Approximating Scheduling Machines with Capacity Constraints

(This is a correction of paper at FAW2009)

Chi Zhang [*], Gang Wang, Xiaoguang Liu, and Jing Liu

Nankai-Baidu Joint Lab, College of Information Technical Science, Nankai University

Abstract. In the Scheduling Machines with Capacity Constraints problem, we are given k identical machines, each of which can process at most m_i jobs. M jobs are also given, where job j has a non-negative processing time length $t_j \geq 0$. The task is to find a schedule such that the makespan is minimized and the capacity constraints are met. In this paper, we present a 3-approximation algorithm using an extension of *Iterative Rounding Method* introduced by Jain [4]. To the best of the authors' knowledge, this is the first attempt to apply *Iterative Rounding Method* to scheduling problem with capacity constraints.

Key words: Approximation, Scheduling, Capacity Constraints, Iterative Rounding

1 Introduction

We consider the Scheduling Machines with Capacity Constraints problem (SMCC): There are k identical machines, and machine i can process at most m_i jobs. Given $M \leq \sum_{1 \leq i \leq k} m_i$ jobs with their processing time lengths, we are to find a schedule of jobs to machines that minimizes the makespan and meets the capacity constraints.

Scheduling problem is a classical **NP**-Hard problem and has been studied extensively. In the general setting, we are given set T of tasks, number k of machines, length $l(t, i) \in \mathbf{Z}^+$ for each $t \in T$ and machine $i \in [1..k]$, the task is to find a schedule for T, namely, a function $f : T \rightarrow [1..k]$, to minimize $\max_{i \in [1..k]} \sum_{t \in T, f(t)=i} l(t, i)$. Lenstra, Shmoys and Tardos [6] gave a 2-approximation algorithm for the general version and proved that for any $\epsilon > 0$ no $(\frac{3}{2} - \epsilon)$-approximation algorithm exists unless $\mathbf{P} = \mathbf{NP}$. Their method based on applying rounding techniques on fractional solution to linear programming relaxation. Gairing, Monien and Woclaw [2] gave a faster combinatorial 2-approximation algorithm for the general problem. They replaced the classical technique of solving the LP-relaxation and rounding afterwards by a completely

integral approach. For the variation in which the number of processors k is constant, Angel, Bampis and Kononov [1] gave a *fully polynomial-time approximation scheme (FPTAS)*. For the uniform variation where $l(t, i)$ is independent of the processor i, Hochbaum and Shmoys [3] gave a *polynomial-time approximation scheme (PTAS)*.

The SMCC problem is one of the uniform variations, with capacity constraints on machines. One special case of SMCC problem in which there are only two identical machines was studied in [8] [10] [11]. Woeginger [9] gave a FPTAS for the same problem. General SMCC problem is a natural generalization of scheduling problem without capacity constraints and can be used in some applications in real world, such as students distributions in university, the Crew Scheduling problem in Airlines Scheduling [12] [13], etc. In the Crew Scheduling problem, crew rotations, sequences of flights legs to be flown by a single crew over a period of a few days, are given. Crews are paid by the amount of flying hours, which is determined by the scheduled rotations. Airline company wants to equalize the salaries of crews, i.e. to make the highest salary paid to crews minimum. Rotations starts and ends at the same crew base and must satisfy a large set of work rules based labor contracts covering crew personnel. In the concern of safety issues, one common contract requirement is the maximum times of flying of a single crew in a period of time. So the aim is to find a scheduling of rotations to crews that minimizes the highest salary and meets the maximum flying times constraints.

In many literature, researchers approached scheduling problem using rounding techniques. Lenstra, Shmoys and Tardos [6] applied rounding method to the decision problem to derive a *ρ-relaxed decision procedure* and then used a binary search to obtain an approximation solution. In the SMCC problem, the capacity constraints defeat many previous methods. In this paper, our algorithm is one of the class of rounding algorithms, but use a different rounding method introduced by Jain [4]. We do not round off the whole fractional solution in a single stage. Instead, we round it off iteratively.

Iterative Rounding Method, introduced by Jain [4], was used in his breakthrough work on the Survivable Network Design problem. This rounding method does not need the half-integrality, but only requires that at each iteration there exist some variables with bounded values. In [4], Jain observed that at each iteration one can always find a edge e has x_e at least $1/2$, which ensures that the algorithm has an approximation ratio of 2 . As a successful extension of Jain's method, Mohit Singh and Lap Chi Lau [7] considered the Minimum Bounded Degree Spanning Trees problem and gave an algorithm that produces a solution, which has at most the cost of optimal solution while violating vertices degrees constraints by 1 at most. As far as the authors know, *Iterative Rounding Method* has been used in graph problems, and has produced many beautiful results.

In this paper, we apply *Iterative Rounding Method* to the scheduling problem with capacity constraints and obtain a 3-approximation algorithm. To the best of the authors' knowledge, this is the first attempt to approach scheduling problem with capacity constraints using *Iterative Rounding Method*.

The rest of the paper is organized as follows. In Section 2, we formulate the SMCC problem as an Integer Program, give its natural relaxation and introduce our relaxation, Bounded Linear Programming Relaxation (BLPR). In Section 3, we present some properties of BLPR and prove theorems that support our algorithm. In Section 4, we present bounding theorems and an approximation algorithm, IRA, and prove that it has an approximation ratio of 3 .

2 Preliminary

Formally, the SMCC problem is as follows: Given a positive integer k, k positive integers $\{m_i | m_i > 0, 1 \le i \le k\}$, M non-negative integers $\{t_j | t_j \ge 0, 1 \le j \le M \le \sum_{i=1}^{k} m_i\}$, we are to solve the following Integer Program (IP):

$$
\begin{aligned}
&\text{minimize} && c \\
&\text{subject to } \sum_{j=1}^{M} x_{ij} t_j - c \le 0 && 1 \le i \le k \\
&\qquad\qquad \sum_{j=1}^{M} x_{ij} \le m_i && 1 \le i \le k \\
&\qquad\qquad \sum_{i=1}^{k} x_{ij} = 1 && 1 \le j \le M \\
&\qquad\qquad x_{ij} \in \{0,1\} && 1 \le i \le k, 1 \le j \le M
\end{aligned}
$$

There are some relaxations, one of which is the following natural Linear Programming Relaxation (LPR) dropping the integrality constraints.

$$
\begin{aligned}
&\text{minimize} && c \\
&\text{subject to } \sum_{j=1}^{M} x_{ij} t_j - c \le 0 && 1 \le i \le k \\
&\qquad\qquad \sum_{j=1}^{M} x_{ij} \le m_i && 1 \le i \le k \\
&\qquad\qquad \sum_{i=1}^{k} x_{ij} = 1 && 1 \le j \le M \\
&\qquad\qquad x_{ij} \ge 0 && 1 \le i \le k, 1 \le j \le M
\end{aligned}
$$

We don't use LPR directly, but use an alternative relaxation, Bounded Linear Programming Relaxation (BLPR): Given a positive integer k, k positive integers $\{m_i | m_i > 0, 1 \le i \le k\}$, M non-negative integers $\{t_j | t_j \ge 0, 1 \le j \le M \le \sum_{i=1}^{k} m_i\}$, a real vector $b = (b_1, b_2, \ldots, b_k)$ and $\mathcal{F} \subseteq \{(i,j) | 1 \le i \le k, 1 \le j \le M\}$, find a feasible solution under the following constraints

$$
\text{subject to} \qquad \sum_{j=1}^{M} x_{ij} t_j \le b_i \qquad\qquad 1 \le i \le k \qquad (1)
$$

$$
\sum_{j=1}^{M} x_{ij} \le m_i \qquad\qquad 1 \le i \le k \qquad (2)
$$

$$
\sum_{i=1}^{k} x_{ij} = 1 \qquad\qquad 1 \le j \le M \qquad (3)
$$

$$
x_{ij} = 1 \qquad\qquad (i,j) \in \mathcal{F} \qquad (4)
$$

$$
x_{ij} \ge 0 \qquad\qquad 1 \le i \le k, 1 \le j \le M \qquad (5)
$$

where vector $b = (b_1, b_2, \ldots, b_k)$, called *upper bounding vector*, is added to depict the different upper bounds of machines more precisely, and $\mathcal{F}$ is added to represent the partial solution in algorithm. Each $(i, j) \in \mathcal{F}$ indicates that job j has been scheduled to machine i . Those $\{x_{ij} | (i, j) \in \mathcal{F}\}$ are considered as constants.

We will show that properly constructing vector $b = (b_1, b_2, \ldots, b_k)$ makes the solution produced by our algorithm under control and easy to analyze.

Definition 1. *In a BLPR problem Λ,* upper bounding vector $b = (b_1, b_2, \ldots, b_k)$ *is called* feasible *if Λ is feasible.*

Keeping the *upper bounding vector b feasible* all the time is the key of our algorithm, which guarantees that we can always find a feasible solution bounded by b .

3   Techniques

Before we present our algorithm, we need to introduce some properties of BLPR.

With respect to the partial solution $\mathcal{F}$, let c_i denote the number of already scheduled jobs in machine i, namely, $c_i = |\{(i, j) | (i, j) \in \mathcal{F}\}|$. Note that $m_i - c_i$ indicates the *free* capacity in machine i .

We call a job *free* if it has not been scheduled to any machine and call a machine *free* if it still has free capacity. For a feasible fractional solution x to Λ, define a bipartite graph $G(x) = G(L, R, E)$, called *supporting graph*, where L represents the set of *free* machines, R represents the set of *free* jobs and $E = \{(i, j) | x_{ij} > 0, (i, j) \notin \mathcal{F}\}$. We denote the number of *free* jobs and the number of *free* machines by M^* and k^* respectively. Note that for *free* job j, $\sum_{(i,j) \in E} x_{ij} = 1$.

Consider the *Constraint Matrix* of Λ, which consists of the coefficients of the left side of equalities and inequalities, except for the non-negativity constraints from (5):

$$
\begin{pmatrix}
\begin{pmatrix} t_1 \ t_2 \ \ldots \ t_M \end{pmatrix} & & & \\
& \begin{pmatrix} t_1 \ t_2 \ \ldots \ t_M \end{pmatrix} & & \\
& & \ddots & \\
& & & \begin{pmatrix} t_1 \ t_2 \ \ldots \ t_M \end{pmatrix} \\
\begin{pmatrix} 1 \ 1 \ \ldots \ 1 \end{pmatrix} & & & \\
& \begin{pmatrix} 1 \ 1 \ \ldots \ 1 \end{pmatrix} & & \\
& & \ddots & \\
& & & \begin{pmatrix} 1 \ 1 \ \ldots \ 1 \end{pmatrix} \\
\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} & \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} & \cdots & \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}
\end{pmatrix}
\tag{6}
$$

where the 1st to k^{th} rows represent the constraints from (1), the $(k+1)^{\text{th}}$ to $(2k)^{\text{th}}$ rows represent the constraints from (2), and the $(2k+1)^{\text{th}}$ to $(2k+M)^{\text{th}}$ rows represent the constraints from (3).

One can verify that the $(k+1)^{\text{th}}$ row can be linearly expressed by the rest of rows. Thus the rank of *Constraints Matrix* is bounded by the following lemma

Lemma 1. Constraints Matrix *has a rank at most $M + 2k - 1$* .   □

Recall that, a basic solution x to Λ is the unique solution determined by a set of linearly independent *tight constraints* that are satisfied as equalities. We remove all zero variables in x so that no *tight constraints* comes from (5). Thus the number of non-zero variables in x never exceeds the rank of *Constraints Matrix*. When $\mathcal{F} = \emptyset$, the following inequality holds

$$|E| \leq M + 2k - 1 \tag{7}$$

We can remove those non-*free* machines from Λ, move fixed variables $\{x_{ij}|(i,j) \in \mathcal{F}\}$ to the right side of the equalities and inequalities as constants and remove variables fixed to 0. By doing this, we obtain a new sub-problem and only focus on *free* jobs and *free* machines. In the new sub-problem, Lemma 1 holds. So in general, the following corollary holds.

Corollary 1. *Given a BLPR problem, Λ, its basic solution x and* supporting graph $G(x) = G(L, R, E)$, *we have*

$$|E| \leq M^* + 2k^* - 1$$

□

We introduce lemmas on the basic solution to Λ when there are no less *free* jobs than twice the *free* machines, namely, $M^* \geq 2k^*$.

Lemma 2. *If Λ is feasible with $M^* \geq 2k^*$ and x is a basic solution, there exist $M^* - 2k^* + 1$ variables with values of 1* .

Proof. For a basic solution x, we construct *supporting graph* $G(x) = G(L, R, E)$. Suppose that l of M^* *free* jobs are of degree of at most one in G. Note that each of them has degree at least one. Each of the rest $M^* - l$ *free* jobs has degree of more than one. The following inequality holds.

$$|E| \geq 2(M^* - l) + l = 2M^* - l \tag{8}$$

By Corollary 1, we have

$$l \geq M^* - 2k^* + 1 \tag{9}$$

□

The following corollary holds

Corollary 2. *If Λ is feasible with $M^* \geq 2k^*$ and x is a basic solution, there exist a* free *machine p and a* free *job q such that $x_{pq} = 1$* .   □

4   A 3-approximation Algorithm

In this section, we present an approximation algorithm IRA . Let $\mathcal{A}$ denote the makespan in the solution produced by IRA, $\mathcal{OPT}$ denote the makespan in the optimal solution to Δ .

We introduce three bounding theorems on BLPR. Noting that when $M^* \geq 2k^*$ we can find a $x_{pq} = 1$, we can schedule q to p without increasing the lengths in the fractional solution. We first show the theorem for the case $M^* \geq 2k^*$.

Theorem 1. *Given a BLPR problem, Λ, with $M^* \geq 2k^*$ and its basic solution x . Based on Λ, we construct a new BLPR problem $f(\Lambda)$ as follows:*

1. *Find a variable $x_{pq} = 1$;*
2. *$\mathcal{F}' \leftarrow \mathcal{F} + (p,q)$;*
3. *The rest parts of $f(\Lambda)$ are the same as Λ .*

If b is a feasible upper bounding vector of Λ then b' is a feasible upper bounding vector of $f(\Lambda)$.   □

When $M^* < 2k^*$ and some *free* machines have *free* capacity of one, we have the following theorem.

Theorem 2. *Given a BLPR problem, Λ, with $M^* < 2k^*$ and its basic solution x. Moreover some* free *machines have* free *capacity of 1. Based on Λ, we construct a new BLPR problem $g(\Lambda)$ as follows:*

1. *Let p denote a machine with 1* free *capacity;*
2. *Let q denote the* free *job with the largest length;*
3. *$\mathcal{F}' \leftarrow \mathcal{F} + (p,q)$;*
4. *$b' \leftarrow (b_1, b_2, \ldots, b_{p-1}, b_p + t_q, b_{p+1}, \ldots, b_k)$;*
5. *The rest parts of $g(\Lambda)$ are the same as Λ .*

If b is a feasible upper bounding vector of Λ then b' is a feasible upper bounding vector of $g(\Lambda)$.

Proof. To schedule q to p, for each *free* machine $s \neq p$ with $x_{sq} > 0$, we move x_{sq} fraction of job q to machine p then move back from p as much as possible but no more than x_{sq} fraction of jobs other than q as in Algorithm 1.

Because q has the largest length among the *free* jobs, we can guarantee that the length of each *free* machine $p' \neq p$ will not increase and the length of machine p will increase by at most t_q . Note that in $g(\Lambda)$, $x_{pq} = 1$ and p is no longer *free.* This implies there is a feasible solution to $g(\Lambda)$.   □

When $M^* < 2k^*$ and every *free* machine has more than 1 *free* capacity, we can schedule these M^* jobs arbitrarily but only assuring that each machine gets no more than 2 jobs. One can prove the following theorem.

Theorem 3. *Given a feasible BLPR problem, Λ, with $M^* < 2k^*$. Moreover every* free *machine has* free *capacity of at least 2. We construct a new BLPR problem $h(\Lambda)$ as follows:*

Algorithm 1 $G - transition$

Require: A BLPR Λ with $M^* < 2k^*$ and machine p has *free* capacity of 1
Ensure: A BLPR $g(\Lambda)$
1: Let x be a basic feasible solution to Λ and q be the longest *free* job;
2: **while** there exists $p' \neq p$ such that $x_{p'q} > 0$ **do**
3:    **if** there exists $q' \neq q$ such that $x_{pq'} > 0$ **then**
4:       $\alpha \leftarrow \min\{x_{p'q}, x_{pq'}\}$;
5:       $x_{p'q} \leftarrow x_{p'q} - \alpha$;
6:       $x_{pq'} \leftarrow x_{pq'} - \alpha$;
7:    **else** $\{\forall q' \neq q, x_{pq'} = 0\}$
8:       $x_{pq'} \leftarrow x_{pq'} + x_{p'q}$;
9:    **end if**
10: **end while**

---

1. *Schedule* free *jobs arbitrarily but only assuring that each machine gets no more than 2 jobs;*
2. *For each machine p, increase b_p by the sum of job(s) scheduled to p and update $\mathcal{F}$ accordingly;*
3. *The rest parts of $h(\Lambda)$ are the same as Λ .*

By doing this, we obtain $h(\Lambda)$ in which all jobs have been scheduled and $h(\Lambda)$ is feasible . □

By Theorem 1, as long as $M^* \geq 2k^*$, we can always schedule a *free* job to a *free* machine but without increasing its length. If $M^* < 2k^*$, Theorem 2 and 3 guarantee we still can make our decision in a fairly simple way. We present our algorithm using *Iterative Rounding Method*, IRA, in Algorithm 2.

Finding a basic solution to a linear program can be done in polynomial time by using the ellipsoid algorithm [5] then converting the solution found into a basic one [4]. Together with the following observation

Lemma 3. *At Line 3, y is a feasible upper bounding vector of Λ .* □

the correctness of IRA follows from Theorem 1, 2 and 3.

Corollary 3. *Algorithm IRA always terminates in polynomial time.* □

The analysis of the performance of IRA is simple with the help of *upper bounding vector b*, noting that once a component of b is increased, the machine will be no longer *free*. We now show that IRA is a 3-approximation algorithm.

Theorem 4. IRA *is a 3-approximation algorithm.*

Proof. Consider any machine p with length $\mathcal{A}$ in the solution produced by IRA . Note that once b_p is increased at Line 9 machine p will no longer be *free*. So exactly one of the following statements is true when the algorithm terminates:

1. b_p hasn't been increased, then $b_p \leq y_p$;
2. b_p has been increased once at Line 9, then $b_p \leq y_p + t_q$ for some q ;

Algorithm 2 IRA

Require: An IP Δ
Ensure: A feasible integral solution $\mathcal{F}$
 1: Construct natural linear programming relaxation Γ ;
 2: Solve Γ optimally and let $y = (y_1, y_2, \ldots, y_k)$ be the lengths of machines in the optimal solution;
 3: Construct a BLPR Λ, letting y be the *upper bounding vector* and $\mathcal{F} = \emptyset$;
 4: **while** $M^* > 0$ **do**
 5:    **if** $M^* \geq 2k^*$ **then**
 6:       $\Lambda \leftarrow f(\Lambda)$;
 7:    **else** $\{M^* < 2k^*\}$
 8:       **if** there exists a machine p with *free* capacity of 1 **then**
 9:          $\Lambda \leftarrow g(\Lambda)$;
10:       **else** {every *free* machine have more than 1 capacity}
11:          $\Lambda \leftarrow h(\Lambda)$;
12:       **end if**
13:    **end if**
14: **end while**
15: **return** $\mathcal{F}$;

---

3. b_p has been increased once at Line 11, then $b_p \leq y_p + t_{q_1} + t_{q_2}$ for some q_1, q_2 .

Note that y_p and $\max_q\{t_q\}$ are two trivial lower bounds of $\mathcal{OPT}$. Also note that after the algorithm terminates, the integral solution produced by IRA, contained in $\mathcal{F}$, is also bounded by *upper bounding vector* b . By definition of *feasible upper bounding vector*, we have inequality

$$\mathcal{A} \leq b_p \leq 3\mathcal{OPT} \tag{10}$$

as expected. $\square$

5   Conclusion

In this paper, we consider the SMCC problem, a uniform variation of general scheduling problem, which has capacity constraints on identical machines. Using an extension of *Iterative Rounding Method* introduced by Jain [4], we obtain a 3-approximation algorithm. This is the first attempt to use *Iterative Rounding Method* in scheduling problem and it shows the power of *Iterative Rounding Method*. It is still unknown that whether the approximation ratio can be improved or whether the *Iterative Rounding Method* can be used to obtain a good approximation algorithm for the non-uniform version of scheduling problem with capacity constraints.

References

1. Angel, E., Bampis, E. and Kononov, A.: A FPTAS for Approximating the Unrelated Parallel Machines Scheduling Problem with Costs. In: 9th Annual

European Symposium on Algorithms, pp.194-205. August 28-31, (2001). LNCS, vol. 2161, Springer-Verlag, London, 194-205, (2001)

2. Gairing, M., Monien, B., and Woclaw, A.: A faster combinatorial approximation algorithm for scheduling unrelated parallel machines. Theor. Comput. Sci. 380, 1-2 (Jun. 2007), 87-99. (2007)

3. Hochbaum, D. S., and Shmoys, D. B.: Using dual approximation algorithms for scheduling problems: theoretical and practical results, J. ACM 34, 144-162, (1987)

4. K. Jain: A factor 2 approximation algorithm for the generalized Steiner network problem, Combinatorica, 21, pp.39-60, (2001)

5. L. G. Khachiyan.: A polynomial algorithm for linear programming (in Russian). Doklady Akademiia Nauk USSR 244, 10931096, (1979). A translation appears in: Soviet Mathematics Doklady 20, 191194, (1979)

6. Lenstra, J. K., Shmoys, D. B., and Tardos, É.: Approximation algorithms for scheduling unrelated parallel machines. Math. Programming 46, 259-271, (1990)

7. Mohit Singh, and Lap Chi Lau.: Approximating minimum bounded degree spanning trees to within one of optimal. In: The thirty-ninth annual ACM Symposium on Theory of Computing, San Diego, California, USA, pp.661-670, (2007)

8. L. H. Tsai.: Asymptotic Analysis of an Algorithm for Balanced Parallel Processor Scheduling. SIAM Journal on Computing, 21, pp.59-94, (1992)

9. Gerhard J. Woeginger.: A comment on scheduling two parallel machines with capacity constraints. Discrete Optimization, Volume 2, Issue 3, September 2005, pp.269-272, (2005)

10. Heng Yang, Yinyu Ye, and Jiawei Zhang.: An approximation algorithm for scheduling two parallel machines with capacity constraints. Discrete Applied Mathematics, Volume 130, Issue 3, pp.449-467, (2003)

11. Jiawei Zhang, and Yinyu Ye.: On the Budgeted MAX-CUT problem and its Application to the Capacitated Two-Parallel Machine Scheduling. Working Paper, Department of Management Sciences, The University of Iowa, (2001)

12. Maximilian M. Etschmaier, and Dennis F. X. Mathaisel.: Airline Scheduling: An Overview. Transportation Science, Vol.19, No.2, May 1985, pp.127-138, (1985)

13. R.A. Rushmeier, K.L. Hoffman, and M. Padberg.: Recent Advances in Exact Optimization of Airline Scheduling Problems. Technical Report, George Mason University, (1995)