Available online at www.sciencedirect.com

SCIENCE DIRECT®

Electronic Notes in Discrete Mathematics 25 (2006) 29–33

Electronic Notes in DISCRETE MATHEMATICS

www.elsevier.com/locate/endm

Performance of a Very Large-Scale Neighborhood for Minimizing Makespan on Parallel Machines

Tobias Brueggemann [1,3] Johann L. Hurink [2,4]

Department of Applied Mathematics, University of Twente,
P.O. Box 217, 7500 AE Enschede, The Netherlands

Tjark Vredeveld [5]

Department of Quantitative Economics, Maastricht University,
P.O. Box 616, 6200 MD Maastricht, The Netherlands

Gerhard J. Woeginger [2,6]

Dept. of Math. and Comp. Sci., Eindhoven University of Technology,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Keywords: local search, performance guarantee, very large-scale neighborhood, parallel machines, makespan.

1571-0653/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.endm.2006.06.058

1 Extended Abstract

We consider the problem of scheduling n jobs with given processing times p_j on m identical parallel machines in order to minimize the makespan. This problem is denoted by $P| |C_{\max}$ and is known to be strongly $\mathcal{NP}$-hard. Since the problem is trivial for $m = 1$, feasible solutions can be given by assignments of jobs to machines.

The workload L_i of a machine i for a given assignment is defined as the sum of processing times of the jobs assigned to machine i. The objective of $P||C_{\max}$ is to minimize the vector of the workloads in the $\mathcal{L}_\infty$-norm, i.e. find an assignment of jobs to machines such that the maximal workload is minimized.

To receive approximative solutions, one may use the following priority based method. We start with an empty schedule and iteratively put a non-scheduled job with longest processing time of all remaining jobs onto the machine with currently minimal workload. This method we call LPT-algorithm and due to Graham [4] it yields a schedule no worse than

$$\frac{C_{\max}^{LPT}}{C_{\max}^{*}} \leq \frac{4}{3} - \frac{1}{3m},$$

where $C_{\max}^{LPT}$ and $C_{\max}^{*}$ denote the makespan received by the LPT-algorithm resp. the optimal makespan. This performance guarantee is proven to be tight.

Another way to receive approximative solutions is inspired by the idea of local search. One starts with an arbitrary assignment of jobs to machines and tries to move a job from its designated machine to another one. This move is applied if either the makespan is decreased or the number of critical machines is reduced, i.e. the number of machines attaining the makespan. The process is iteratively repeated until no such job can be found. The assignment obtained at the end of this iterative improvement procedure we call a *move-optimal* assignment. Due to Finn and Horowitz [3] we receive for move-optimal

[1] supported by the Netherlands Organization for Scientific Research (NWO) grant 613.000.225 (Local Search with Exponential Neighborhoods).

[2] supported by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society).

[3] Email: `t.brueggemann@math.utwente.nl`

[4] Email: `j.l.hurink@math.utwente.nl`

[5] Email: `t.vredeveld@ke.unimaas.nl`

[6] Email: `gwoegi@win.tue.nl`

assignments a performance guarantee of

$$\frac{C_{\max}^{move}}{C_{\max}^{*}} \leq 2 - \frac{2}{m+1},$$

where $C_{\max}^{move}$ denotes the makespan of a move-optimal assignment. Moreover, this bound is tight.

The idea of local search and iterative improvement is often successful in delivering solutions of good quality for hard problems. Local search is roughly described by a neighborhood given for any solution and a method for advancing to an improving solution of the neighborhood. To obtain move-optimal assignments, one may define the neighborhood of a given assignment to consist of all assignments that differ in the designated machine of at most one job. The neighborhood is of size $n(m-1)$ and hence, an improving neighbor (if it exists) can be found in polynomial time.

Considerably amount of work has been carried out to give different neighborhoods and performance guarantees for the considered problem and similar ones. Regarding the problem of minimizing the makespan see Vredeveld [6] and Schuurman and Vredeveld [5] for an overview. If the objective is to minimize total weighted completion time, Brueggemann et al. [2] give a performance guarantee of $\frac{3}{2} - \frac{1}{2m}$ for move-optimal schedules. Over the last years, very large-scale neighborhoods came into the picture. These very large-scale neighborhoods mostly contain an up to exponential number of solutions but allow a polynomial exploration. A survey about very large-scale neighborhood techniques is given by Ahuja et al. [1].

We introduce a very large-scale neighborhood of exponential size (in the number of machines) that is based on a matching in a complete graph. The idea is to partition the jobs assigned to the same machine into two sets. This partitioning is done for every machine with some chosen rule to receive $2m$ *parts*. A new assignment is received by putting to every machine the jobs of exactly two parts. The neighborhood $\mathcal{N}_{split}$ consists of all possible rearrangements of the parts to the machines. The best assignment of $\mathcal{N}_{split}$ can be calculated in time $\mathcal{O}(m \log m)$ by determining the perfect matching having minimum maximal edge weight in an improvement graph, where the vertices correspond to parts and the weights on the edges correspond to the sum of the processing times of the jobs belonging to the parts. Hence, the problem of determining the best neighbor is similar to solving a bottleneck assignment problem, which can be done by ordering the cost-matrix of the assignment problem, so that it fulfills the bottleneck Monge property.

A locally optimal solution of the neighborhood $\mathcal{N}_{split}$ is an assignment for which it is not possible to obtain a better solution by rearranging the parts.

Such an assignment we call *split-optimal* (and can be received by an iterative improvement process). For split-optimal assignments we prove the following:

Theorem 1.1 *Assume that for every machine a partition into two disjoint sets is received in the way that the two sets are itself move-optimal. If $C_{\max}^{split}$ denotes the makespan of a split-optimal solution, then*

$$\frac{C_{\max}^{split}}{C_{\max}^{*}} \leq \frac{2m}{m+1} = 2 - \frac{2}{m+1},$$

where $C_{\max}^{*}$ denotes the optimal makespan. Moreover, this bound is tight.

¿From the previous theorem we can conclude that split-optimal schedules have the same worst-case performance guarantee as move-optimal schedules. But, the worst-case examples have a different structure. In a second step, we try to combine these two neighborhoods and receive the following theorem.

Theorem 1.2 *Assume that for every machine a partition into two disjoint sets is received in the way that the two sets are itself move-optimal. If $C_{\max}^{s+m}$ denotes the makespan of a schedule that is move-optimal* and *split-optimal, then*

$$\frac{C_{\max}^{s+m}}{C_{\max}^{*}} \leq \frac{2m+2}{m+3} = 2 - \frac{4}{m+3},$$

where $C_{\max}^{*}$ denotes the optimal makespan. Moreover, this bound is tight for $m = 2$ and any odd m.

The performance guarantee for schedules that are both, split-optimal and move-optimal, is slightly better than for move-optimal or split-optimal schedules alone. But the asymptotic behavior of the guarantees are still the same, i.e. for $m \to \infty$ we have a tight guarantee of 2 for any local optimal solution of any neighborhood considered so far.

In a last step, we modify the move-neighborhood. A given schedule is called *lexicographic-move-optimal* if no move of a single job leads to a schedule where the ordered vector of machine loads is lexicographically smaller than that of the considered schedule. A lexicographic-move-optimal schedule has the same worst-case performance guarantee as move-optimal schedules, but if such a schedule is also split-optimal, we can prove the following guarantee.

Theorem 1.3 *Assume that for every machine a partition into two disjoint sets is received by using the LPT-algorithm. If $C_{\max}^{s+m}$ denotes the makespan of a schedule that is lexicographic-move-optimal* and *split-optimal, then*

$$\frac{C_{\max}^{s+lm}}{C_{\max}^{*}} \leq \frac{3}{2},$$

where $C^*_{\max}$ denotes the optimal makespan. Moreover, worst-case instances exist that have a ratio of:

$$\frac{C^{s+lm}_{\max}}{C^*_{\max}} = \frac{3}{2} - \frac{1}{2m-2}.$$

Therefore, by combining the neighborhoods in a clever way, we can improve the performance guarantee a lot and get near the guarantee for the LPT-algorithm, which is $4/3$ for $m \to \infty$.

References

[1] R.K. Ahuja, E. Özlem, J. B. Orlin, A.P. Punnen [2002]: *A survey of very large-scale neighborhood search techniques.* In: Discrete Applied Mathematics 123, 75-102.

[2] T. Brueggemann, J.L. Hurink, W. Kern [2006]: *Quality of move-optimal schedules for minimizing total weighted completion time.* To appear in: Operations Research Letters.

[3] G. Finn, E. Horowitz [1979]: *A linear time approximation algorithm for multiprocessor scheduling.* In: Nordisk Tidskrift for Informationsbehandling (BIT), Vol. 19, No. 3, 312-320.

[4] R. L. Graham [1969]: *Bounds on multiprocessing timing anomalies.* In: SIAM Journal on Applied Mathematics, Vol. 17, No. 2, 416-429.

[5] P. Schuurman and T. Vredeveld [2006]: *Performance guarantees of local search for multiprocessor scheduling.* To appear in: Informs Journal on Computing.

[6] T. Vredeveld [2002]: *Combinatorial Approximation Algorithms.* PhD-Thesis. Eindhoven, The Netherlands. ISBN: 90-386-0532-3.