Scheduling on a Constant Number of Machines

F. Afrati[1], E. Bampis[2], C. Kenyon[3], and I. Milis[4]

[1] NTUA, Division of Computer Science, Heroon Polytechniou 9, 15773, Athens, Greece
[2] LaMI, Université d'Evry, Boulevard François Mitterrand, 91025 Evry Cedex, France
[3] LRI, Bât 490, Université Paris-Sud, 91405 Orsay Cedex, France
[4] Athens University of Economics, Dept. of Informatics, Patission 76, 10434 Athens, Greece

Abstract. We consider the problem of scheduling independent jobs on a constant number of machines. We illustrate two important approaches for obtaining polynomial time approximation schemes for two different variants of the problem, more precisely the multiprocessor-job and the unrelated-machines models, and two different optimization criteria: the makespan and the sum of weigthed completion times.

1 Introduction

In the past few years, there have been significant developments in the area of approximation algorithms for $\mathcal{NP}$-hard scheduling problems, see *e.g.* [8] and the references at the end of this paper. Our current, admittedly optimistic, opinion is the following: for any scheduling problem where the schedules can be stretched without unduly affecting the cost function, and in which each job is specified by a constant number of parameters, there should be a way to construct a PTAS by a suitable combination of known algorithmic and approximation techniques.

We present here two approaches for obtaining efficient PTAS that we illustrate by two examples dealing eachone with a different optimization criterion. In the first case, we consider the problem of minimizing the makespan for a *multiprocessor job system* [12]. In this model, we are given a set of jobs J such that each job requires to be processed simultaneously by several processors. In the *dedicated* variant of this model, each job requires the simultaneous use of a prespecified set of processors fix_j. Since each processor can process at most one job at a time, jobs that share at least one resource cannot be scheduled at the same time step and are said to be *incompatible*. Hence, jobs are subject to compatibility constraints. Thus, every job is specified by its execution time p_j and the prespecified subset of processors fix_j on which it must be executed. By t_j we denote the starting time of job j and the completion time of j is equal to $C_j = t_j + p_j$. The objective is to find a feasible schedule minimizing the makespan $C_{\max}$, i.e. the maximum completion time of any job. Using the standard three field notation of Graham *et al.* [14], this problem is classified as $Pm|\text{fix}_j|C_{\max}$. In the second case, we are given n independent jobs that have to be executed on m unrelated machines. Each job j is specified by its execution times $p_j^{(i)}$ on

© Springer-Verlag Berlin Heidelberg 1999

each machine M_i, $i = 1, 2, \ldots, m$, and by its positive weight w_j. The jobs must be processed without interruption, and each machine can execute at most one job at a time. Hence, if t_j is the starting time of job j executed on machine M_i in a particular schedule then the completion time of j is $C_j = t_j + p_j^{(i)}$. The objective is to minimize the weighted sum of job completion times $\sum_{j \in J} w_j C_j$. Using the standard three-field notation, the considered problem is denoted as $Rm| \,|\sum w_j C_j$. For example, one may think of an application where each job is associated with specific deliverables for customers: then the sum of completion times measures the quality of a schedule constructed.

In the next section, we present the most important known results for the dedicated variant of the multiprocessor job problem, and we briefly sketch the principle of the method used in [2] in order to obtain a PTAS. In the third section, we also start with a brief state of the art about the unrelated machines problem and we give the rough idea of the PTAS proposed in [1].

2 Minimizing the makespan

2.1 State of the art

The problem of scheduling independent jobs on dedicated machines has been extensively studied in the past in both the preemptive and the non-preëmptive case. In the *preemptive* case, each job can be at no cost interrupted at any time and completed later. In the *non-preemptive* case, a job once started has to be processed (until completion) without interruption.

The non-preemptive three-processor problem, i.e. $P3|\text{fix}_j|C_{\max}$, has been proved to be strongly NP-hard by a reduction from 3-partition [4,17]. The best constant-factor approximation algorithm is due to Goemans [13] who proposed a $\frac{7}{6}$-algorithm improving the previous best performance guarantee of $\frac{5}{4}$ [11]. A linear time PTAS has been proposed in [2]. For unit execution time jobs, *i.e.* $Pm|\text{fix}_j, p_j = 1|C_{max}$, the problem is solvable in polynomial time through an integer programming formulation with a fixed number of variables [17]. However, if the number of processors is part of the problem, *i.e.* $P|\text{fix}_j, p_j = 1|C_{\max}$, the problem becomes NP-hard. Furhermore, Hoogeveen et al. [17] showed that, for $P|\text{fix}_j, p_j = 1|C_{\max}$, there exists no polynomial approximation algorithm with performance ratio smaller than 4/3, unless $P = NP$.

2.2 The principle of the algorithm

The PTAS presented in [2] is based on the transformation of a preemptive schedule to a non-preemptive one. A simple approach to obtain a feasible non-preemptive schedule from a preemptive schedule S is to remove the preempted jobs from S and process them sequentially in a naïve way at the end of S. The produced schedule has a makespan equal to the makespan of S plus the total processing time of the delayed jobs. However, even if S is an optimal preemptive schedule, the schedule thus produced is almost certainly not close to optimal, since there can be a large number of possibly long preempted jobs. The algorithm in [2] does use a preemptive schedule to construct a non-preemptive schedule in the way just described. However, to ensure that the non preemptive solution is close to optimal, only "short" jobs are allowed to be preempted.

The intuitive idea of the algorithm is the following: first, partition the set of jobs into "long" jobs and "short" jobs. This is similar in spirit to Hall and Shmoys' polynomial time approximation scheme for single-machine scheduling [16]. For each possible schedule of the long jobs, complete it into a schedule of J, **assuming that the short jobs are preemptable**[1]. From each such preemptive schedule, construct a non-preemptive schedule. Finally, among all the schedules thus generated, choose the one that has the minimum makespan. The details are given in [2].

Theorem 1. [2] *There is an algorithm A which given a set J of n independent multiprocessor jobs, a fixed number m of dedicated processors and a constant $\epsilon > 0$, produces, in time at most $O(n)$, a schedule of J whose makespan is at most $(1+\epsilon)C_{\max}^{opt}$.*

This algorithmic paradigm is appropriate for problems in which the objective function is the makespan. Indeed, the same approach has been used in order to obtain PTASs for the parallel variant of the multiprocessor job problem [3], and for scheduling malleable parallel tasks [20].

3 Minimizing the weigthed completion time

3.1 State of the art

The problem of scheduling jobs to minimize the total weighted job completion time is one of the most well-studied problems in the area of scheduling theory. The basic situation is the single-machine case with no release dates; in that case, in 1956 Smith designed a very easy greedy algorithm: sequencing in order of non-decreasing p_j/w_j ratio produces an optimal schedule [41]. The problem $R| \,|\sum C_j$ is also polynomial [18]. However with general weights and $m \geq 2$ machines the problem is $\mathcal{NP}$-hard, even for fixed m and identical machines (*i.e.* when $p_j^{(i)} = p_j$ is independent of the machine) [5] [24].

There has been a lot of recent progress on scheduling on *identical* parallel machines: Kawagachi and Kyan [21] showed an 1.21 approximation algorithm. For any fixed number of identical parallel machines Sahni proposed a PTAS [32]. Woeginger [43] extended this result in the case of uniform parallel machines which run at different but not job dependent speeds. Moreover, Skutella and Woeginger [38] proposed a PTAS for the case where the number of identical parallel machines is an input of the problem $(P| \,|\sum w_j C_j)$.

[1] A job is *preemptable* if it can be at no cost interrupted at any time and completed later.

The situation is not so good for scheduling on *unrelated* machines. A sequence of papers has proposed various approximation algorithms [29,19,36,37]. When the number of machines is a parameter of the problem the last of these papers, due to Schultz and Skutella, provides an $(3/2-\epsilon)$-approximation algorithm. More recently Skutella gave an approximation algorithm with performance ratio 1.5 for a constant number m of machines, $m > 2$, and 1.2752 for the two-machine problem [39]. In [1], an $O(n \log n)$-time approximation scheme (PTAS) for scheduling on general unrelated machines, when the number m of machines is fixed, *i.e.* for the problem $Rm| \quad | \sum w_j C_j$, and an $O(n \log n)$-time approximation scheme (PTAS) for the single-machine-release dates case $1|r_j| \sum w_j C_j$, have been presented. Independently, Chekuri, Karger, Khanna, Stein and Skutella [6] designed a PTAS for scheduling on a constant number of unrelated machines with release dates.

3.2 The principle of the algorithm

The approach of the previous section cannot be used when the considered objective function is the sum of the weighted completion times since each job is now specified by its execution(s) time(s) and its weight, and thus the partition into short and long jobs based on the execution times does not help. The general principle used in [1] in order to obtain a PTAS is that the only really well-understood minsum scheduling problem is Smith's setting (single-machine-no-release-dates), and so the problem is simplified until it closely resembles that setting. Two well-known ingredients have been used:

1. grouping and rounding, as in the classical bin-packing approximation schemes, in order to simplify the problem; and
2. dynamic programming to design an algorithm, once we are close to but not exactly in Smith's setting (Skutella and Woeginger's paper [38] gave us a strong belief that jobs should only interact with jobs that had similar ratios, and hence dynamic programming should work).

For simplicity of notation, we present the principle of the algorithm when there are just two machines, but it should be clear that this holds for any constant number of machines.

We start from a simple but fundamental observation. Consider the restriction of the optimal schedule to each of the two machines: then Smith's ratio rule applies, i.e. the jobs executed on machine M_i are processed in order of non-increasing ratios $r_j^{(i)} = w_j/p_j^{(i)}$. In particular, observe that if two jobs executed on the same machine have the same ratio on that machine, then their relative order does not matter.

The algorithm [1] is based on the following two results:

The first result states that unrelated machines are not all that different from identical machines: if the processing times of a job are very different on the two machines (i.e. if $p_j^{(1)} < \epsilon p_j^{(2)}$ or $p_j^{(2)} < \epsilon p_j^{(1)}$), then we can schedule it on the machine on which it has the shorter processing time. Thus, if a job j satisfies

$p_j^{(1)} < \epsilon p_j^{(2)}$ then we say that j is **M_1-decided**, and similarly if $p_j^{(2)} < \epsilon p_j^{(1)}$ then we say that j is **M_2-decided.**

The second result states that if the ratios of two jobs are very different, then we can neglect their interaction.

Exploiting these two results we define windows of ratios with appropriate size such that any undecided job has its two ratios in the same window or in adjacent windows, and that we only need to worry about interactions between jobs which ratios are in the same window or in adjacent windows. This is the key to the dynamic program.

Theorem 2. [1] *There is a polynomial time approximation scheme for* $Rm| \,|\sum w_jC_j$.

With additional ideas one can extend this approach to deal with single-machine scheduling in the presence of release dates.

Theorem 3. [1] *There is a polynomial time approximation scheme for* $1|r_j|\sum w_jC_j$.

The details of the algorithm can be found in [1].

References

1. F. Afrati, E. Bampis, C. Kenyon, I. Milis, *Scheduling to minimize the weighted sum of completion times, submitted.*
2. A.K. Amoura, E. Bampis, C. Kenyon, Y. Manoussakis, *How to schedule independent multiprocessor tasks, Proc. ESA'97, LNCS No 1284, pp. 1-12, 1997.*
3. A.K. Amoura, E. Bampis, C. Kenyon, Y. Manoussakis, *Scheduling independent multiprocessor tasks, submitted.*
4. J. Blazewicz, P. Dell'Olmo, M. Drozdowski, and M.G. Speranza, *Scheduling Multiprocessor Tasks on Three Dedicated Processors, Information Processing Letters, 41:275-280, 1992.*
5. J.L. Bruno, E.G. Coffman Jr., R. Sethi, *Scheduling independent tasks to reduce mean finishing time, Communications of the Association for Computing Machinery, 17, 382-387, 1974.*
6. C. Chekuri, D. Karger, S. Khanna, C. Stein, M. Skutella, *Scheduling jobs with release dates on a constant number of unrelated machines so as to minimize the weighted sum of completion times, submitted.*
7. C. Chekuri, R. Motwani, B. Natarajan, C. Stein, *Approximation techniques for average completion time scheduling, in Proc. SODA'97, 609-618, 1997.*
8. B. Chen, C.N. Potts, G.J. Woeginger, *A review of machine scheduling: Complexity, algorithms and approximability, Report Woe-29, TU Graz, 1998.*
9. S. Chakrabarti, C.A. Phillips, A. S. Shulz, D. B. Shmoys, C. Stein, J. Wein, *Improved scheduling algorithms for minsum criteria, in F. Meyer auf der Heide and B. Monien, editors, Automata, Languages and Programming, volume 1099 of LNCS, pp. 646-657, Springer, Berlin, 1996.*
10. F. A. Chudak, D. B. Shmoys, *Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds, in Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 581-590, 1997.*

11. P. DELL'OLMO, M. G. SPERANZA, AND Z. TUZA, *Efficiency and Effectiveness of Normal Schedules on Three Dedicated Processors, Submitted.*
12. J. DU, AND J. Y-T. LEUNG, *Complexity of Scheduling Parallel Task Systems, SIAM J. Discrete Math., 2(4):473-487, 1989.*
13. M. X. GOEMANS, *An Approximation Algorithm for Scheduling on Three Dedicated Processors. Disc. App. Math., 61:49-59, 1995.*
14. R.L. GRAHAM, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN, *Optimization and approximation in deterministic sequencing and scheduling , Ann. Discrete Math., Vol. 5, 287-326, 1979.*
15. L. A. HALL, A. S. SHULZ, D. B. SHMOYS, J. WEIN, *Scheduling to minimize average completion time: Off-line and on-line approximation algorithms, Mathematics of Operations Research, 22: 513-544, 1997.*
16. L.A. HALL AND D.B. SHMOYS, *Jackson's rule for single-machine scheduling: making a good heuristic better, Mathematics of Operations Research, 17:22-35, 1992.*
17. J. HOOGEVEEN, S. L. VAN DE VELDE, AND B. VELTMAN, *Complexity of Scheduling Multiprocessor Tasks with Prespecified Processors Allocation, Disc. App. Math., 55:259-272, 1994.*
18. W.A. HORN, *Minimizing average flow time with parallel machines, Operations Research 21, pp. 846-847, 1973.*
19. L.A. HALL, A.S. SCHULTZ, D.B. SHMOYS, J. WEIN, *Scheduling to minimize average completion time, Mathematics of Operations Research, 22, 513-544, 1997.*
20. K. JANSEN, L. PORKOLAB, *Linear-Time approximation schemes for malleable parallel tasks, Technical Report MPI-I-98-1-025, Max Planck Institute, also in Proceedings of SODA'99.*
21. T. KAWAGUCHI, S. KYAN, *Worst case bound of an LRF schedule for the mean weighted flow-time problem, SIAM J. on Computing, 15, 1119-1129, 1986.*
22. M. X. GOEMANS, *Improved approximation algorithms for scheduling with release dates, in Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 591-598, 1997.*
23. J. LABETOULLE, E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN, *Preemptive scheduling of uniform machines subject to release dates, in W.R. Pulleyblanck (ed.) Progress in Combinatorial Optimization, Academic Press, 245-261, 1984.*
24. J.K. LENSTRA, A.H.G. RINNOOY KAN, P. BRUCKER, *Complexity of machine scheduling problems, Annals of Discrete Mathematics, 1, 343-362, 1977.*
25. J. K. LENSTRA, D. B. SHMOYS, E. TARDOS, *Approximation algorithms for scheduling unrelated parallel machines, Mathematical programming 46:259-271, 1990.*
26. R. H. MÖHRING, M. W. SCHÄFFTER, A. S. SHULZ, *Scheduling jobs with communication delays: Using unfeasible solutions for approximation, in J. Diaz and M. Serna, editors, Algorithms-ESA'96. volume 1136 of LNCS, pp. 76-90, Springer, Berlin 1996.*
27. A. MUNIER, M. QUEYRANNE, A. S. SHULZ, *Approximation bounds for a general class of precedence-constrained parallel machine scheduling problems, in R. E. Bixby, E. A. Boyd, R. Z. Rios-Mercado, editors, Integer Programming and Combinatorial Optimization, volume 1412 of LNCS, pp. 367-382, Springer, Berlin, 1998.*
28. C. PHILIPS, C. STEIN, J. WEIN, *Scheduling job that arrive over time in Proc. 4th Workshop on Algorithms and Data Structures (1995), to appear in Mathematical Programming.*
29. C. PHILIPS, C. STEIN, J. WEIN, *Task scheduling in networks, SIAM J. on Discrete Mathematics, 10, 573-598, 1997.*

30. C. PHILIPS, C. STEIN, J. WEIN, *Minimizing average completion time in the presence of release dates, Mathematical Programming 82:199-223, 1998.*
31. C. A. PHILLIPS, A. S. SHULZ, D. B. SHMOYS, C. STEIN, J. WEIN, *Improved bounds on relaxations of a parallel machine scheduling problem, Journal of Combinatorial Optimization, 1:413-426, 1998.*
32. S. SAHNI, *Algorithms for scheduling independent tasks, Journal of the Association for Computing Machinery, 23, 116-127, 1976.*
33. M. W. P. SAVELSBERGH, R. N. UMA, J. M. WEIN, *An experimental study of LP-based approximation algorithms for scheduling problems, In Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 453-462, 1998.*
34. L. SCHRAGE, *A proof of the shortest remainig processing time processing discipline, Operations Research 16, pp. 687-690, 1968.*
35. A. S. SHULZ, *Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds, in W. H. Cunningham, S. T. McCormick, and M. Queyranne, editors, Integer Programming and Combinatorial Optimization, volume 1084 of LNCS, pp. 301-315, Springer, Berlin, 1996.*
36. A.S. SCHULZ, M. SKUTELLA, *Random-based scheduling: New approximations and LP lower bounds, In J. Rolimmm ed., Randomization and Approximation Techiques in Computer Science, LNCS 1269, 119-133, 1997.*
37. A.S. SCHULZ, M. SKUTELLA, *Scheduling-LPs bear probabilities: Randomized approximations for min-sum criteria, In R. Burkard and G.J. Woeginger eds, ESA'97, LNCS 1284, 416-429, 1997.*
38. A.S. SCHULZ, G.J. WOEGINGER, *A PTAS for minimizing the weigthed sum of job completion times on parallel machines, In Proceedings of STOC'99.*
39. M. SKUTELLA, *Semidefinite relaxations for parallel machine scheduling, In Proc. FOCS, 472-481 1998.*
40. M. SKUTELLA, *Convex quadratic programming relaxations for network scheduling problems, In Proc. ESA'99, 1999, to appear.*
41. W. SMITH, *Various optimizers for single-stage production, Naval Res. Logist. Quart. Vol. 3, 59-66, 1956.*
42. D. SHMOYS, E. TARDOS, *An approximation algorithm for the generalized assignment problem, Mathematical Programming, 62, 461-474, 1993.*
43. G.J. WOEGINGER, *When does a dynamic programming formulation guarantee the existence of an FPTAS, Report Woe-27, Institut für Mathematik B, Technical University of Graz, Austria, April 1998.*