Workload Balancing in Parallel Machines based on Semi-matching Theory

Yong Zhan

School of Mechanical and Electrical Engineering, Harbin Engineering University, China

zhanyong@hrbeu.edu.cn

Keywords: Parallel machine, Machine eligibility restrictions, Semi-matching, Load balancing.

Abstract. This paper addresses the problem of scheduling jobs on parallel machines with machine eligibility restrictions so as to balance the workload among machines. A weighted bipartite graph model G=[J ∪ M,E,W] is used to formulate the studied problem. We propose a two-phase algorithm to find the optimal solution. In phase 1,the initial solution construction algorithm is used to build an initial solution, followed by an improvement algorithm to improve the initial solution to optimal one in phase 2. Computational results are presented for a set of randomly generated instances.

Introduction

Parallel machines scheduling problem is important from both the theoretical and practical points of view. It is a generalization of the single machine scheduling problem. Meanwhile, many real manufacturing settings can be naturally formulated as parallel machines scheduling problems. Therefore, parallel machines scheduling problems have been studied extensively in the literature.

For general problems with machine eligibility restrictions, some two-phase algorithms are developed. Paper [1] presents a two-phase algorithm for the unrelated parallel machines scheduling problem. In phase 1, constructive heuristics are used to build an initial solution, followed by an improvement heuristic to improve the initial solution in phase 2. Paper [2] deals with the problem with time windows and eligibility constraints. They develop a constraint-graph-based construction algorithm for generating near-optimal solutions, then use genetic algorithm to enhance the near-optimal solutions. For more detailed information about parallel machines scheduling problem with machine eligibility restrictions, the reader is referred to the survey papers [3,4].

The purpose of this paper is to develop an exact algorithm based on semi-matching theory.The rest of this paper is organized as follows: in Section 2, we give the formal introduction of the studied problem. Next, a weighted bipartite graph model is presented in Section 3. In Section 4 and Section 5, we describe the proposed two-phase algorithm. In Section 6, the computational results are presented. Finally, Section 7 concludes the paper.

Problem description

The problem considered in this paper can be formally described as follows: a set $J=\{J_1, J_2,\ldots, J_n\}$ of n independent jobs are scheduled on m identical parallel machines $M=\{M_1, M_2,\ldots, M_m\}$. Each job has to be processed by exactly one machine. The processing time of job J_i is denoted by T_i. In identical parallel machine scheduling problem, each job has the same processing time no matter the machine to which it is assigned. Machine eligibility restrictions are considered in this paper. That is, not all of the m parallel machines are capable of processing each job. For each job, it is eligible to be processed on certain machines only.

We select the minimization of the maximum completion time or makespan(C_{max}) as the optimization criterion. The makespan is determined by the maximum workload among parallel machines, where the workload of a machine is the sum of processing times of all jobs assigned to it. The following notations and definitions are used to define the studied problem.

Adjacency matrix $A_{n\times m}$: the machine eligibility restrictions can be represented by an adjacency matrix $A_{n\times m}$, where entry A_{ij} in row i and column j is

$$A_{ij} = \begin{cases} 1 & \text{if job } J_i \text{ can be processed by machine } M_j \\ 0 & \text{otherwise} \end{cases}$$

C_j: the workload of machine M_j.

C_{max}: the maximum workload in a feasible schedule.

x_{ij}: a binary variable which takes value 1 if J_i is assigned to M_j and 0 otherwise.

A straightforward Mixed Integer Linear Programming(MILP) formulation for the studied problem is as follows:

$$\min C_{max} \tag{1}$$

St:

$$C_{max} \geq C_j \qquad \forall j \tag{2}$$

$$C_j = \sum_{i=1}^{n} x_{ij} p_i A_{ij} \qquad \forall j \tag{3}$$

$$\sum_{j=1}^{m} x_{ij} A_{ij} = 1 \qquad \forall i \tag{4}$$

$$x_{ij} \in \{0,1\} \qquad \forall i, j \tag{5}$$

Weighted Bipartite Graph Model for the Studied Problem

Instances of the studied problem can be modeled over a weighted bipartite graph $G = [J \cup M, E, W]$. Figure 1(a) shows the weighted bipartite graph model for the instance with five jobs and four machines. In Figure 1(a), the numbers in vertices of set J are job indices, and the processing time of each job is noted next to the edge.

A semi-matching F in $G = [J \cup M, E, W]$ is defined to be a set of edges, $F \subseteq E$, such that each vertex in J is an endpoint of exactly one edge in F.An edge e is called a matched edge if e belongs to F; otherwise, it is an unmatched edge.The semi-matching showed in Figure 1(b) gives an assignment for the instance, where J_1 is assigned to M_1, J_2 and J_3 are assigned to M_3, J_4 is assigned to M_4, and J_5 is assigned to M_2.

For more detailed information about semi-matching theory, the reader is referred to the papers [5, 6].

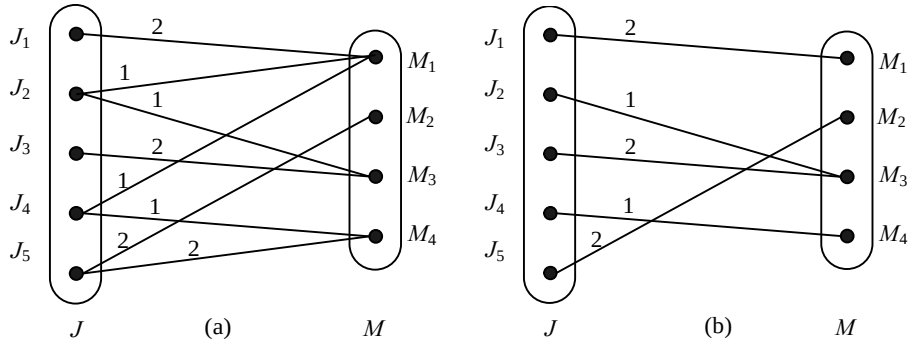

Fig. 1 Bipartite graph model for the instance with five jobs and four machines

Initial Solution Construction Algorithm

The initial solution construction algorithm has two modules: job sequencing and machine selection. Job sequencing module determines the arrangement of the jobs that is to be assigned. Machine selection module finds which machine is used to process a given job. The initial solution construction algorithm is described as follows.

Input: $G = [J \cup M, E, W]$.

Output: initial solution F_i

Step1: unfinished job set $J_w \leftarrow J$, $(F_i)_{n\times m} \leftarrow 0$.

Step2: if $J_w = \varnothing$, output F_i; otherwise, select a job J_x from J_w based on Shortest Processing First rule.

Step3: select a machine M_y in U_x with the minimum workload.

Step4: $(F_i)_{x,y} \leftarrow 1$.

Step5: delete J_x from J_w, go to Step2.

Initial Solution Improvement Algorithm

An improving solution is searched by modifying assignment partly and this is repeated until the optimal solution is reached. In the framework of weighted bipartite graph model, improvement algorithm based on optimal cost-reducing path search is proposed in this paper.

Alternating path is defined as a sequence of edges that are alternately contained in a semi-matching F and not contained in F. An alternating path is called a simple alternating path if no edge is repeated in the path. The notation $F \oplus P$ denotes the symmetric difference of semi-matching F and alternating path P; that is, $F \oplus P = (F\text{-}P) \cup (P\text{-}F)$. we propose the basic initial solution improvement algorithm as follows.

Input: a weighted bipartite graph $G = [J \cup M, E, W]$ and the initial semi-matching F_i.

Output: an optimal semi-matching F_o minimizing the maximum workload.

Step 1. Set a set of edges $F_o \leftarrow F_i$.

Step 2. Select the machine vertex M_x with the maximum workload in F_o.

Step 3. Build an alternating search tree T rooted at M_x.

Step 4. Find the optimal cost-reducing path P_o in the tree T, if P_o exists, go to Step 5. Otherwise, output F_o.

Step 5. $F_o \leftarrow F_o \oplus P_o$, go to Step 2.

Computational Results

The two-phase algorithm presented in this paper is coded in Visual Studio 2010 C# and implemented on a i7@2.60GHz personal computer with 4G memory. The central processing unit (CPU) time limit is set to 60 minutes.

We use the workload deviation to evaluate the quality of the initial solution. If the maximum workload and the minimum workload are denoted by C_{max} and C_{min} respectively, then the workload deviation Z is calculated as

$$Z = \frac{C_{max} - C_{min}}{C_{max}} \times 100\% \tag{6}$$

We use the following additional notations in our experiments:

Z_s: the workload deviation of the initial solution generated by SPT rule.

T_s: the CPU time required to improve the initial solution generated by SPT rule.

Table 1 reports the averages results of the computational experiments. All CPU times are in seconds. As can be seen from Table 1, the solution times increase considerably with increase in the number of jobs and number of machines. This is due to the fact that the increase of n and m enlarge the size of the alternating search tree. In the worst case, the proposed algorithm can handle 8×20 size problem in reasonable time, which is smaller than 60 minutes.

Conclusions

In this study, the workload balancing problem in parallel machines with eligibility restrictions is studied. We develope a two-phase exact approach based on a semi-matching model. The proposed algorithm builds an initial solution by applying greedy heuristics, then improves it to the optimal

solution based on alternating path search method. A computational experiment is designed for evaluating the performance of the proposed algorithm. The results reveal that it can handle 8×20 size problem in reasonable time, which is smaller than 60 minutes.

Table 1. Results of computational experiments

m	n	Z_r	T_r	F_i	F_o
2	10	0.083	0.001	268	250
2	20	0.144	0.003	652	606
4	10	0.104	0.004	188	144
4	20	0.014	22.363	224	211
6	10	0.626	0.243	126	99
6	20	0.052	1162	227	190
8	10	0.196	29.143	121	86
8	20	0.196	3262	184	151

Acknowledgements

This work is financially supported by the Research Fund for the Doctoral Program of Higher Education, China(20132304120021).

References

[1] A. Al Salem, R. L. Armacost: Qatar University (2002)

[2] D. T. Eliiyi, A. G. Korkmaz, A. E. Çiçk: Technological and Economic Development of Economy Vol.15(2) (2009), p.245

[3] J. Y. T. Leung, C. L. Li: International Journal of Production Economics Vol. 116(2) (2008), p.251

[4] L. W. Liao, G. J. Sheen: European Journal of Operational Research Vol.184(2) (2008), p.458

[5] N. Harvey, R. Ladner, L. Lovasz, T. Tamir: Journal of Algorithms Vol.59(1) (2006), p.53

[6] C. P. Low: Lecture Notes in Computer Science Vol. 3959 (2006), p.159-170