Theoretical Computer Science 460 (2012) 10–15

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Combination of parallel machine scheduling and vertex cover

Zhenbo Wang *, Zhenhua Cui

Department of Mathematical Sciences, Tsinghua University, Beijing, 100084, China

ARTICLE INFO

Article history:
Received 25 November 2011
Received in revised form 18 May 2012
Accepted 2 June 2012
Communicated by D.-Z. Du

Keywords:
Approximation algorithm
Combination of optimizations
Local ratio
LPT
Scheduling
Vertex cover

ABSTRACT

This paper studies a combination of parallel machine scheduling and the vertex cover problem. Given some weighted vertices in an undirected graph, a set of vertices is called a vertex cover if for each edge at least one endpoint belongs to this set. Our problem is to schedule a set of weighted vertices on m identical parallel machines such that the set of vertices is a vertex cover and the makespan is minimized. We develop an approximation algorithm based on the local ratio method and the LPT rule, and prove that it is a $(3-\frac{2}{m+1})$-*approximation* algorithm.

1. Introduction

Parallel machine scheduling and the vertex cover problem are two fundamental problems in the fields of combinatorial optimization and theoretical computer science. They have been intensively studied because of both their theoretical interest and their wide applicability. Thousands of papers were devoted to the study of computational complexity, algorithms and applications for the two problems and their extensions. This paper considers a combination of parallel machine scheduling and the vertex cover problem. We will briefly introduce some related results on the two problems, and the readers are referred to [10,19,16] for comprehensive reviews.

The objective of parallel machine scheduling considered in this paper is to minimize the makespan, i.e. the completion time of a schedule. Both parallel machine scheduling and the vertex cover problem are NP-hard [13]. Therefore, many studies focused on designing approximation algorithms and analyzing their worst-case performance. For a minimization problem, a polynomial time algorithm is called a ρ-*approximation* algorithm for some $\rho > 1$, if it delivers a solution that is at most ρ times the optimum for every instance. The algorithm is also said to be ρ-*approximate*.

In 1966, Graham [14] presented the LS (list scheduling) algorithm for parallel machine scheduling to minimize the makespan. Whenever a machine becomes idle, the LS algorithm assigns the first available job on that machine. Graham proved that the LS algorithm is $(2-\frac{1}{m})$-*approximate*, probably the first time the worst-case performance of an approximation algorithm was analyzed. In the LS algorithm, if the jobs are sorted in order of nondecreasing processing times, then the algorithm is known as the LPT (longest processing time) rule. Graham proved that the LPT rule is $(\frac{4}{3}-\frac{1}{3m})$-*approximate* [15]. It is also known that PTAS exists for parallel machine scheduling [17] and FPTAS exists if the number of machines is fixed [20].

The vertex cover problem is one of the six basic NP-complete problems of Garey and Johnson [13]. Dinur and Safra [11] proved that the vertex cover problem can not be approximated within a factor of 1.36 unless $P = NP$. Moreover, based on the unique 2-prover-1-round game conjecture, Khot and Regev [18] showed that the vertex cover problem is hard to

* Corresponding author. Tel.: +86 10 62772796; fax: +86 10 62787945.
E-mail address: zwang@math.tsinghua.edu.cn (Z. Wang).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.06.003

be approximated within any constant factor better than 2. On the other hand, a 2-*approximation* algorithm can be easily obtained by the prime-dual approach [21] or the local ratio method [3]. This paper is mainly concerned with the local ratio method, and will describe the 2-*approximation* algorithm in the next section. The local ratio method was first presented by Bar-Yehuda and Even [4], and can be applied to obtain a $(2 - \frac{\log\log n}{2\log n})$-*approximation* algorithm for vertex cover problem [5]. Since then, the local ratio method was applied to a broad range of optimization problems [1,2,6–9,12], and the method itself has been developed as a unified framework for approximation algorithms [3].

This paper studies a combination of parallel machine scheduling and the vertex cover problem. Given some weighted vertices in an undirected graph, schedule a set of vertices on m identical parallel machines such that the set of vertices is a vertex cover and the makespan is minimized. This problem is inspired by such a scenario: we are going to manufacture devices to monitor a given network in which clients are nodes and connections are edges. Each connection needs to be monitored by assigning a device on one of the clients of this connection. The devices will be produced on m parallel machines. How do we minimize the completion time of devices such that they can be fixed on some of the clients to cover the network? To the best of our knowledge, there is little research addressing this problem. Beyond of theoretical interest of combining two important combinatorial optimization problems, it is not trivial from the viewpoint of technique. Even if we can optimally solve parallel machine scheduling and vertex cover problem respectively, it does not guarantee to obtain a good solution for the combination problem. For example, given m parallel machines, $m + 1$ jobs $\{J_1, J_2, \ldots, J_{m+1}\}$ with processing times $\{m, 1, \ldots, 1\}$ and a simple network with m edges such that J_1 connects to each of the other jobs, we can find that $\{J_1\}$ is an optimal solution for the vertex cover problem, and then we can optimally schedule it on one of machines, and the makespan is m. Obviously, the optimal solution for the combination problem is to assign each of $\{J_2, \ldots, J_{m+1}\}$ on one machine respectively, and the optimal makespan is 1. To obtain a satisfactory result for the combination problem, we need integrate parallel machine scheduling, the vertex cover problem and their corresponding algorithms.

The rest of the paper is organized as follows. In Section 2, we formally introduce the problem and some related results. We present the approximation algorithm and analyze its performance in Section 3, and some concluding remarks are provided in Section 4.

2. Preliminaries

2.1. The problem

The problem considered in this paper can be formally described as follows. Given m ($m \geq 1$) identical parallel machines $M = \{M_1, \ldots, M_m\}$, n jobs $J = \{J_1, \ldots, J_n\}$ with processing times $P = \{p_1, \ldots, p_n\}$ and an undirected graph $G = (V, E, W)$ with vertex set $V = J$, edge set E, and weight set of vertices $W = P$, find a schedule to minimize the makespan such that if G contains an edge between J_i and J_k, at least one of J_i and J_k should be processed. We denote this problem by $P_m|\text{vertex cover}|C_{max}$. Obviously, the problem is NP-hard and is hard to be approximated within any constant factor better than 2 even for $m = 1$ because of the hardness of the vertex cover problem.

This paper will develop an approximation algorithm mainly based on the LPT rule and the local ratio method. We will include their brief proofs since the proofs of our result build directly upon them.

2.2. LPT rule for parallel machine scheduling

The LPT rules can be described as follows:

Algorithm 1 LPT rule

1: sort the jobs in order of non-increasing processing times,
2: whenever a machine becomes idle, assign the first available job on it.

The following analysis comes from Graham [15]. Without loss of generality, we may assume that schedule starts at time zero, and the jobs are ordered in a list such that $p_1 \geq p_2 \geq \cdots \geq p_n$. Let J_k be the latest completed job, and y be the time point that J_k starts its processing. Without loss of generality, it can be assumed that J_k is the last job on the list. Let C^*_{max} and C_{max} be the makespans derived by an optimal algorithm and the LPT rule respectively.

It is easy to see that $C_{max} = p_k + y$, $y \leq \frac{1}{m}(\sum_{i=1}^{n} p_i - p_k)$ and $C^*_{max} \geq \frac{1}{m}\sum_{i=1}^{n} p_i$. We have

$$\begin{aligned}\frac{C_{max}}{C^*_{max}} &= \frac{p_k + y}{C^*_{max}} \\ &\leq \frac{\frac{1}{m}\sum_{i=1}^{n} p_i + \left(1 - \frac{1}{m}\right) p_k}{C^*_{max}} \\ &\leq 1 + \left(1 - \frac{1}{m}\right)\frac{p_k}{C^*_{max}}.\end{aligned}$$

If $p_k \leq \frac{C^*_{max}}{3}$, we have $\frac{C_{max}}{C^*_{max}} \leq \frac{4}{3} - \frac{1}{3m}$; otherwise, in an optimal schedule at most two jobs can be processed on each machine, and it is not hard to see that the LPT rule will result in an optimal schedule. Therefore, the LPT rule is $(\frac{4}{3} - \frac{1}{3m})$-*approximate*.

2.3. Local ratio method for vertex cover

The local ratio method is an approximation paradigm for NP-hard problems. It is remarkably simple and elegant. It cuts the combinatorial structure of the problem into several pieces which may be observed straightly, and the following local ratio theorem guarantees its performance. The theorem is concerned with a minimization problem: given a weight vector $w \in R^n$, and a set of feasibility constraints C, find a solution vector x satisfying the constraints in C that minimizes the scalar product $w \cdot x$.

Theorem 1 (*Local Ratio Theorem [3]*). *Let $w, w_1, w_2 \in R^n$ be such that $w = w_1 + w_2$. Let $x \in R^n$ be a feasible solution (with respect to C) that is r-*approximate *with respect to w_1, and with respect to w_2. Then, x is r-*approximate *with respect to w as well.*

As a direct application of the local ratio method, the vertex cover problem has a 2-*approximation* algorithm. Let $w(u)$ denote the weight of vertex u. The local ratio method is stated as follows.

Algorithm 2 Local ratio method for vertex cover

```
1: while there exists an edge (u, v) such that min{w(u), w(v)} > 0 do
2:    ϵ = min{w(u), w(v)},
3:    w(u) ← w(u) − ϵ,
4:    w(v) ← w(v) − ϵ,
5: end while
6: return {v|w(v) = 0}.
```

The algorithm splits the original weight vector w into some components, i.e. $w = w_1 + w_2 + \cdots + w_k + w_0$ where w_i $(i = 1, 2, \ldots, k)$ denotes the decrease occurred to w in the ith round, and w_0 is the remained weight vector while the algorithm ends. Let (u_i, v_i) be the edge selected in the ith round, and ϵ_i be the decrease on each of its endpoints. As a feasible solution must cover at least one endpoint of an edge and at most two of them, comparing the solution x obtained by the algorithm with any feasible solution x^*, we have

$$w_i \cdot x \leq 2\epsilon_i,\ w_i \cdot x^* \geq \epsilon_i.$$

Therefore, we have $w_i \cdot x \leq 2w_i \cdot x^*$. Together with the fact $w_0 \cdot x = 0 \leq 2w_0 \cdot x^*$, The Local Ratio Theorem guarantees that x is 2-*approximate* to the original weight vector w.

3. Algorithm and analysis

For the problem $P_m|\text{vertex cover}|C_{max}$, it is natural to study a two-stage approach, i.e. first find a feasible solution for the vertex cover problem and then schedule the jobs (vertices) on the parallel machines. Unfortunately, the instance described in the Introduction shows that it may result in a bad solution even if we can optimally solve the two problems respectively. For the same instance, the solution found by the local ratio method includes all of the jobs, and then the LPT rule will return a schedule with makespan m for $m > 1$. Therefore, to obtain a satisfied solution, we need to integrate the two problems and the corresponding algorithms. Notice that the job with the longest processing time occupies one machine in the instance, and if we replace it by its neighbors, the solution will still be feasible (cover the graph) and the makespan will be improved. This inspires us to connect the LPT rule and the local ratio method by a replacement policy.

The main idea lies that we first solve a vertex cover problem by the local ratio method, and then obtain a current schedule by the LPT rule. While the last completed job occupies one machine and it has not been replaced before, we replace it by its neighbors and schedule the jobs by the LPT rule. Otherwise, output the best schedule so far.

Before formally describing this algorithm, we give some notations. Let S_i be the set of jobs processed on machine M_i by an algorithm, and then we can use the set $\{S_1, S_2, \ldots, S_m\}$ to describe a schedule. Let $p(S) = \sum_{j \in S} p_j$ be the total weight of jobs in a set S and $N_j = \{J_i | (J_j, J_i) \in E\}$ be the set of neighbors of job J_j. Use $\{S_1, S_2, \ldots, S_m\}$ to record a best schedule so far with makespan C_{max}, and let S be the set of jobs in this schedule. Use $\{S'_1, S'_2, \ldots, S'_m\}$ to record a current schedule with makespan C'_{max}, and let S' be the set of jobs in this schedule. We name the algorithm LLR that can be described as follows.

First check the feasibility of the LLR algorithm. If the jobs in a set S cover the graph, then the graph will still be covered when we replace a job $J_k \in S$ by its neighbors. The algorithm starts from a set derived by the local ratio method that covers the graph, and hence we will find a feasible solution when the algorithm ends.

Algorithm 3 LLR algorithm

solve vertex cover problem by the local ratio method to obtain a job set S'.
schedule the jobs in S' by the LPT rule, let the schedule be $\{S'_1, S'_2, \ldots, S'_m\}$ with makespan C'_{max}, and the last completed job be J_k.
let $S = S'$, $\{S_1, S_2, \ldots, S_m\} = \{S'_1, S'_2, \ldots, S'_m\}$ and $C_{max} = C'_{max}$, unmark all jobs.
while J_k is a single job processed on one machine and J_k is not marked **do**
  mark J_k, $S' \leftarrow S' \cup N_k \setminus \{J_k\}$,
  schedule the jobs in S' by the LPT rule, let the schedule be $\{S'_1, S'_2, \ldots, S'_m\}$ with makespan C'_{max}, and the last completed job be J_k.
  if $C'_{max} < C_{max}$ **then**
    $S \leftarrow S'$, $\{S_1, S_2, \ldots, S_m\} \leftarrow \{S'_1, S'_2, \ldots, S'_m\}$, $C_{max} \leftarrow C'_{max}$.
  end if
end while
return S, $(S_1, S_2, \ldots, S_m)$ and C_{max}.

Now we consider the computational complexity of the LLR algorithm. The local ratio method and the LPT rule can be realized in $O(|E|)$ and $O(n \log n)$ times respectively, and hence the first three steps need $O(\max\{|E|, n \log n\})$ time. The main computation occurs when the last completed job occupies one machine and it is not marked. Notice that any job can be replaced at most once, and hence there are at most n replacements, and each of them corresponds to $O(n)$ time to update the sets S', S, $\{S_1, S_2, \ldots, S_m\}$ and C_{max}, $O(n)$ time to assign jobs in a given list on m parallel machines, and the total computation time for these operations is $O(n^2)$. To sort the jobs in the LPT rule after replacing J_k by N_k, we only need insert the jobs of N_k into the remaining list, and it needs $O(|N_k| \log n)$ time, whose total time is $O(\sum_{k=1}^{n} |N_k| \log n) = O(|E| \log n)$. Therefore, it needs $O(\max\{n^2, |E| \log n)\}$ time during the iterations. The last step needs $O(n)$ time to output a solution. Therefore, the LLR algorithm will halt in $O(\max\{n^2, |E| \log n)\}$ time.

Notice that the LLR algorithm ends only when the last completed job in a "current schedule" does not occupy one machine or it has been marked before. Let S' be the job set in this schedule with makespan C'_{max}, and J_k be the last completed job which starts its processing at time y. This is to say $C'_{max} = y + p_k$. Let C_{max} be the makespan returned by the LLR algorithm. Recall that the schedule returned is the best one during the LLR algorithm, and we have $C_{max} \leq C'_{max}$. Let S_0 be the set of jobs obtained by the local ratio method, and S^* be the job set of an optimal solution for $P_m|\text{vertex cover}|C_{max}$, and C^*_{max} be the optimal makespan.

We first present a technical lemma.

Lemma 1. *If any marked job belongs to S^*, then the LLR algorithm will return an optimal solution.*

Proof. Any marked job, say J_i, corresponds to a current schedule with makespan p_i. Since the schedule returned is the best one during the LLR algorithm, we have $C_{max} \leq p_i$. If J_i belongs to S^*, then it is obvious that $C^*_{max} \geq p_i$, and hence we have $C_{max} \leq C^*_{max}$. It implies that the LLR algorithm returns an optimal solution. □

In the following, we will generally assume that any marked job does not belong to S^* since otherwise the LLR algorithm will find an optimal solution. The following two lemmas will focus on the last "current schedule".

Lemma 2. $\forall \tilde{S} \subset S'$, *if* $\tilde{S} \cap S^* = \emptyset$, *then* $p(\tilde{S}) \leq p(S^*)$.

Proof. If a marked job, say J_j, does not belong to S^*, we know all of its neighbors must belong to S^* to satisfy the covering constraints. Since we have assumed that any marked job does not belong to S^*, the jobs in $S' \setminus S_0$ must belong to S^*. Given $\tilde{S} \subset S'$ such that $\tilde{S} \cap S^* = \emptyset$, we know $\tilde{S} \subset S_0$. Recall the procedure of the local ratio method. Let ϵ_i be the drop of each endpoint of an edge in the ith round. Since $\tilde{S} \subset S_0$ and $\tilde{S} \cap S^* = \emptyset$, we know that $\tilde{S}$ contains at most one of the endpoints since at least one of them belongs to S^*. Therefore $p(\tilde{S})$ drops at most ϵ_i while $p(S^*)$ drops at least ϵ_i. At the end of the local ratio method, $p(\tilde{S})$ must drop to zero, and the remaining weight of S^* is no less than zero. Therefore, the original weights $p(\tilde{S})$ and $p(S^*)$ must satisfy $p(\tilde{S}) \leq p(S^*)$. □

The next lemma will evaluate the ratio of C'_{max} to C^*_{max}. A simple observation is that $C^*_{max} \geq \frac{1}{m} p(S^*)$.

Lemma 3. $\frac{C'_{max}}{C^*_{max}} \leq (1 - \frac{1}{m}) \frac{p_k}{C^*_{max}} + 2$.

Proof. Based on a similar argument for the LPT rule, we have

$$\begin{aligned} \frac{C'_{max}}{C^*_{max}} &= \frac{p_k + y}{C^*_{max}} \\ &\leq \frac{\frac{1}{m} p(S') + (1 - \frac{1}{m}) p_k}{C^*_{max}}. \end{aligned} \tag{1}$$

Noticing that $S' = (S' \cap S^*) \cup (S' \setminus S^*)$ and $C^*_{max} \geq \frac{1}{m}p(S^*)$, we have

$$\frac{\frac{1}{m}p(S')}{C^*_{max}} \leq \frac{p(S')}{p(S^*)} = \frac{p(S' \cap S^*)}{p(S^*)} + \frac{p(S' \setminus S^*)}{p(S^*)} \leq 2, \tag{2}$$

where the last inequality comes from Lemma 2. Put (1) and (2) together, and the result follows. □

The following theorem presents our main result.

Theorem 2. *The LLR Algorithm is a* $\left(3 - \frac{2}{m+1}\right)$-approximation *algorithm for* $P_m|vertex\ cover|C_{max}$.

Proof. Consider the last "current schedule". As defined before, let S' be the job set in this schedule with makespan C'_{max}, and J_k be the last completed job which starts its processing at time y. We know $C'_{max} = y + p_k$.

When the algorithm ends, it holds that either J_k is a single job processed on one machine and J_k has been marked before, or J_k is not a single job processed on one machine. We will analyze the performance of the LLR algorithm with respect to the two cases.

Case 1: *J_k is a single job processed on one machine and J_k has been marked*

In this case, J_k must be deleted in some iteration since it has been marked before, and be rejoined by replacing one of its marked neighbors, say J_i. Notice that any marked job J_j corresponds to a schedule with makespan p_j, and we have $C_{max} \leq \min\{p_k, p_i\}$ since the the schedule returned is the best one during the algorithm. To satisfy the covering constraints, any feasible solution must include at least one of J_k and J_i, and hence $C^*_{max} \geq \min\{p_k, p_i\} \geq C_{max}$. In this case, the LLR algorithm returns an optimal solution.

Case 2: *J_k is not a single job processed on one machine*

In this case, we have $y \geq p_k$ by the LPT rule. We will show that $C^*_{max} \geq p_k$. Notice that there are at least $m + 1$ jobs in S' each with processing time no less than p_k. If any of them belong to S^*, then we have $C^*_{max} \geq p_k$. Otherwise, by Lemma 2, we have $p(S^*) \geq (m + 1)p_k$, and then $C^*_{max} \geq \frac{1}{m}p(S^*) \geq \frac{m+1}{m}p_k \geq p_k$.

Assume that there are t jobs in S' such that each of them starts processing at time zero and is completed no earlier than time y. Let the set of these jobs be T, and we have $p(T) \geq ty$. These jobs occupy t machines before time y, and each of the remaining $m - t$ machines is assigned at least two jobs before time y. Let U be the set including those jobs and J_k, and then we have $|U| \geq 2(m - t) + 1$ and the processing time of each job in U is no less than p_k.

Subcase 2.1: $T \cap S^* \neq \emptyset$

In this case, C^*_{max} is no less than the processing time of each job in $T \cap S^*$, and hence $C^*_{max} \geq y$. We have

$$\frac{C'_{max}}{C^*_{max}} \leq \frac{y + p_k}{y} = 2 \leq 3 - \frac{2}{m+1}. \tag{3}$$

Subcase 2.2: $T \cap S^* = \emptyset$ *and* $t \geq \frac{m+1}{2}$

Since $T \cap S^* = \emptyset$, by Lemma 2, we have $p(S^*) \geq p(T) \geq \frac{m+1}{2}y$. Therefore, we have $C^*_{max} \geq \frac{1}{m}p(S^*) \geq \frac{m+1}{2m}y$. We have shown that $C^*_{max} \geq p_k$, and thus we have

$$\frac{C'_{max}}{C^*_{max}} = \frac{y + p_k}{C^*_{max}} \leq \frac{y}{C^*_{max}} + 1 \leq \frac{2m}{m+1} + 1 = 3 - \frac{2}{m+1}. \tag{4}$$

Subcase 2.3: $T \cap S^* = \emptyset$ *and* $t < \frac{m+1}{2}$

Subcase 2.3.1: $|U \cap S^*| \geq m + 1$

At least two jobs in $|U \cap S^*|$ should be assigned on one machine in the optimal solution. This implies that $C^*_{max} \geq 2p_k$. By Lemma 2, we have

$$\frac{C'_{max}}{C^*_{max}} \leq \left(1 - \frac{1}{m}\right)\frac{p_k}{C^*_{max}} + 2 \leq \frac{5}{2} - \frac{1}{2m} \leq 3 - \frac{2}{m+1}. \tag{5}$$

Subcase 2.3.2: $|U \cap S^*| \leq m$

Since $|U \cap S^*| \leq m$, at least $2(m - t) + 1 - m = m - 2t + 1$ jobs belong to $U \setminus S^*$. Noticing that $T \cap S^* = \emptyset$, $T \cap U = \emptyset$ and $(T \cup (U \setminus S^*)) \subset S'$, by Lemma 2, we have

$$p(S^*) \geq p(T \cup (U \setminus S^*)) = p(T) + p(U \setminus S^*) \geq ty + (m - 2t + 1)p_k = t(y - 2p_k) + (m + 1)p_k. \tag{6}$$

Subcase 2.3.2.1: $y \geq 2p_k$

Since $y \geq 2p_k$ and (6), we have $p(S^*) \geq t(y-2p_k)+(m+1)p_k \geq (m+1)p_k$, and $C^*_{max} \geq \frac{1}{m}p(S^*) \geq \frac{m+1}{m}p_k$. By Lemma 3, we have

$$\frac{C'_{max}}{C^*_{max}} \leq \left(1-\frac{1}{m}\right)\frac{p_k}{C^*_{max}}+2 \leq \left(1-\frac{1}{m}\right)\frac{m}{m+1}+2 = 3-\frac{2}{m+1}. \tag{7}$$

Subcase 2.3.2.2: $y < 2p_k$

Since $y < 2p_k$, $t < \frac{m+1}{2}$ and (6), we have $p(S^*) \geq t(y-2p_k)+(m+1)p_k \geq \frac{m+1}{2}(y-2p_k)+(m+1)p_k = \frac{m+1}{2}y$. Based on a similar argument of obtaining (4), we also have $\frac{C'_{max}}{C^*_{max}} \leq 3-\frac{2}{m+1}$.

Putting all of the results together and noticing that $C_{max} \leq C'_{max}$, we know

$$\frac{C_{max}}{C^*_{max}} \leq \frac{C'_{max}}{C^*_{max}} \leq 3-\frac{2}{m+1}, \tag{8}$$

and hence the LLR algorithm is a $(3-\frac{2}{m+1})$-*approximation* algorithm. The following instance shows that the bound is tight.

Consider an instance of $P_m|\text{vertex cover}|C_{max}$ with m machines and a graph $G = \{J, E, P\}$, in which $J = \{J_1, \ldots, J_{2m+1}\}$, $E = \{(J_i, J_{i+1})|i = 1, \ldots, 2m\}$, and the processing times are $p_1 = p_3 = \cdots = p_{2m+1} = m$, $p_2 = p_4 = \cdots = p_{2m} = m+1$. We consider the edges in the order of $(J_1, J_2) \to (J_2, J_3) \to \cdots \to (J_{2m}, J_{2m+1})$, and then the local ratio method will return all jobs in J as a vertex cover, and then the LPT rule will return a schedule $\{\{J_2, J_1, J_{2m+1}\}, \{J_4, J_3\}, \ldots, \{J_{2m}, J_{2m-1}\}\}$. The last completed job J_{2m+1} is not a single job on M_1, therefore the LLR algorithm will return this schedule with $C_{max} = 3m+1$. However, the optimal solution is $\{\{J_2\}, \{J_4\}, \ldots, \{J_{2m}\}\}$ with $C^*_{max} = m+1$. Thus $\frac{C_{max}}{C^*_{max}} = 3-\frac{2}{m+1}$ for this instance. □

4. Conclusions

This paper considers a combination of parallel machine scheduling and the vertex cover problem. We present an approximation algorithm that bridges the local ratio method and the LPT rule by a replacement policy, and prove that it is $(3-\frac{2}{m+1})$-*approximate*. Though the local ratio method used in this paper and the LPT rule are not the most efficient algorithms for the two problems respectively, their integration performs even better than optimally solving each of the two problems successively.

It is interesting to find an approximation algorithm with a better performance ratio, or give a nontrivial inapproximability result for this problem. This study may also inspire us to consider the combinations of other optimization problems.

Acknowledgments

Wang's research work has been supported by SRF for ROCS, SEM, Bilateral Scientific Cooperation Project between Tsinghua University and K.U. Leuven, and NSFC No. 11171177. We would like to thank the anonymous referees for helpful comments and suggestions. We also thank Roel Leus for helpful discussions.

References

[1] V. Bafna, T. Fujito, A 2-approximation algorithm for the undirected feedback vertex set problem, SIAM J. Discrete Math. 12 (1999) 289–297.
[2] A. Bar-Noy, R. Bar-Yehuda, J. Naor, B. Shieber, A unified approach to approximating resource allocation and scheduling, J. ACM 48 (2001) 1069–1090.
[3] R. Bar-Yehuda, K. Bendelm, Local ratio: a unified framework for approximation algorithms, ACM Comput. Surv. 36 (2004) 422–463.
[4] R. Bar-Yehuda, S. Even, A linear time approximation algorithm for the weighted vertex cover problem, J. Algorithms 2 (1981) 198–203.
[5] R. Bar-Yehuda, S. Even, A local-ratio theorem for approximating the weighted vertex cover problem, Ann. Discrete Math. 25 (1985) 27–45.
[6] R. Bar-Yehuda, M. Halldórsson, J. Naor, H. Shachnai, I. Shapira, Scheduling split intervals, in: Proc. 13th ACM-SIAM Symp. on Discrete Algorithms, 2002, pp. 732–741.
[7] R. Bar-Yehuda, D. Rawitz, On the equivalence between the primal–dual schema and the local-ratio technique, in: 4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, in: Lecture Notes in Computer Science, vol. 2129, 2001, pp. 24–35.
[8] R. Bar-Yehuda, D. Rawitz, Approximating element-weighted vertex deletion problems for the complete k-partite property, J. Algorithms 42 (2002) 20–40.
[9] R. Bar-Yehuda, D. Rawitz, Local ratio with negative weights, Oper. Res. Lett. 32 (2004) 540–546.
[10] B. Chen, C.N. Potts, G.J. Woeiginger, A review of machine scheduling: complexity, algorithm and approximability, in: D-Z. Du, P. Pardalos (Eds.), Handbook of Combinatorial Optimization, vol. 3, Kluwer Academic Publishers, 1998, pp. 21–169.
[11] I. Dinur, S. Safra, On the hardness of approximating minimum vertex-cover, Ann. Math. 162 (2005) 439–485.
[12] T. Fujito, A unified approximation algorithm for node-deletion problems, Discrete Appl. Math. 86 (1998) 213–231.
[13] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, WH Freeman, San Francisco, 1979.
[14] R.L. Graham, Bounds for certain multiprocessing anomalies, Bell Syst. Tech. J. 45 (1966) 1563–1581.
[15] R.L. Graham, Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17 (1969) 416–426.
[16] D.S. Hochbaum, Approximation Algorithms for NP-Hard Problems, PWS Publishing, 1997, pp. 94–143.
[17] D.S. Hochbaum, D.B. Shmoys, Using dual approximation algorithms for scheduling problems: theoretical and practical results, J. ACM 34 (1987) 144–162.
[18] S. Khot, O. Regev, Vertex cover might be hard to approximate to within $2-\epsilon$, J. Comput. Syst. Sci. 74 (2008) 335–349.
[19] J.Y. Leung, Handbook of Scheduling, CRC Press, 2004.
[20] S. Sahni, Algorithms for scheduling independent tasks, J. ACM 23 (1976) 116–127.
[21] D.P. Williamson, The primal dual method for approximation algorithms, Math. Program. 91 (2002) 447–478.