Heuristics for Scheduling on Restricted Identical Machines

Juan J. Gálvez
DIIC, Computer Science Faculty
University of Murcia, Spain
Email: jjgalvez@um.es

Pedro M. Ruiz
DIIC, Computer Science Faculty
University of Murcia, Spain
Email: pedrom@um.es

Antonio F. G. Skarmeta
DIIC, Computer Science Faculty
University of Murcia, Spain
Email: skarmeta@um.es

***Abstract*—In this paper we propose a simple efficient algorithm for the problem of job scheduling on restricted identical machines, with the objective of minimizing makespan. This problem is a special case of $R\,||\,C_{max}$, where a job can only be assigned to a subset of machines. The algorithm runs in $O(n \log n + S)$ where n is the number of jobs and S the sum of valid machines of every job. Experimental results show that it can compete with the best algorithms in terms of the fitness of the solutions, while being more efficient.**

I. Introduction

We consider the problem of scheduling n independent jobs on m identical machines without preemption, with the restriction that a job can only be assigned to a *subset* of machines. The scheduling problem appears in many disciplines, and concerns the allocation of a set of limited resources to activities, with the objective of optimizing one or more performance measures. In communications, this problem frequently appears as a load balancing problem; examples include the assignment of users to servers [1], and flows to gateways [2].

The restricted machines model appeared in the work by Gairing et al. [3]. This model reflects a special case of the job scheduling problem on unrelated machines [4], which is NP-hard. The focus of this paper is on developing a simple efficient algorithm for the problem of scheduling to minimize the makespan (i.e. the maximum load on a machine). We develop heuristics for a List Scheduling (LS) based algorithm. LS is a well-known simple combinatorial algorithm often used for scheduling problems. We evaluate the empirical performance of the algorithm.

The rest of the paper is organized as follows. In section II we review related work. In section III we define the restricted scheduling problem. Section IV explains the proposed algorithm. Section V shows experimental results. In Section VI we summarize our conclusions.

II. Related Work

The problem we consider is a special case of job scheduling on unrelated machines. This problem was shown to be NP-hard. Lenstra et al. [4] proved that unless $\mathcal{P} = \mathcal{NP}$, there is no polynomial-time approximation algorithm for the optimum schedule with approximation factor less than $\frac{3}{2}$.

The first published approximation algorithms for scheduling on unrelated machines were based on list scheduling [5], [6]. Many more methods have been proposed since then, with techniques ranging from combinatorial approaches with partial enumeration to integer programming with branch-and-bound and cutting planes [7], [8].

One of the best algorithms recently proposed for this problem was presented by Gairing et al. in [9]. A similar approach was previously considered for the special case of *restricted* machines in [3]. The advantages of their approach is that it is a fairly simple combinatorial algorithm, which achieves an approximation factor of 2, and runs faster than previous methods with the same approximation guarantee.

Our approach is based on list scheduling using heuristics adapted for the particular problem of restricted scheduling. It is very simple, and owing to its simplicity, is faster than previous methods.

III. Restricted Scheduling Problem

This is a non-preemptive job scheduling problem with n jobs and m identical machines. $\mathbb{J}$ and $\mathbb{M}$ are the set of jobs and machines, respectively. The load of a job j is w_j, and equals its processing time. Let $W = \sum_{j \in \mathbb{J}} w_j$. A job j can only be assigned to a subset of machines $S_j \subseteq \mathbb{M}$. We call this the set of *valid machines* of the job. The number of valid machines of job j is $|S_j|$. The sum of valid machines is $S = \sum_{j \in \mathbb{J}} |S_j|$. $A(j)$ represents the machine assigned to job j.

The objective is to find an assignment of jobs to machines that minimizes the makespan of the schedule.

Following the notation of Graham et al. [10], the problem is a special case of $R\,||\,C_{max}$ with the following additional restriction: $\forall j \in \mathbb{J} : p_{ij} \in \{w_j, \infty\}\ \forall i \in \mathbb{M}$. That is, the processing time of a job on a machine is either the weight of the job or ∞ if the job cannot be processed on the machine.

IV. Scheduling Algorithm

We propose a LS-based algorithm called RSA. LS is frequently used in scheduling problems due to its simplicity and the fact that any optimal schedule can be constructed with an appropriately chosen list. The key therefore relies on generating the list of jobs. To this end, we define a special order for jobs.

The LS-based algorithm (Alg. 1) works as follows: after sorting the list of jobs, it iteratively assigns each job to the current least loaded machine.

Algorithm 1 LS-based algorithm.

```
1: Sort $\mathbb{J}$ // Comparison sort based on Algorithm 2
2: for $j$ in $\mathbb{J}$ do
3:     $bestMachine \leftarrow \arg\min_{i \in S_j} i.load$
4:     $bestMachine.load \leftarrow bestMachine.load + w_j$
5:     $A(j) \leftarrow bestMachine$
```

Algorithm 2 Job comparison function used for sorting.

```
1: function less_than(j,k) do
2:     if $(|S_j| = 1)$ and $(|S_k| > 1)$ then
3:         return true
4:     if $(|S_j| > 1)$ and $(|S_k| = 1)$ then
5:         return false
6:     if $\frac{w_j}{|S_j|} > \frac{w_k}{|S_k|}$ then
7:         return true
8:     else
9:         return false
```

The order of jobs is defined as follows. Jobs with only *one* valid machine are selected first to be assigned before any other jobs (lines 2-5 of Alg. 2). For all other cases (lines 6-9), a heuristic similar to Longest Processing Time (LPT) first is used. This heuristic takes into account the number of valid machines of each job. The idea is that jobs with more valid machines should be chosen last, because they offer more opportunities to balance load, and these opportunities should not be used up too early. In addition, choosing the jobs with more load first also remains important, which leads to considering both factors in the decision.

Theorem 1. *The time complexity of the algorithm in the worst case is $O(n \log n + S)$.*

Proof:

In line 1, the list of jobs is sorted by a comparison sort, requiring $O(n \log n)$.

The loop of lines 2-5: Line 3 requires traversing the valid machines of the job. The loop runs in $O(n\frac{\sum_{j\in\mathbb{J}}|S_j|}{n} + n) = O(S+n) = O(S)$.

■

V. Experimental results

We present computational results to study the performance of the algorithm. For comparison, we have chosen a subset of previously proposed algorithms. We want to compare against LS algorithms which are similar in simplicity and running time to ours, and with Gairing's algorithm for the reasons mentioned in section II. We have selected the following: LS algorithm using LPT heuristic; Algorithm 2 of Davis and Jaffe [6]; and algorithm of Gairing [3]. We call them LPT, DAVIS and GAIRING, respectively. The algorithm of Davis et al. was designed for the more general unrelated machine problem. We don't choose algorithms proposed by Ibarra [5] because most are equivalent or worse than LPT when applied to the restricted scheduling problem, and because the algorithms later proposed by Davis offer better approximation guarantees. The algorithm by Gairing was proposed for the restricted scheduling problem. We also test an obvious modification of the LPT heuristic which first assigns jobs with $|S_j| = 1$. We call it LPTA.

TABLE I
Benchmark set parameters.

Parameter	Value
m	$[3, 20]$
n	$[1, 25m]$
w_j	uniform distribution in $[1, 20]$
$\|S_j\|$	distributions $\mathcal{U}_1$ and $\mathcal{G}_1$
S_j	probability samples $\mathcal{U}_2$ and $\mathcal{G}_2$

Because the optimum is unknown, we measure the approximation ratio to a lower bound of the optimum. We define the optimal lower bound as:

$$C^\lambda = \max\{C^1_{max}, \lceil \frac{W}{m} \rceil\} \tag{1}$$

C^1_{max} is the makespan when only the jobs with *one* valid machine are assigned.

A problem instance is given by multiple parameters, listed in Table I. To generate problem instances, the first two parameters m and n take on fixed values (shown in Table I). We generate all possible pairs of (m, n) values. For each pair, 100 random problem instances are generated. In each instance, w_j is generated from a uniform distribution in interval $[1, 20]$. $|S_j|$ can be generated from two distributions $\mathcal{U}_1$ and $\mathcal{G}_1$. $\mathcal{U}_1$ is a uniform distribution in $[1, m]$ and $\mathcal{G}_1$ is a geometric distribution with $p = \frac{1}{3}$. To select S_j, we use two probability samples: $\mathcal{U}_2$ and $\mathcal{G}_2$. $\mathcal{U}_2$ generates indexes to the list of machines from a uniform distribution, and $\mathcal{G}_2$ from a geometric distribution with $p = \frac{4}{m}$. When using geometric distributions, if the generated value is above the domain upper bound, it is truncated to the upper bound.

Given the above, we can distinguish between four sets of problem instances:

- Model A: using $\mathcal{U}_1$ and $\mathcal{U}_2$
- Model B: using $\mathcal{G}_1$ and $\mathcal{U}_2$
- Model C: using $\mathcal{U}_1$ and $\mathcal{G}_2$
- Model D: using $\mathcal{G}_1$ and $\mathcal{G}_2$

Each set contains a total of 517500 instances.

The results for each set are shown in Fig. 1. The x axis shows the number of machines and the y axis shows C/C^λ where C is the makespan calculated by the algorithms. Each point is the average of all scenarios which fall in that category. Confidence intervals are shown at 95% level.

In these tests, RSA is the best performing in some cases and in others performs similar to GAIRING. We can see that the obvious modification of the LPT heuristic of first assigning the jobs with $|S_j| = 1$ can substantially improve the solution. The heuristic of RSA of taking the number of valid machines of each job into account also contributes to a better solution, specially when machine sampling is not based on a uniform sample, i.e. some machines have more probability of being selected than others (models C and D). Note than in models

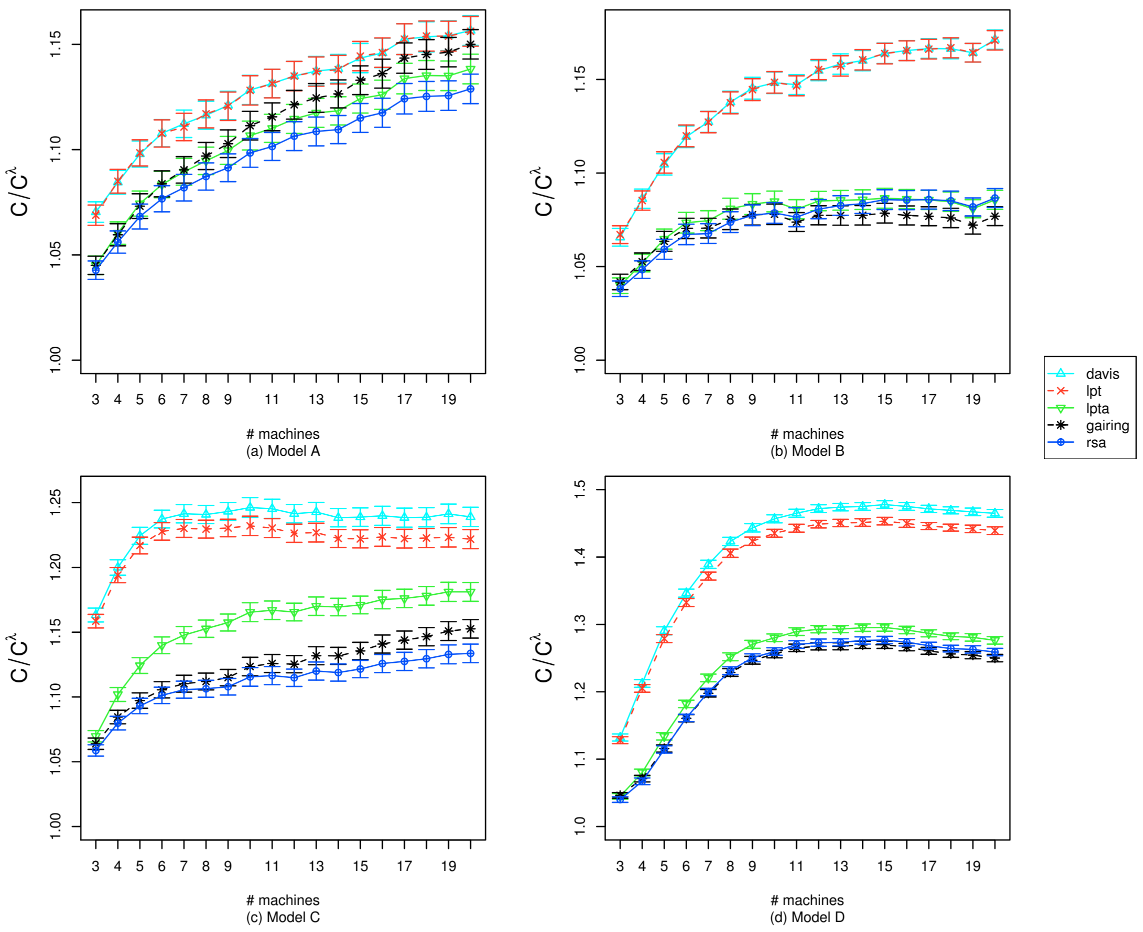

Fig. 1. Benchmark set computational results.

B and D most jobs have a low number of valid machines (1 or 2). This contributes to a worse solution of the traditional LPT heuristic, but also means that the distance between RSA and LPTA is less in these cases.

A. *Comparison of running times*

- RSA, LPT: $O(n \log n + S)$
- DAVIS: $O(mn \log n)$
- GAIRING: $O(mS \log W)$

RSA is faster than DAVIS and GAIRING. Because $S \in [n, nm]$, GAIRING can take up to $O(m^2 \frac{\log W}{\log n})$ longer. DAVIS can take up to $O(m \log n)$ times longer.

VI. Conclusions

Computational results show that our proposal can compete with the best algorithms. It is also very simple and the most efficient. For these reasons, we believe it justified to use it for the problem of restricted scheduling on identical machines, specially in applications where a low execution time is important.

References

[1] Y. Azar, J. S. Naor, and R. Rom, “The competitiveness of on-line assignments,” in *SODA ’92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, 1992, pp. 203–210.

[2] J. J. Galvez, P. M. Ruiz, and A. F. Gomez-Skarmeta, “A Feedback-based Adaptive Online Algorithm for Multi-Gateway Load-Balancing in Wireless Mesh Networks,” in *IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, 2010.

[3] M. Gairing, T. Lücking, M. Mavronicolas, and B. Monien, “Computing nash equilibria for scheduling on restricted parallel links,” in *STOC ’04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 2004, pp. 613–622.

[4] J. K. Lenstra, D. B. Shmoys, and E. Tardos, “Approximation algorithms for scheduling unrelated parallel machines,” *Math. Program.*, vol. 46, no. 3, pp. 259–271, 1990.

[5] O. H. Ibarra and C. E. Kim, “Heuristic algorithms for scheduling independent tasks on nonidentical processors,” *J. ACM*, vol. 24, no. 2, pp. 280–289, 1977.

[6] E. Davis and J. M. Jaffe, “Algorithms for scheduling tasks on unrelated processors,” *J. ACM*, vol. 28, no. 4, pp. 721–736, 1981.

[7] S. Martello, F. Soumis, and P. Toth, “Exact and approximation algorithms for makespan minimization on unrelated parallel machines,” *Discrete Appl. Math.*, vol. 75, no. 2, pp. 169–188, 1997.

[8] E. Mokotoff and P. Chretienne, “A cutting plane algorithm for the unrelated parallel machine scheduling problem,” *European Journal of Operational Research*, vol. 141, no. 3, pp. 515–525, 2002.

[9] M. Gairing, B. Monien, and A. Woclaw, “A faster combinatorial approximation algorithm for scheduling unrelated parallel machines,” in *ICALP ’05: Proc. of the 32nd International Colloquium on Automata, Languages and Programming*, 2005, pp. 828–839.

[10] R. Graham, E. Lawler, J. Lenstra, and A. R. Kan, “Optimization and approximation in deterministic sequencing and scheduling: A survey,” *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.