ATLANTIS PRESS
Advances in Engineering Research (AER), volume 130
5th International Conference on Frontiers of Manufacturing Science and Measuring Technology (FMSMT 2017)

A Direct Proof of the 4/3 Bound of LPT Scheduling Rule

Xiao Xin
College of Foreign Studies, Shandong Institute of Business and Technology, Yantai, 264005, China
xinxiaoyt@hotmail.com

Keywords: Scheduling, Parallel Machines, Longest Processing Time First, Worst-case Analysis.

Abstract. The problem of scheduling independent jobs on identical parallel machines for minimizing makespan has been intensely studied in the literature. One of the most popular constructive algorithms for this problem is the LPT (Longest Processing Time First) rule whose approximation ratio has been proved by contradiction. A direct proof of its approximation ratio is presented, which can be regarded as an acquisition of knowledge by deductive means.

Introduction

Scheduling n independent jobs on m identical parallel machines with the objective of minimizing makespan (the maximum completion time of all n jobs) is a fundamental problem in combinatorial optimization. It has been intensely studied since the publication of the pioneering work of McNaughton [1]. There is a rich literature on this problem (see, e.g., [2-4]). This problem is routinely encountered in many real-life applications such as production lines, shipping docks, universities, and computer systems [5, 6].

Using the three-field notation of Graham et al. [7], the above problem is denoted as $P\|C_{\max}$, where P represents the identical parallel machines environment and $C_{\max}$ is the makespan objective function. Since the problem is strongly NP-hard [8], computation of complete enumeration of all possible schedules is prohibitively large. Therefore, approximation algorithms are frequently applied to obtain near-optimal schedules because of their polynomial time complexity, especially for the instances with large number of jobs. Approximation algorithms are usually evaluated by their approximation ratios. The *approximation ratio* of an algorithm for a minimization problem is the worst-case ratio on any input instance between the value of the solution obtained by the algorithm and the optimal solution value. An algorithm with approximation ratio ρ is called a ρ*-approximation algorithm.* A family of algorithms $\{A_\varepsilon\}$ is called a *polynomial time approximation scheme* (PTAS) if, for any arbitrarily small positive constant ε, A_ε is a $(1+\varepsilon)$-approximation algorithm running in time that is polynomial in the input size of the problem instance [9].

The most popular approximation algorithms for $P\|C_{\max}$ are List Scheduling [10], LPT (Longest Processing Time First) [11], MULTIFIT [12], COMBINE [13], and LISTFIT [14]. Laha and Behera [15] gave a comprehensive review and evaluation of these popular algorithms. There also exist polynomial time approximation schemes for $P\|C_{\max}$ [16, 17].

List Scheduling and LPT have approximation ratios $2-1/m$ and $4/3-1/(3m)$, respectively. They have the same time complexity, namely $O(n\log mn)$. MULTIFIT, COMBINE and LISTFIT have the same approximation ratio, namely $13/11+2^{-b}$, and their time complexities are $O(n\log n+bn\log m)$, $O(n\log n+bn\log m)$ and $O(n^2\log n+bn^2\log m)$, respectively. Clearly, MULTIFIT, COMBINE and LISTFIT provide better performance at the expense of higher time complexities. In many situations, LPT performs better and consumes the least computational time. In addition, LPT is easy to implementable.

Graham [11] proposed LPT and proved its approximation ratio by contradiction. The analysis presented in [11] is quite involved. Moreover, indirect proof has traditionally been criticized as showing merely 'that' its conclusion is true and not 'why' it is true. In this paper we present a

direct proof of the $4/3$ bound of LPT, which can be regarded as an acquisition of knowledge by deductive means.

The remainder of this paper is organized as follows. In Section 2, we formalize the problem of identical parallel machines scheduling, and then illustrate the LPT rule. In Section 3, we provide a worst-case analysis of the LPT rule and give a simple and direct proof of its $4/3$ bound. We conclude this paper in Section 4.

Preliminaries

For simplicity, we assume without loss of generality that the machines are indexed in non-decreasing order of their GoS levels such that $g(M_1) \le g(M_2) \le \cdots \le g(M_m)$. Let $g(M_{m+1}) = +\infty$. Let $\mathcal{J}_i = \{ j \in \mathcal{J} \mid g(M_i) \le g_j < g(M_{i+1}) \}$, $i = 1, 2, \ldots, m$. Certainly, we have $\mathcal{J} = \bigcup_{i=1}^{m} \mathcal{J}_i$. The jobs in $\mathcal{J}_i$ ($i = 1, 2, \ldots, m$) can be processed on any of the machines $M_1, M_2, \ldots, M_i$, but cannot be processed on any of the machines $M_{i+1}, M_{i+2}, \ldots, M_m$.

In this section we will formalize the problem of scheduling independent jobs on identical parallel machines for minimizing makespan, and illustrate the LPT rule proposed in [11].

As mentioned above, the problem is denoted as $P \| C_{\max}$. In this problem, there are n independent jobs to be processed on m identical parallel machines. Each machine can process at most one job at a time. Once a job starts processing on one machine, it must be completed on that machine without preemption. The ready times of all jobs are zero. Let p_j denote the processing time of job j (including any setup time required). Let J_i denote the subset of jobs assigned to machine M_i in a generated schedule. Let $C(J_i) = \sum_{j \in J_i} p_j$ denote the *load* of machine M_i in the schedule. The *makespan* of the schedule is defined to be $C_{\max} = \max_{1 \le i \le m} C(J_i)$. Thus, the schedule is determined by the subsets of jobs $J_1, J_2, \ldots, J_m$.

The well-known *longest processing time first* (LPT) rule [11] for $P \| C_{\max}$ works as follows. It first sorts all the jobs in non-increasing order of their processing times. Then, it schedules the jobs one by one in this order such that each job is assigned to the machine with currently smallest load (ties broken arbitrarily).

Worst-case analysis

In this section we will study the behavior of the LPT rule for $P \| C_{\max}$ and give a direct proof for its 4/3 bound.

Let OPT denote the makespan of an optimal schedule for $P \| C_{\max}$. All the jobs are classified as long, median and short jobs. Job j is called a *long job* if $p_j > 2OPT/3$, or a *median job* if $OPT/3 < p_j \le 2OPT/3$, or a *short job* if $p_j \le OPT/3$.

The LPT rule solves $P \| C_{\max}$ in the following three phases. In the first phase, it assigns m longest jobs each to an empty machine. If there are less than m jobs, then the algorithm terminates. In the second phase, it continues to assign the remaining jobs to the machines until the currently selected least loaded machine has to be a machine which has processed two jobs already (i.e., each machine having the least load has processed two jobs already), or there are no jobs left, whichever occurs first. In the third phase, it assigns the remaining jobs until there are no jobs left.

Let $C^1_{\max}, C^2_{\max}, C^3_{\max}$ denote the makespans after the three phases respectively. Let Σ denote the schedule generated by the LPT rule with makespan $C_{\max}$. Clearly, we have $C_{\max} = C^3_{\max}$. The following three lemmas analyze the behavior of the LPT rule.

Lemma 1. *When the first phase completes, all long jobs get assigned. Moreover,* $C^1_{\max} \le OPT$.

Proof. Note that the number of long jobs is at most m. When the first phase completes, each long job has been assigned to a different machine. Since each machine processes at most one long job, and the processing time of any job is less than or equal to OPT, it follows that $C_{\max}^{1} \le OPT$. □

Lemma 2. *When the second phase completes, all median jobs get assigned; and if a job has to be assigned to a machine which has processed a long job already, this job must be a short job. Moreover,* $C_{\max}^{2} \le 4OPT/3$.

Proof. Let n_L denote the number of long jobs. These n_L long jobs have to be assigned to n_L different machines in any optimal schedule. In any optimal schedule, any machine which has been assigned a long job already cannot be assigned a median job. If there are more than $2(m-n_L)$ median jobs, then any optimal schedule cannot finish all the jobs by time OPT. Hence, the number of the median jobs is at most $2(m-n_L)$. The second phase assigns all the median jobs to the $m-n_L$ machines which process no long jobs. Each of these $m-n_L$ machines processes at most two median jobs. Therefore, all median jobs can be assigned; and if a job has to be assigned to a machine which has processed a long job already, this job must be a short job. Combining the fact that each machine processes at most two jobs after the second phase, we get $C_{\max}^{2} \le 4OPT/3$. □

Lemma 3. *When the third phase completes, all short jobs get assigned. Moreover,* $C_{\max}^{3} \le 4OPT/3$.

Proof. It is obvious that all jobs can be assigned when the algorithm terminates. Let j denote the job which is completed last in Σ. Assume without loss of generality that job j is processed on machine M_i. If M_i processes exactly one job, then Σ is an optimal schedule. If M_i processes exactly two jobs, then by Lemma 2, they cannot be two long jobs. We have $C_{\max}^{3} \le 4OPT/3$. If M_i processes more than two jobs, than by Lemma 2, job j must be a short job. Before job j is assigned, the load on M_i is at most OPT. (Otherwise, all the jobs cannot be completed by time OPT in any feasible schedule.) It follows that $C_{\max}^{3} \le OPT + p_j \le 4OPT/3$.

Combining the above three lemmas, we get:

Theorem 1. *The LPT rule is a 4/3-approximation algorithm for* $P \| C_{\max}$.

Conclusion

In this paper, we considered the problem of scheduling independent jobs on identical parallel machines for minimizing makespan. We obtained a simple and direct proof for the 4/3 bound of the LPT rule. It would be interesting to conduct a rigorous analysis which can lead to a direct proof for the $4/3-1/(3m)$ bound of the LPT rule.

References

[1] R. Mcnaughton, Scheduling with deadlines and loss functions, Management Science. 6 (1959) 1-12.

[2] B. Chen, C. N. Potts and G. J. Woeginger, A review of machine scheduling: Complexity, algorithms and approximability, in: Handbook of combinatorial optimization, ed: Springer, 1998, pp. 1493-1641.

[3] J. Y. Leung, Handbook of scheduling: algorithms, models, and performance analysis, CRC Press, 2004.

[4] P. Brucker, Scheduling algorithms (fifth edition), Springer, 2007.

[5] E. Mokotoff, Parallel machine scheduling problems: a survey, Asia Pacific Journal of Operational Research. 18 (2001) 193-242.

[6] L. H. Su, C. Y. Lien, Scheduling parallel machines with resource-dependent processing times, International Journal of Production Economics. 117 (2009) 256-266.

[7] R. L. Graham, E. L. Lawler, J. K. Lenstra and A. R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Annals of discrete mathematics. 5 (1979) 287-326.

[8] E. L. Lawler, J. K. Lenstra, A. H. R. Kan and D. B. Shmoys, Sequencing and scheduling: Algorithms and complexity, Handbooks in operations research and management science. 4 (1993) 445-522.

[9] C. H. Papadimitriou, K. Steiglitz, Combinatorial optimization: algorithms and complexity, Courier Dover Publications, 1998.

[10] R. L. Graham, Bounds for certain multiprocessing anomalies, Bell System Technical Journal. 45 (1966) 1563-1581.

[11] R. L. Graham, Bounds on multiprocessing timing anomalies, SIAM Journal on Applied Mathematics. 17 (1969) 416-429.

[12] E. G. C. Jr, M. R. Garey and D. S. Johnson, An Application of Bin Packing to Multi-Processor Scheduling, SIAM Journal on Computing. 7 (1978) 1-17.

[13] C. Y. Lee, J. D. Massey, Multiprocessor scheduling: combining LPT and MULTIFIT, Discrete Applied Mathematics. 20 (1988) 233-242.

[14] J. N. D. Gupta, A. J. Ruiz-Torres, A LISTFIT heuristic for minimizing makespan on identical parallel machines, Production Planning & Control. 12 (2001) 28-36.

[15] D. Laha, D. K. Behera, A comprehensive review and evaluation of LPT, MULTIFIT, COMBINE and LISTFIT for scheduling identical parallel machines, International Journal of Information & Communication Technology, 2015.

[16] D. S. Hochbaum, D. B. Shmoys, Using dual approximation algorithms for scheduling problems theoretical and practical results, Journal of the ACM. 34 (1987) 144-162.

[17] N. Alon, Y. Azar, G. J. Woeginger and T. Yadid, Approximation schemes for scheduling on parallel machines, Journal of Scheduling. 1 (1998) 55-66.