0506-095) June 24, 2005
Mathematics

IBM Research Report

A Survey on Minimizing Makespan for the Preemptive Job Shop with Two Machines

Tracy Kimbrel
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

A SURVEY ON MINIMIZING MAKESPAN FOR THE PREEMPTIVE JOB SHOP WITH TWO MACHINES

Tracy Kimbrel

IBM T.J. Watson Research Center
kimbrel@us.ibm.com

We survey results for the job shop scheduling problem $J2\,|pmtn\,|\,C_{\max}$ for the case of two machines with preemption. With respect to polynomial time approximation algorithms, the problem is essentially equivalent to the case in which there are two machines and each operation has unit length. This assumption can be achieved in polynomial time with only ϵ loss by scaling and rounding the inputs to give polynomial-valued operation lengths. Then since preemption is allowed we may take an operation to be composed of multiple unit-length operations (on the same machine). (The assumption can be removed with no loss for all but the last algorithm presented but will be used throughout to simplify descriptions.)

The problem then reduces to a constrained matching problem as follows. We consider each job to be a directed chain of precedence arcs between unit length operations. We seek a matching (set of undirected arcs) such that that there are no directed cycles in the mixed graph containing both the precedence and matching arcs. Each matching arc is between an operation on machine M_1 and an operation on machine M_2 which will be performed concurrently. It is straightforward to derive a valid schedule from this graph.

1 Hefetz and Adiri give a polynomial-time exact solution in the case that each job contains a sequence of unit-time operations that always alternate between M_1 and M_2. [Hefetz and Adiri, 1982]

2 Jansen, Solis-Oba, and Sviridenko give a polynomial-time approximation scheme (PTAS) in case there are a constant number of operations per job (without any transformation to unit-time operations) both for the preemptive and nonpreemptive problems. This algorithm applies to the case of any constant number of machines, not just two. [Jansen et al., 2003]

3 Sevastianov and Woeginger give a linear-time $\frac{3}{2}$-approximation algorithm for minimizing the makespan. The algorithm divides the jobs into two sets S and U. It matches operations as follows. Work from S on M_1 is matched as much as possible to work from U on M_2. No attempt is made to match operations in S on M_2 to operations in U on M_1. Because of this, it is quite simple to find a matching that does not induce any cycles. The authors show that for some instances, an appropriate choice of S and U will yield makespan at most $3L/2$ where L is the maximum of the loads on machines M_1 and M_2. For other instances, the jobs must be divided into three sets, and two iterations of the matching process are needed. Again, for an appropriate partitioning, the makespan is shown to be at most $3L/2$. [Sevastianov and Woeginger, 1998]

4 Kimbrel and Saia match the approximation ratio of 3/2 in the randomized online setting. The algorithm requires only a single random bit and thus is trivially derandomized to also give a linear-time offline algorithm. [Kimbrel and Saia, 2000]

This algorithm starts with an arbitrary permutation π of the n jobs. It chooses randomly between two priority schemes. Under one scheme, priority for machine M_1 is given to jobs according to π and priority for machine M_2 is given according to π^r. Under the second scheme, these priority orderings are reversed.

A schedule is constructed by placing each job in one of two queues, corresponding to the machine required by its first operation, ordered by priority. The online algorithm repeatedly matches the first operations of the jobs at the heads of the two queues and removes those operations from the jobs. Each of these two jobs is reinserted into the opposite queue at the proper position according to the priority ordering if needed (i.e., the next operation is on the other machine). If the operation removed was the last, the job is simply removed from the queue. If all jobs are in the same queue, no match is possible and the first operation in the job at the head is simply removed. The sum of lengths of the schedules given by the two priority schemes is at most 3 times the optimum.

5 Anderson *et al.* give an algorithm with a tighter approximation guarantee of $L + \ell/2$ in terms of the maximum machine load L and the maximum job length ℓ, but this is still 3/2 in the worst case. The algorithm matches work on both machines from each job (except two), and interleaves the work from different jobs in a careful way. [Anderson et al., 2001]

Without loss of generality assume that the load on machine M_1 is L. The algorithm chooses any permutation π such that all jobs with more operations on M_1 than on M_2 precede all other jobs. It considers only matching arcs between operations o_1 and o_2 in jobs j_1 and j_2 such that o_1 is on M_1, o_2 is on M_2, and $j_1 < j_2$. Operations on machine M_2 are ordered as follows: the decreasing priority order consists of the operations in j_1 in precedence order, followed by operations in j_2, and so on. Similarly, operations on machine M_1 are ordered as follows: the decreasing priority order consists of the operations in j_n in precedence order where n is the number of jobs, followed by operations in j_{n-1}, and so on.

The operations on machine M_2 are processed in decreasing priority order. When looking for matches for operations in job j, all operations on M_1 in jobs up to and including job $j-1$ have been released into a pool of operations available for matching. These operations are consumed in decreasing priority order.

Note that this algorithm provides an approximation ratio better than 3/2 except in extreme instances in which a single job contains about half the total work or more.

6 The approximation results above for this problem use as a lower bound the maximum of L and ℓ. Sevastianov and Woeginger note than any approximation algorithm with ratio better than 3/2 for the two machine case would require a new, non-trivial lower bound on the optimal makespan. Bansal *et al.* give such a result. Their algorithm has an approximation ratio of less than 1.45. [Bansal et al., 2005]

In case the largest job has significantly less than half the total work, the algorithm of Anderson et al. can be used. If the largest job has significantly more than half the work, the length ℓ of this job gives a good enough lower bound on the optimal schedule length so that any reasonable schedule (that does not simultaneously idle both machines) has makespan less than 3/2 times the optimal.

If there is a "big" job B containing about half the total work, the algorithm of Bansal et al. attempts to maximize the number of operations of other ("small") jobs that are matched to operations of B. No attempt is made to match any two operations both belonging to jobs other than B. This special case of the matching problem is solved via linear programming relaxation of an integer programming formulation and rounding so that the number of operations performed concurrently with those in B is at least a $(1 - 1/e)$ fraction of the

most possible. Even if the optimal schedule matches all these and all remaining operations among jobs other than B, the lower bound so obtained is shown to be enough to guarantee an approximation ratio less than 3/2.

The LP can be informally described as follows: Consider an operation i of a small job j and imagine traversing the operations k of B in precedence order. At step k match a fraction $x_{i,j,k}$ of i to k, and then discard (i.e., leave unmatched) a fraction $y_{i,j,k}$ of i before moving on to stage $k+1$. These matchings must obey precedence constraints in a fractional sense; that is,

$$\sum_{t=0}^{k-1}(x_{i-1,j,t}+y_{i-1,j,t})-\sum_{t=0}^{k-1}(x_{i,j,t}+y_{i,j,t})-x_{i,j,k}\geq 0.$$

The solution is rounded as follows:

Let $0 \leq s_{i,j,k} \leq 1$ denote the extent to which operation i of job j has been matched to operations $1,\ldots,k-1$ of job B or discarded during the first $k-1$ steps, i.e., $s_{i,j,k} = \sum_{t=1}^{k-1}(x_{i,j,t}+y_{i,j,t})$. Let $v_{i,j,k} = s_{i,j,k} + x_{i,j,k}$; thus $v_{i,j,k} - s_{i,j,k}$ is exactly the extent to which i is matched to k. Also note that $y_{i,j,k} = s_{i,j,k+1} - v_{i,j,k}$. Note that the fractional precedence constraints imply that $s_{i-1,j,k} \geq v_{i,j,k}$.

(a) For each job j choose $u_j \in [0,1]$ uniformly at random.

(b) For each operation i in job j assign i to operation k of B if and only if $s_{i,j,k} \leq u_j < v_{i,j,k}$.

(c) Let $N(k)$ denote the number of operations assigned to k. $N(k)$ is a random variable. If $N(k)=0$, k is not matched to any operation. If $N(k)=1$, match k to the unique i assigned to it. If $N(k)\geq 2$, arbitrarily match k to one of the operations assigned to it, and discard the remaining $N(k)-1$ operations (never to be matched again).

Combining these different cases gives a worst case approximation ratio of about 1.442 for the general case.

References

Anderson, E., Jayram, T., and Kimbrel, T. (2001). Tighter bounds on preemptive job shop scheduling with two machines. *Computing*, 67:83–90.

Bansal, N., Kimbrel, T., and Sviridenko, M. (2005). Job shop scheduling with unit processing times. In *Proceedings of the ACM/SIAM Symposium on Discrete Algorithms*, Vancouver, British Columbia, Canada.

Hefetz, N. and Adiri, I. (1982). An efficient optimal algorithm for the two-machine, unit-time job-shop schedule-length problem. *Mathematics of Operations Research*, 7:354–360.

Jansen, K., Solis-Oba, R., and Sviridenko, M. (2003). Makespan minimization in job shops: a linear time approximation scheme. *SIAM J. Discrete Math.*, 16:288–300.

Kimbrel, T. and Saia, J. (2000). On-line and off-line preemptive two-machine job shop scheduling. *Journal of Scheduling*, 3:355–364.

Sevastianov, S. and Woeginger, G. (1998). Makespan minimization in preemptive two machine job shops. *Computing*, 60:73–79.