An Algorithm for Distributing Jobs in Claster Environment

G. Chelidze, B.Mamporia, Nodari Vakhania

Affiliations: Tbilisi State University, Niko Muskhelishvili Institute of Computational Mathematics of Georgian Technical University, Facultad de Ciencias, UAEMor Cuernavaca 62210, Mexico. E-mails g.chelidze@tsu.ge , badrimamporia@yahoo.com nodari@uaem.mx

Abstract. Cluster scheduling serial and parallel jobs need to be distributed among parallel CPUs. The corresponding combinatorial optimization problems which arise here are intractable (NP-hard). Hence one cannot expect to "solve" such problems optimally. Besides, the Objective criteria are often contradictory (there might be no single criterion). Here we propose a heuristic method that can be used for the distribution of jobs on parallel CPUs with the objective is to minimize CPU idle-time. Our algorithm works on non-identical CPUs when the speed of a CPU is job-dependent.

A cluster scheduler aims to distribute jobs in a fair manner while optimizing overall cluster efficiency, e.g., the idle CPU time. In CPU time sharing systems the system must decide which of the arriving jobs to assign to the processor and when. The jobs may arrive over time or, in case of the scheduled maintenance and other scheduled computer services (such as operating system updates), job arrival time and its (approximate) processing time are known in advance.

Here we deal with the latter kind of scenario, where our objective is to minimize the overall job

processing time, which also yields the minimal processor idle-time. We consider a general model rarely dealt with in cluster scheduling when CPUs are non-identical and their speed is job-dependent, i.e., a CPU may work fast or slow depending on the type of request assigned to it. In the literature such parallel processors are referred to as unrelated. In practice, non-identical CPUs may well form part of the same cluster when it is gradually upgraded with new faster CPUs (while the old slower ones are also kept).

The scheduling problems that arise in cluster environment are mostly NP-hard. Hence, one naturally thinks on an approximate solution method. In this paper, we propose a heuristic approximation algorithm for the solution of the unrelated CPU time sharing problem. Before we describe our generic problem formally, we give a brief overview of the related problems.

In general, multiprocessor scheduling problems deal with a group of parallel processors that can process a given set of jobs {1,2,…,n} under certain restrictions. A group of *identical* processors processes every job for the same time, whereas in the *uniform* processor environment, processor M_i has its own speed s_i (the same for all jobs).

hence the job processing times must be given individually for each processor. Scheduling unrelated parallel processors non-preemptively with the objective to minimize the *makespan*, that is, the maximum job completion time, commonly abbreviated as $R//C_{\max}$ (see Graham et al. [8]), is among the heaviest strongly NP-hard problems. The best known polynomial approximation algorithm for this problem has an asymptotic performance ratio 2 (the performance ratio of a schedule is the ratio of the objective function value of that schedule to the optimal objective function value). For many NP-hard scheduling problems, a strong restriction on the job processing times may convert the problem to a polynomially solvable one. For $R//C_{\max}$ however, the problem is known to remain NP-hard even for two allowable integer job processing times p and q, p<q, and q$\neq$ 2p, abbreviated $R/p_{ji}\in\{p,q\}/C_{\max}$ (see Lenstra et al.[12]). The authors in [12] have shown that the version with p=1 and q=2 can be polynomially solved using a reduction from a version of the assignment (matching) problem. Later Vakhania et al. [15] showed that a similar kind of reduction for times p and 2p is not possible, and have proposed another polynomial-time algorithm using linear programming thus showing that $R/p_{ji}\in\{p,2p\}/C_{\max}$ is the maximal polynomially solvable special case of $R//C_{\max}$ with restricted job processing times (note that for unrelated processors, at least two possible job processing times must be specified as otherwise we will be brought to the identical/uniform processor environment). The authors in [15] have also shown that the problem $R/p_{ji}\in\{p,q\}/C_{\max}$ can be solved in polynomial time with an absolute worst-case error of q. For a much simpler identical processor environment, the most scheduling problems still remain NP-hard. For instance, the problem of scheduling already two identical processors with the objective to minimize the makespan is NP-hard. However, the approximation in polynomial time for scheduling identical and also uniform processors can be done efficiently. An O(n\log n) MULTIFIT algorithm for the problem $R//C_{\max}$ gives a performance ratio of 13/11 for identical processors and of 7/5 for uniform processors (see Friesen [3]}, Yue [16], Friesen \& Langston [4]. There also exist polynomial approximation schemes for uniform processors, see Hochbaum and Shmoys [9]. Scheduling problems on identical and uniform processors with restricted job processing times have been also studied. Among such problems ones with unit and equal-length jobs are well studied (see, for example Kravchenko and Werner [11]). The first polynomial-time approximation algorithm for the problem $R//C_{\max}$ was proposed by Ibarra and Kim [10] with an unattractive performance ratio of m, and a better polynomial-time algorithm with a performance ratio within $2\sqrt{m}$ was suggested by Davis and Jaffe [1]. Potts [13] gave an approximation algorithm with the performance ratio 2, which is polynomial in n and exponential in m. One decade later (and more than two decades ago), Lenstra et al. [12] have developed the first polynomial-time algorithm (in both the number of jobs n and the number of processors m with the same performance ratio 2. The authors in [12] have also shown that no polynomial algorithm

with a performance ratio of 1.5 or less may exist for the problem $R//C_{\max}$ (unless P=NP). The algorithm is based on rounding of a fractional solution obtained by linear programming. Later the rounding approach was used in a better performance algorithm with the ratio 2-1/m Shchepin \& Vakhania[14], and it was shown that no rounding-based algorithm can give a better performance ratio than 2-1/m Ebenlendr et al.[2] have proposed a special case of scheduling on unrelated processors in which each job can be assigned to at most two processors. Their 1.75-approximation algorithm also applies the rounding of the fractional solution obtained by linear programming. Even for this restricted version, it remains NP-hard to find a better than a 1.5-approximation algorithm. As to the heuristic algorithms without a guaranteed performance for our problem $R//C_{\max}$, we just mention a few recent papers with interesting results. If any job can be processed on any processor, we refer e.g. to the works by Fanjul-Peyro and Ruiz [5],[6], and if jobs can be processed only on particular processors, we refer the reader to [7] and the works cited in these papers.

Let $J=(j_1, j_2, \cdots j_n)$ are jobs, $M_1, \cdots, M_m$ are processors. M_j has his own speed s_i (the same for all jobs). Assume, that the speed of the slowest processor M_1 is 1. Let s_i is ordered such that s_i is increased by i. Let the processing time of the job j by the processor M_i be p_{ij}. If the speed of the job j by the processor M_1 is p_{1j}, then $p_{ij} = p_{1j} \big/ s_i$. We have the matrix

$$\begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{pmatrix}, \qquad (1)$$

where $p_{i1} \le p_{i2} \le \cdots \le p_{in}; p_{1j} \ge p_{2j} \ge \cdots p_{mj}$ for all $i=1,2,\cdots,m$ and $j=1,2\cdots,n$. Let $\pi=(A_1, A_2, \cdots, A_m)$ is the partition of J, such that $A_i \ne \varnothing$, $A_k \cap A_l = \varnothing$, $\cup A_i = J$. Denote by $\Pi \equiv \{\pi=(A_1, A_2, \cdots A_m),\ A_i \ne \varnothing,\ A_k \cap A_l = \varnothing,\ \cup A_i = J\}$ the set of all such partitions of J. Let $S_i = \sum_{j_j \in A_i} p_{ij}$. Consider

$$\min_{\pi \in \Pi} \max_{A_k, A_l \in \pi} |S_k - S_l|. \qquad (2)$$

The job scheduling problem is to find the partition $\pi_0 = (A_1^0, A_2^0, \cdots A_m^0)$ such that

$$\max_{A_k^0, A_l^0 \in \pi} |S_k - S_l| = \min_{\pi \in \Pi} \max_{k,l, A_i \in \pi} |S_k - S_l|. \qquad (3)$$

Then, the optimal processing time of performance of the jobs $J=(j_1, j_2, \cdots j_n)$ is

$$T=\max_{i\le m}\sum_{j_l\in A_i^0} p_{il}\ .$$

It is obvious, that $\left[\sum_j p_{mj}\Big/ m\right]\le T\le\left[\sum_j p_{1j}\Big/ m\right]+1$.

We have also $\left[\sum_j p_{mj}\Big/ m\right]\le\left[\sum_j p_{(m-1)j}\Big/ m\right]\le....\le\left[\sum_j p_{2j}\Big/ m\right]\le\left[\sum_j p_{1j}\Big/ m\right]$.

Consider the set $\Pi\equiv\{\pi=(A_1,A_2,\cdots A_m),\ A_i\neq\varnothing,\ A_k\cap A_l=\varnothing,\ \cup A_i=J\ \}$. For the simplest case, when the sets $A_1,A_2\cdots A_{(m-1)}$ contain one element each of them and A_m contains $n-m+1$ elements, the quantity of such elements in Π are (n-m+1)!. That is, the quantity of elements of Π is huge ($\sum_{m_1+m_2+\cdots m_m=n} C_n^{m_1}C_{(n-m_1)}^{m_2}\cdots C_{(n-m_{1\cdots m_m})}^{m_m}$). Our goal in future development is, using the matrix (1), to construct the discrete probability measure P on Π, such that for $B\in\Pi$, $P(B)$ will be the probability of the event "B-contains the optimal partitions of Jobs $A_1^0,A_2^0,\cdots A_m^0$." Then we will seek the heuristic algorithm on "eventless" $B\in\Pi$ with probability near to 1. Another direction of our approach for the future development is the following: analyzing the matrix (1) we hope to receive the approximate value of T. Then using the Monte-Carlo method, simulate the process on the probability space Π to receive the partition of jobs $A_1^\varepsilon,A_2^\varepsilon,\cdots A_m^\varepsilon$ such that for $T_\varepsilon\equiv\max_{A_i^\phi}\sum_{j\in A_i^\varepsilon} p_{ij}$, the inequality $|T-T_\varepsilon|\le\varepsilon$ holds for sufficiently small ε.

At the end of our presentation we give the one simple heuristic algorithm to schedule jobs to the processors: consider the matrix (1). Let For the j_n job choose the processor M_m, for the j_{n-1} job choose the processor M_{m-1} and so forth, for the job j_{n-m} choose to the processor M_1. Consider the corresponding processing times $p_{mn},p_{(m-1)(n-1),\cdots},p_{1(n-m)}$. Let us ordered them in the increasing order and denote by $Q_1=(q_1^1,q_2^1,\cdots q_m^1)$ this schedule and the corresponding processing times on the processors denote by $S_{q_1^1}^{(1)},S_{q_2^1}^{(1)},\cdots,S_{q_m^1}^{(1)}.\ S_{q_1^1}^{(1)}\le S_{q_2^1}^{(1)}\le\cdots\le S_{q_m^1}^{(1)}$. For the job $j_{(n-m-1)}$ choose the processor $M_{q_1^1}$ and consider the corresponding processing times $S_{q_1^1}^{(1)}+p_{q_1^1(n-m-1)},S_{q_2^1}^{(1)},\cdots,S_{q_m^1}^{(1)}$. Let us again ordered them to the increasing order. Let $Q_2=(q_1^2,q_2^2,\cdots,q_m^2)$ be the this schedule and denote the processing times in such a way: $S_{q_1^2}^{(2)},S_{q_2^2}^{(2)},\cdots,S_{q_m^{21}}^{(2)}.\ S_{q_1^2}^{(2)}\le S_{q_2^{21}}^{(2)}\le\cdots\le S_{q_m^{21}}^{(2)}$. For the job $j_{(n-m-2)}$ choose the processor q_1^2. We have the following processing times on the processors: $S_{q_1^2}^{(2)}+p_{q_1^2(n-m-2)},S_{q_2^2}^{(2)},\cdots,S_{q_m^2}^{(2)}$. Let us again ordered them to the increased order. Let $Q_3=(q_1^3,q_2^3,\cdots,q_m^3)$ be the corresponding schedule and denote by

$S^{(3)}_{q_1^3}, S^{(3)}_{q_2^3}, \cdots, S^{(3)}_{q_m^3}$ the corresponding processing times on the processors. And so forth, for the job j_1 choose the processor q_1^{n-m} and at lust, we have the schedule $Q_{n-m} = (q_1^{n-m}, q_2^{n-m}, \cdots, q_m^{n-m})$ and the following processing times for the processors $S^{(n-m)}_{q_1^{n-m}} + p_{q_1^{n-m}1}, S^{(n-m)}_{q_2^{n-m}}, \cdots, S^{(n-m)}_{q_m^{n-m1}}$.

References

[1]. E. Davis and J. M. Jaffe. Algorithms for scheduling tasks on unrelated processors.

J. ACM 28, 721 - 736, 1981.

[2].T. Ebenlendr, M. Krï¿½\' al and J. Sgall. Graph Balancing: A Special Case of Scheduling

Unrelated Parallel processors. *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms,* 483-490, 2008.

[3]. D. K. Friesen. Tighter bound for the MULTIFIT processor scheduling algorithm *SIAM J. Comput.,* 13, 170-181, 1984

[4]. D. K. Friesen and M. A. Langston. Bounds for MULTIFIT

scheduling on uniform processors . *SIAM J. Comput*, 12, 60-70, 1983

[5]. L. Fanjul-Peyro and R.Ruiz. Size-reduction heuristics for the unrelated parallel processors scheduling problem. *Comp. Oper. Res.,* 38, 301 -- 309, 2011.

[6].L. Fanjul-Peyro and R.Ruiz. Iterated greedy local search methods for unrelated parallel processor scheduling. *Eur. J. Oper. Res*., 207, 55 -- 69, 2010.

[7].L. Fanjul-Peyro and R.Ruiz. Scheduling unrelated parallel processors with optional processors and jobs selection. *Comp. Oper. Res*., 39, 1745 -- 1753, 2012.

[8].R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan.

Optimization and approximation in deterministic sequencing and scheduling: A survey.

Ann. Discr. Math., 5: 287 -- 328, 1979.

[9].D. S. Hochbaum and D. B. Shmoys. A polynomial approximation

scheme for scheduling on uniform processors: using the dual approximation

approach", *SIAM J. Comput*. 17, 539-551, 1988

[10].O.H. Ibarra and C.E. Kim. Heuristic algorithms for scheduling independent tasks on non-identical processors. *J. ACM* 24: 280 -- 289, 1977.

[11].S.A. Kravchenko and F. Werner. Parallel processor problems with equal processing times: A survey. *J. of Sched.,* 14, 435 -- 444, 2011.

[12] J.K. Lenstra, D.B. Shmoys and E. Tardos. Approximation algorithms for scheduling unrelated parallel processors. *Math. Progr.,* 46, 259 - 271, 1990.

[13].C.N. Potts. Analysis of a linear programming heuristic for scheduling unrelated parallel processors. Discr. Appl. Math., 10, 155 - 164, 1985.

[14].E. Shchepin and N. Vakhania. An optimal rounding gives a better approximation for scheduling unrelated processors. *Oper. Res. Lett.,* 33, 127 - 133, 2005.

[15]N. Vakhania, J.Hernandez, F.Werner. Scheduling unrelated processors with two types of jobs.

International Journal of Production Research Vol. 52, No. 13, p. 3793 - 3801, 2014 (http://dx.doi.org/10.1080/00207543.2014.888789).

[16]M. Yue. On the exact upper bound for the multifit processor scheduling algorithm. *Ann. Oper. Res*., 24, 233-259, 1990.