Journal of Engineering Mechanics and Machinery (2016) 1: 28-33
Clausius Scientific Press, Canada

Approximation Algorithm for Scheduling Parallel Machines with Machine Eligibility Restrictions and special jobs

Zhan Yong[a], Zhong Yuguang[b]

College of Mechanical and Electrical, Harbin Engineering University, Harbin, 150001, China
[a]zhanyong@hrbeu.edu.cn, [b]zhongyuguang@hrbeu.edu.cn

Keywords: parallel machine, machine eligibility restrictions, approximation algorithm

Abstract: This paper addresses the scheduling problem of parallel machines with machine eligibility restrictions and special jobs with the objective of minimizing the makespan. Each job can only be assigned to a specific subset of the machines. And the processing times of jobs are restricted to one of two values, 1 andε. A semi-matching model G=[$J\cup M$,E,W] is presented to formulate this scheduling problem. We propose an approximation algorithm, which is composed of two steps, that is, initial solution construction and initial solution improvement. The initial solution construction algorithm is developed to build a feasible solution by performing a simple greedy heuristic method. The initial solution is used as a starting point by the improvement algorithm. The main idea of the improvement algorithm is to construct alternating tree, then to find the optimal alternating path for each vertex in M iteratively. In order to improve efficiency, the length of each path in alternating tree is limited to 4 at most.

1. Introduction

We consider a special case of parallel machine scheduling problem with machine eligibility restrictions and special jobs. Each job can only be assigned to a specific subset of the machines. And the processing times of jobs are restricted to one of two values, 1 and ε. The objective is to minimize the makespan.

Most parallel machine scheduling problems are NP-hard. Due to the problem complexity, the development of approximation algorithms has received increasing attention in recent years. DAMODARAN[1] addresses parallel batch processing machines with unequal job ready times, and a simulated annealing algorithm is presented. WANG[2] presents a hybrid differential evolution algorithm for parallel machine scheduling with splitting jobs to minimizing the makespan. TAN[3] presents a SPT (shortest processing time) algorithm for two parallel machines scheduling problem with given unavailable periods. Whenever a machine becomes idle, the SPT algorithm assigns the job with shortest processing time on that machine. TAN prove that the SPT algorithm is 3/2-approximate. LIU[4] deals with a parallel machines scheduling problem with linear increasing processing time, and propose a LS (list scheduling) algorithm. Liu prove that the LS algorithm is $(1+b_{max})^{\frac{m-1}{m}}$-approximate, where b_{max} is the maximum deteriorating rate of job.

In this paper, a semi-matching model G=[$J\cup M$,E,W] is presented to formulate the studied problem into an optimal semi-matching searching problem. We propose an approximation algorithm, which is composed of two steps, that is, initial solution construction and initial solution improvement.

The remainder of this paper is organized as follows. In section 2, a formal definition and the semi-matching model for the scheduling problem under studied are presented. In section 3, the initial solution construction algorithm based on heuristics is described in detail. The improvement algorithm is presented in section 4. This paper concludes with section 5.

2. Problem Definition and Semi-matching Model

The problem studied in this paper can be formally described as follows. We are given m (m≥1) parallel machines $M = \{ M_1, M_2, M_3,......, M_m\}$ and n jobs $J = \{ J_1, J_2, J_3,......, J_n\}$. Machine eligibility restrictions are considered, which imply that any job J_n can only be assigned to one of the machine in a certain subset of M. And the processing times of jobs are restricted to one of two values, 1 and ε. We select the minimization of makespan as the optimization criterion.

In this paper, the semi-matching theory is adopt to model the studied problem. Let $G = [J\cup M, E, W]$ be a weighted bipartite graph, where J and M are two disjoint sets of vertices, and $E \subseteq J \times M$ is a set of edges. For convenience, a vertex of J and a vertex of M are called J-vertex and M-vertex respectively. Each J-vertex represents a job, and each M-vertex represents a machine. If an edge $e_{ij} \in E$ connects $J_i \in J$ to $M_j \in M$, whose weight is $W(e_{ij})$, it means that M_j can process J_i, and the weight of edge e_{ij} represents the time needed by M_j to finish J_i.

A set $F \subseteq E$ is a semi-matching if each J-vertex is incident with exact one edge in F[5]. Obviously, a semi-matching F corresponds to a assignment of jobs to machines. If F is a load balanced semi-matching, we can minimize the completion time. As shown in Fig.1(a), we represent the 3 parallel machines and 5 jobs as the vertices of a weighted bipartite graph. Fig. 1(b) is a semi-matching of the graph shown in Fig. 1(a).

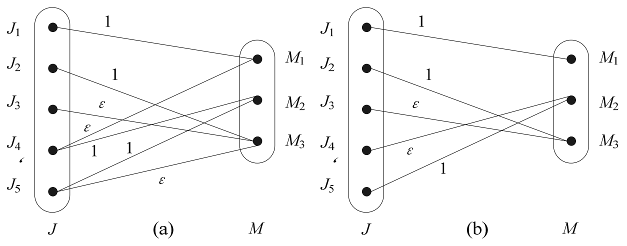

Fig.1. A example of a weighted bipartite graph and one of its semi-matchings

Given a edge $e_{ij} \in F$, let J_i and M_j be two endpoints of e_{ij}. We say that e_{ij} is a matched edge, J_i and M_j are matched vertices with respect to F, and J_i is a neighbor of M_j, respectively.Let $N(J_i)$ denote the set of neighbors of vertex $J_i \in J$. Similarly, $N(M_j)$ is defined for a vertex $M_j \in M$. Given a semi-matching F, the load on vertex $M_j \in M$ is defined as the sum of weights of the edges that are incident with M_j, denoted by $\delta_F(M_j)$:

$$\delta_F(M_j) = \sum_{e_{ij}\in F} W(e_{ij}) \tag{1}$$

The goal of this paper is to find a semi-matching with balanced load on M-vertices. The objective used in this paper is to minimize the maximum load of V-vertices, which can be expressed as

$$\min(\max\{\delta_F(M_j)\}) \tag{2}$$

3. Initial Solution Construction Algorithm based on Heuristics

Here, we present our initial solution construction algorithm based on heuristics.

```
**Algorithm 1.** initial solution construction algorithm
**Input**: weight bipartite graph $G = [J \cup M, E, W]$.
**Output**: initial solution $F_i$.
Step 1. Initialization: unfinished job set $J_w \leftarrow J$, $F_i \leftarrow \phi$.
Step 2. If $J_w \leftarrow \phi$, output $F_i$; otherwise, randomly select a currently available job $J_x$ from $J_w$.
Step 3. Select the eligibility machine $M_y$ of $J_x$ with the minimum workload.
Step 4. $(F_i)_{xy} \leftarrow 1$.
Step 5. Delete $J_x$ from $J_w$, go to Step 2.
```

4. Improvement Algorithm based on Alternating Path with Limited Length

For a given semi-matching F in G, edge $e_{ij} \in F$ and $e_{ij} \notin F$ are called matched edge and unmatched edge, respectively. F-alternating path is a simple path that its edges are alternately matched and unmatched.

Let P be a F-alternating path with respect to a semi-matching F, and the edges in P are denoted by $E(P)$. The notation $F \oplus E(P)$ is defined as the symmetric difference of sets F and $E(P)$; i.e. $F_n = F \oplus E(P) = (F - E(P)) \cup (E(P) - F)$. If the length of P is even, then F_n is a new semi-matching by switching matched and unmatched edges along P. After the operation of symmetric difference, the number of edges in semi-matching does not increase or decrease, but the load distribution among M-vertices changes. F-alternating path P is called a cost-reducing path, if $\max\{\delta_{F_n}(M_j)\} \le \max\{\delta_F(M_j)\}$.

In all alternating paths starting from M_j and all related semi-matchings, the optimal semi-matching F_o is a semi-matching with the smallest $\max\{\delta_{F_o}(M_j)\}$, and the corresponding alternating path is called the optimal alternating path.

The algorithm based on alternating path is simple to implement, but it is needed to search the whole alternating tree rooted at each vertex in M to find the optimal alternation path. In theory, this kind of algorithm can yield the optimal solution. But for the large-sized problem, the running time of traversing the whole alternating tree is too costly. In the study of bipartite graph matching, it is a practical method to restrict the length of the path to reduce the size of the searching tree[6]. Therefore, we follow this principle, and our improvement algorithm only use short alternating paths and circles which contain at most 2 edges in semi-matching to decrease the maximal load of F_i.

To ensure that the new semi-matching produced by the improvement algorithm is legal, all legal short alternating paths and circles are shown in Fig. 2. For convenience, the short alternating path and circle are all called short path.

The main idea of the improvement algorithm is to start with a initial solution and to decrease the maximal load of the semi-matching by a local improvement, which searches the optimal short path for every vertex $M_j \in M$ iteratively. In every iteration, we first build a alternating tree by BFS discipline to find out the optimal short path, and then perform the symmetric difference operation to produce a new semi-matching with smaller maximal load. The detail of the improvement algorithm is as follows.

```
**Algorithm 2.** initial solution improvement algorithm
**Input**: initial solution $F_i$.
**Output**: near-optimal solution $F_a$.
Step 1. initialization: $F_a \leftarrow F_i$, $M_t \leftarrow M$.
```

Step 2. select a vertex $M_j \in M_t$, build the alternating tree T rooted at M_j, where edges in F_i are directed from *M-vertex* to *J-vertex* and edges not in F_i are directed from *J-vertex* to *M-vertex*.

Step 3. find the optimal short path P_{M_j}.

Step 4. $F_a \leftarrow F_a \oplus E(P_{M_j})$.

Step 5. delete M_j from M_t, and if $M_t \neq \phi$, goto Step 2. Otherwise, output F_a.

Lemma 1. Let F_o be the optimal semi-matching of $G = [U \cup V, E, W]$, and F_a be a near-optimal semi-matching yielded by our algorithm. The maximal loads of F_o and F_a are denoted by $\max(F_o)$ and $\max(F_a)$, respectively. For F_a, if there is no short path that can decrease $\max(F_a)$, then:

$$\max(F_a) \leqslant (1+\varepsilon)\max(F_o) \tag{3}$$

Proof. Let $E_a = F_a - F_o$ and $E_o = F_o - F_a$ be two sets of edges. Obviously, we can build short paths by selecting edges from E_a and E_o alternately, and F_a can be improved to F_o by the operation of the symmetric difference between these short paths and F_a.

Let P be a short path , which consists four edges selecting from E_a and E_o alternately, as shown in Fig. 3. The four edges of P are denoted by e_i、e_j、e_k and e_l, and the edges of e_i and e_k belong to E_a, the other two edges of e_j and e_l belong to E_o . The M-vertices in P are denoted by M_i, M_j and M_k, and let M_j be the vertex with the maximal load in F_a and F_o. $\delta_a(M_j)$ and $\delta_o(M_j)$ represent the load on M_j in F_a and F_o, respectively. Other notations used here include:

E_{aj}: the set of edges connecting M_j in F_a.

E_{oj}: the set of edges connecting M_j in F_o.

E_{cj}: intersection set of E_{aj} and E_{oj}.

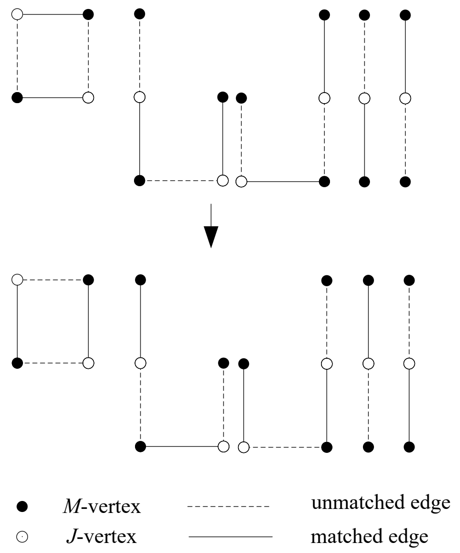

Fig.2. Legal short paths

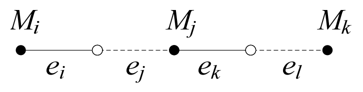

Fig.3. Short path P

If P is not an optimal short path, the reason for this must be that the load on M_k after switching operation exceeds $\delta_a(M_j)$. Therefore, Eq.4 must hold.

$W(e_k) - W(e_j) \le W(e_l)$ (4)

$W(e_k) \le W(e_l) + W(e_j)$ (5)

Let $\varepsilon = \frac{W(e_l)}{W(e_j)}$, then Eq.(13) is converted into Eq.6.

$W(e_k) \le (1+\varepsilon) W(e_j)$ (6)

In F_a, the load on M_j can be defined as $\delta_a(M_j) = \sum_{e \in E_{aj}} W(e)$.

And because:

$$\begin{aligned}
\sum_{e \in E_{aj}} W(e) &= \sum_{e \in E_{cj}} W(e) + \sum_{e \in (E_{aj} - E_{oj})} W(e) \\
&= \sum_{e \in E_{cj}} W(e) + \sum_{e_k \in (E_{aj} - E_{oj})} W(e_k) \\
&\le \sum_{e \in E_{cj}} W(e) + (1+\varepsilon) \sum_{e_j \in (E_{oj} - E_{aj})} W(e_j) \\
&< (1+\varepsilon)\{\sum_{e \in E_{cj}} W(e) + \sum_{e_j \in (E_{oj} - E_{aj})} W(e_j)\} \\
&= (1+\varepsilon) \sum_{e \in E_{oj}} W(e) \\
&= (1+\varepsilon)\delta_o(v_j)
\end{aligned}$$

In conclusion,

$\delta_a(v_j) \leqslant (1+\varepsilon)\ \delta_o(v_j)$ (7)

Thus,

$\max(F_a) \leqslant (1+\varepsilon) \max(F_o)$ (8)

From Eq.8, we can know that the worst-case ratio of the improvement algorithm is $1+\varepsilon$.

5. Summary

We have developed an approximation algorithm for the parallel machines scheduling problem with machine eligibility restrictions and special jobs, which is composed of 2 steps of initial solution construction and initial solution improvement. The initial solution is generated by a heuristic algorithm. In the improvement algorithm, we restrict the length of alternating path to 4 in order to improve the efficiency. And we prove that the worst-case ratio of the improvement algorithm is $1+\varepsilon$.

Acknowledgements

This work is financially supported by the Research Fund for the Doctoral Program of Higher Education, China(20132304120021); and by the National Natural Science Foundation of China (51275104).

References

[1] PURUSHOTHAMAN D, MARIO C V: Expert Systems with Applications Vol. 39(1) (2012), p.1451

[2] WANG W L, WANG H Y, ZHAO Y W , ZHANG L P, XU X L: Computers & Operations Research Vol.40(5)

(2013), p.1196

[3] TAN Z Y, CHEN Y, ZHANG A: International Journal of Production Economics Vol.146(1) (2013), p.293

[4] LIU M, ZHENG F F, WANG S J, XU Y F: Theoretical Computer Science Vol.497(29) (2013), p.108

[5] LOW C P: Information Processing LettersVol.100(4) (2006), p.154

[6] DORATHA E D, STEFAN H: Lecture Notes in Computer Science Vol.2647 (2003), p.107