A Note on Exact Algorithms for the Identical Parallel Machine Scheduling Problem

Mauro Dell'Amico

DISMI, Università di Modena e Reggio Emilia, Italy; dellamico@unimore.it

Silvano Martello

DEIS, Università di Bologna, Italy; smartello@deis.unibo.it

Abstract

A recently published paper by E. Mokotoff presents an exact algorithm for the classical $P||C_{\max}$ scheduling problem, evaluating its average performance through computational experiments on a series of randomly generated test problems. It is shown that, on the same types of instances, an exact algorithm proposed 10 years ago by the authors of the present note outperforms the new algorithm by some orders of magnitude.

Given n *jobs* with associated processing times $p_1, \ldots, p_n$, and m parallel identical *machines*, each of which can process at most one job at time, the *Parallel Machine Scheduling Problem* is to assign each job to exactly one machine so that the maximum completion time of a job (*makespan*) is minimized. Using the three-field classification introduced in Graham, Lawler, Lenstra and Rinnooy Kan [3], the problem is denoted in the scheduling literature as $P||C_{\max}$. We assume, as is usual, that the processing times are positive integers and that $1 < m < n$.

The problem is known to be $\mathcal{NP}$-hard in the strong sense (see Garey and Johnson [2]). In a recently published paper, Mokotoff [4] presented a cutting plane algorithm for its exact solution, based on iterated addition of valid inequalities to the ILP model, and solution of the LP relaxation of the resulting problem. In an older paper, Dell'Amico and Martello [1] had presented a branch-and-bound algorithm for the same problem, based on sophisticated lower and upper bound computations and the use of dominance criteria.

The algorithm by Mokotoff [4] has been implemented by the author using the simplex and branch-and-bound methods included in the CPLEX 5.6 callable library. It has been computationally tested, on a Pentium 500, on three classes of random instances obtained by uniformly generating the p_i values in the ranges $[1, 100]$, $[10, 100]$ and $[50, 100]$, for various instance sizes. We executed the C implementation of the Dell'Amico and Martello [1] algorithm on instances generated in the same way. As we could not find a Pentium 500, we used a slightly slower Celeron 434 MHz. Table 1 compares, on small-size instances, our

Table 1: Small-size test sets. Average CPU times over 10 instances.
Seconds of Pentium 500 (Mokotoff) or Celeron 434 (Dell'Amico-Martello).

		$p_j \in [1, 100]$		$p_j \in [10, 100]$		$p_j \in [50, 100]$	
m	n	Mokotoff	Dell'Amico-Martello	Mokotoff	Dell'Amico-Martello	Mokotoff	Dell'Amico-Martello
3	5	0.06	0.000005	0.02	0.000007	0.06	0.000010
3	6	0.01	0.000010	0.02	0.000014	0.06	0.000007
3	7	0.09	0.000024	0.08	0.000038	0.10	0.000019
3	8	0.10	0.000036	0.09	0.000076	0.12	0.000131
3	9	0.05	0.000040	0.12	0.000101	0.11	0.000073
3	10	0.06	0.000100	0.12	0.000210	0.22	0.000278
3	11	0.14	0.000113	0.22	0.000355	0.54	0.000256
3	12	0.08	0.000157	0.12	0.000112	0.38	0.000361
3	13	0.14	0.000125	0.19	0.000009	1.38	0.000618
3	14	0.08	0.000012	0.15	0.000093	1.48	0.000590
3	15	0.22	0.000014	0.19	0.000014	0.27	0.000011
4	5	0.01	0.000002	0.01	0.000001	0.01	0.000001
4	6	0.03	0.000003	0.01	0.000003	0.07	0.000008
4	7	0.02	0.000018	0.02	0.000022	0.05	0.000028
4	8	0.06	0.000021	0.08	0.000017	0.12	0.000021
4	9	0.04	0.000076	0.12	0.000102	0.25	0.000047
4	10	0.03	0.000043	0.30	0.000088	0.74	0.000373
4	11	0.09	0.000215	0.90	0.000297	0.99	0.001426
4	12	0.57	0.000210	1.17	0.000406	0.95	0.000448
4	13	2.98	0.000537	5.00	0.000841	15.74	0.000921
4	14	0.62	0.000300	0.14	0.000403	39.84	0.000608
4	15	0.57	0.000290	0.49	0.000256	47.06	0.001546
5	6	0.01	0.000002	0.02	0.000001	0.01	0.000001
5	7	0.01	0.000008	0.01	0.000011	0.03	0.000010
5	8	0.01	0.000013	0.02	0.000021	0.04	0.000028
5	9	0.02	0.000022	0.08	0.000017	0.02	0.000030
5	10	0.05	0.000034	0.06	0.000072	0.06	0.000029
5	11	0.37	0.000092	0.30	0.000133	3.61	0.000231
5	12	0.09	0.000126	1.81	0.000324	17.96	0.000471
5	13	1.68	0.000183	5.06	0.000359	74.23	0.000930
5	14	5.39	0.000201	30.98	0.000414	50.04	0.000926
5	15	2.85	0.000247	0.29	0.000925	52.73	0.001007

CPU times with those taken from [4]. All entries give average values computed over 10 instances. As the CPU times of the older algorithm are well below the precision of the timing routine, each instance was solved 1 000 times, and the total time accumulated over the 10 instances was divided by 10 000. In Table 2 we compare the same algorithms on larger instances (also used in [4]) with p_i values uniformly random in the range $[1, 100]$. The entries give the average CPU time and the number of instances (out of 10) not solved to optimality. The results show that the Dell'Amico and Martello [1] algorithm is faster than its competitor by 3–4 orders of magnitude on the small-size instances of Table 1, and even more on the large-size instances of Table 2. In addition, it was able to solve to optimality all the generated instances.

Table 2: Large-size test sets. Average CPU times over 10 instances; p_j random in $[1, 100]$. Seconds of Pentium 500 (Mokotoff) or Celeron 434 (Dell'Amico-Martello).

		Mokotoff		Dell'Amico-Martello	
m	n	time	unsolved instances	time	unsolved instances
3	20	0.12	0	0.000014	0
3	50	0.10	0	0.000017	0
3	100	0.22	0	0.000013	0
3	200	0.12	0	0.000020	0
3	500	0.01	0	0.000044	0
5	20	0.57	0	0.000753	0
5	50	0.35	0	0.000021	0
5	100	0.67	0	0.000026	0
5	200	1.23	0	0.000034	0
5	500	0.01	0	0.000057	0
15	20	0.01	0	0.000004	0
15	50	864.47	2	0.000062	0
15	100	329.66	0	0.000059	0
15	200	348.79	0	0.000104	0
15	500	83.54	0	0.000147	0
100	200	2042.08	0	0.058836	0
100	500	2768.05	2	0.000900	0
100	1000	5098.40	0	0.002001	0

Acknowledgements

We thank the Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) and the Consiglio Nazionale delle Ricerche (CNR), Italy, for the support given to this project.

References

[1] M. Dell'Amico and S. Martello, 1995. Optimal scheduling of tasks on identical parallel processors, *ORSA Journal on Computing* 7, 191–200.

[2] M.R. Garey and D.S. Johnson, 1979, *Computer and Intractability: a Guide to the Theory of NP-Completness,* Freeman, San Francisco.

[3] R.L Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey, *Annals of Discrete Mathematics 5,* 287-326.

[4] E. Mokotoff, 2004. An Exact Algorithms for the Identical Parallel Machine Scheduling Problem, *European Journal of Operational Research 152,* 758–769