Journal of Mathematics and System Science 6 (2016) 60-65
doi: 10.17265/2159-5291/2016.02.002

Some Discussions on Parallel Bounded Batch Scheduling to Minimize the Sum of Squared Machine Loads

Zengxia Cai, Xianzhao Zhang*
College of Science, Linyi University, Linyi 276005, PR China.

Received: October 04, 2015 / Accepted: November 01, 2015 / Published: February 25, 2016.

Abstract: We study the problem of scheduling n jobs on m parallel bounded batch machines to minimize the sum of squared machine loads. Each batch contains at most B jobs, and the processing time of a batch is equal to the longest processing time of the jobs in this batch. We prove this problem to be NP-hard. Furthermore, we present a polynomial time approximation scheme (PTAS) and a fully polynomial time approximation scheme (FPTAS) for this problem.

Keywords: Scheduling; Parallel batch; Polynomial time approximation scheme; FPTAS

1. Introduction

A batch machine is a machine that can process up to B jobs simultaneously as a batch. The research on this model is motivated by burn-in operations in semiconductor manufacturing (Lee et al. (1992)). There are two variants of the burn-in model: the unbounded model, in which there is no upper bound on the number of jobs that may be processed in the same batch; and the bounded model, in which at most $B < n$ jobs can be processed in the same batch.

In this paper we study the following problem: We are given a job set $J = \{J_1, J_2, \cdots, J_n\}$. Each job J_j has a processing time $p_j (j = 1, 2, \cdots, n)$ (p_j is a positive integer), which specifies the minimum time needed to process this job. All jobs are available at time 0. We are also given m identical parallel batch machines $M_1, M_2, \cdots, M_m$. At most $B < n$ jobs can be processed at one time on each machine. The jobs processed together form a batch, and the processing timeof a batch is equal to the longest processing time of the jobs among this batch. Preemption is not allowed. That is, once the processing of a batch starts on a machine, this machine is occupied until this process is completed. The load L_i for machine $M_i (i = 1, 2, \cdots, m)$ is defined to be the sum of the processing time of the batches assigned to it. The goal is to minimize the sum of squared machine loads. This problem arises e.g. when placing a set of records on a sectored drum so as to minimize the average latency time (Cody and Coffman (1976)) and in other storage allocation problems (Chandra and Wong (1975)). Following the notation of Graham et al. (1979), this problem may be denoted by $P_m \mid p-batch, B < n \mid \sum_{i=1}^{m} L_i^2$.

Related work: Many results appear concerning the problem discussed in this paper. Chandra and Wong (1975) analyzed the LPT rule for problem $P_m \| \sum L_i^2$ and proved a performance guarantee of $\frac{25}{24}$. This result was slightly improved by Leung and Wei (1995). Avidor et al. (2001) showed that LS has performance guarantee $\frac{4}{3}$. As for a more general problem $P_m \| \sum L_i^p$ (which is equivalent to minimize

Corresponding author: Xianzhao Zhang, College of Science, Linyi University, Linyi 276005, PR China.

the l_p-norm), Chandra and Wong (1975) showed that for every fixed $p \geq 1$, the LPT rule achieved a constant approximation ratio whose value depends on p and might be as large as $\frac{3}{2}$.

Our contributions: We prove that the problem $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$ is binary NP-hard. Moreover, we put forward a PTAS and an FPTAS for this problem. Taking into account its complexity, we declare that problem $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$ has been solved completely.

This paper is organized as follows: In section 2 we provide an NP-hardness for problem $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$. In section 3 a polynomial time approximation scheme (PTAS) is presented for this problem. In section 4 we propose an FPTAS for this problem. In section 5 we give some concluding remarks.

2. An NP-hardness Proof

In this section, we solve the complexity for problem $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$. First we give a simple property for the optimal schedule.

Lemma 1. We denote by π^* the optimal schedule to $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$, then the batches in π^* obey the FBLPT rule.

Proof: For a feasible schedule π violating the FBLPT rule, we may adjust it by simple exchanges without increasing the objective value. We omit the details.

Theorem 1. Problem $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$ is binary NP-hard.

Proof: Our proof proceeds by a reduction from the binary NP-complete problem PARTITION.

PARTITION

Given a set $\{a_1, a_2, \cdots, a_n\}$ of n positive integers, is it possible to partition the index set $\{1,2,\cdots,n\}$ into two disjoint subsets X and Y such that $\sum_{j \in X} a_j = A$, where $A = \frac{\sum_{j=1}^{n} a_j}{2}$.

Given an instance of PARTITION, we construct an instance of $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$ as follows: The job set consists of $n+1$ different job types $1,2,\cdots,n+1$. Type $i(1 \leq i \leq n)$ is composed of B jobs $\{J_{i,1}, ots, J_{i,B}\}$ with processing time a_i. There are $(m-2)B$ jobs $\{J_{n+1,1}, \cdots, J_{n+1,(m-2)B}\}$ with processing time A in type $n+1$.

Obviously this instance can be constructed in polynomial time.

In the remainder of the proof, we show that PARTITION has a solution if and only if there is a schedule for the corresponding instance of the scheduling problem with $\sum_{i=1}^{m} L_i^2 \leq mA^2$.

First, suppose that X and Y define a solution to PARTITION. Consider a schedule with $n+m-2$ batches that is formed in the following way. The B jobs in type $i(1 \leq i \leq n)$ form a batch, and the $(m-2)B$ jobs in type $n+1$ are divided into $m-2$ batches, each containing exactly B jobs. Each batch from type $n+1$ is assigned to one machine, and there are only two machines (M_1 and M_2) unoccupied. The batches from type $i(1 \leq i \leq n)$ with $i \in X$ are assigned to machine M_1, while the batches remaining are assigned to M_2. Clearly we have found a schedule with objective value $\leq mA^2$.

Conversely, suppose that there exists a schedule with $\sum_{i=1}^{m} L_i^2 \leq mA^2$, the optimal value for the scheduling instance must be no greater than mA^2. We denote by π^* the optimal schedule for the instance. From Lemma 1, the B jobs in type $i(1 \leq i \leq n)$ form a batch, and the $(m-2)B$ jobs in type $n+1$ are divided into $(m-2)$ batches in π^*. If we regard each batch as one job, we obtain a

feasible schedule π^1 for $P_m \| \sum_{i=1}^{m} L_i^2$ with objective value $\le mA^2$ from π^*. Here the job set is $\{J_1, J_2, \cdots, J_{n+m-2}\}$, where $p_i = a_i (1 \le i \le n)$, and $p_{n+1} = p_{n+2} = \cdots = p_{n+m-2} = A$. We denote by L_{i1} the load for machine M_i in π^1.

Note that $\sum_{i=1}^{m} L_{i1} = \sum_{i=1}^{n+m-2} p_i = mA$, on the other hand, we have $\sum_{i=1}^{m} L_i^2 \ge mA^2$ (the relationship between arithmetic and geometric averages).

Hence we get $\sum_{i=1}^{m} L_i^2 = mA^2$. This implies $L_{11} = L_{21} = \cdots = L_{m1} = A$. The index set for jobs $J_i (1 \le i \le n)$ that are assigned to the same machine is X, and $\{1, 2, \cdots, n\} \setminus X = Y$. Therefore, $\sum s_{j \in X} a_j = A$, which shows that X and Y define a solution to PARTITION.

3. A PTAS for $P_m \mid p\text{-}batch, B < n \mid \sum_{i=1}^{m} L_i^2$

In this section, we propose a polynomial time approximation scheme for $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$. Due to the close relationship between $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$ and $P_m \| \sum_{i=1}^{m} L_i^2$, we first give a PTAS for $P_m \| \sum_{i=1}^{m} L_i^2$.

We denote by $p_{sum} = \sum_{j=1}^{n} p_j$ the overall job processing time and by $p_{max} = \max_{1 \le j \le n} \{p_j\}$ the length of the longest job. The objective value of an optimal schedule is denoted by OPT. It is easy to see that for $K = \max\{p_{max}^2, p_{sum}^2 / m\}$, $OPT \ge K$.

In the following we transform an arbitrary instance I of $P_m \| \sum_{i=1}^{m} L_i^2$ into a simplified instance I^1. First the jobs in I are classified into big jobs and small ones. The classification depends on a precision parameter $\varepsilon (0 < \varepsilon < 1)$.

Job J_j is called big if it has processing time $p_j > \varepsilon\sqrt{K}$, else it is called a small one.

The instance I^1 contains all the big jobs from instance I. Let S denote the sum of the processing times over all small jobs in I. Then I^1 contains $\lfloor S/(\varepsilon\sqrt{K}) \rfloor$ jobs of length $\varepsilon\sqrt{K}$ (Intuitively speaking, the small jobs in I are first merged into a long job of length S, and then this long job is cut into lots of chunks of length $\varepsilon\sqrt{K}$. If the last chunk is strictly smaller than $\varepsilon\sqrt{K}$, then we simply disregard it).

We declare that the optimal value OPT^1 for instance I^1 is fairly close to the optimal value OPT of instance I.

To prove this, let $S_i (1 \le i \le m)$ be the total size of small jobs on machine M_i in an optimal schedule for I. We denote by L_i^* the load of machine M_i in an optimal schedule for instance I. We adjust the optimal schedule for instance I as follows: On machine M_i, leave every big job where it is, and replace the small jobs by $\lfloor S_i/(\varepsilon\sqrt{K}) \rfloor$ chunks of length $\varepsilon\sqrt{K}$. By assigning the chunks we increase the load of machine M_i by at most

$$\lceil S_i/(\varepsilon\sqrt{K}) \rceil \varepsilon\sqrt{K} - S_i \le (S_i/(\varepsilon\sqrt{K}) + 1)\varepsilon\sqrt{K} - S_i \le \varepsilon\sqrt{K}.$$

Since

$$\lceil S_1/(\varepsilon\sqrt{K}) \rceil + \cdots + \lceil S_m/(\varepsilon\sqrt{K}) \rceil \ge S/(\varepsilon\sqrt{K}) \ge \lfloor S/(\varepsilon\sqrt{K}) \rfloor,$$

the resulting schedule π (or part of it) is a feasible schedule for instance I^1. We have

$$\sum_{i=1}^{m} L_i^2(\pi) \le \sum_{i=1}^{m} (L_i^* + \varepsilon\sqrt{K})^2 = \sum_{i=1}^{m} (L_i^*)^2 + 2\varepsilon\sqrt{K}\sum_{i=1}^{m} L_i^* + m\varepsilon^2 K \le (1 + 2\sqrt{m}\varepsilon + m\varepsilon^2) OPT$$

We conclude that $OPT^1 \le (1 + 2\sqrt{m}\varepsilon + m\varepsilon^2) OPT$.

It is more easier to solve instance I^1. The total processing time of jobs in I^1 is no more than p_{sum}, and each job in I^1 has length at least $\varepsilon\sqrt{K}$, the number of jobs in I^1 is bounded above by $p_{sum}/(\varepsilon tK) \le \sqrt{m}/\varepsilon$. We may find the optimal solution to I^1 by enumeration. The running time is $O(m^{\sqrt{m}/\varepsilon})$.

The remaining problem is to transform the solution back. Consider an optimal schedule σ^1 for the simplified instance I^1. For $i=1,2,\cdots,m$ we denote by $L_{i,1}$ the load of machine M_i in σ^1, by $B_{i,1}$ the total size ofbig jobs on M_i, and by $S_{i,1}$ the total size of the chunks with length $\varepsilon\sqrt{K}$. Obviously we have $L_{i,1} = B_{i,1} + S_{i,1}$.

We construct the following schedule σ for I: Each big job is placed on the same machine as in σ^1. We reserve an interval of length $S_{i,1} + 2\varepsilon\sqrt{K}$ on machine M_i. We then greedily put the small jobs into these reserved intervals: First we start packing small jobs into the reserved interval on M_i, until we meet some small job that does not fit in any more. We then turn to machine M_2, and so on.

Since the size of a small job is at most $\varepsilon\sqrt{K}$, the total size of the packed small jobs on machine $M_i(1 \le i \le m)$ is at least $S_{i,1} + \varepsilon\sqrt{K}$. From $\sum_{i=1}^{m} S_{i,1} > S - \varepsilon\sqrt{K}$, we can pack all the small jobs into these reserved intervals.

The load of machine M_i in σ is at most $L_{i,1} + 2\varepsilon\sqrt{K}$, so we have

$$\sum_{i=1}^{m} L_i^2(\sigma) \le \sum_{i=1}^{m} (L_{i,1} + 2\varepsilon\sqrt{K})^2 =$$

$$OPT^1 + 4\varepsilon\sqrt{K}\sum_{i=1}^{m} L_{i,1} + 4\varepsilon^2 Km \le \cdot$$

$$OPT^1 + (4\varepsilon\sqrt{m} + 4\varepsilon^2 m)K \le$$

$$(1 + 6\varepsilon\sqrt{m} + 5o n^2 m)OPT$$

The objective value of σ is within a $1+O(\varepsilon)$ factor of the optimal objective value. We have reached the desired PTAS for $P_m \| \sum_{i=1}^{m} L_i^2$.

In summary, the PTAS for $P_m \| \sum_{i=1}^{m} L_i^2$ may be stated as follows:

Algorithm 1

Step 1: Transform an instance I of $P_m \| \sum_{i=1}^{m} L_i^2$ into a simplified instance I^1.

Step 2: Obtain the optimal schedule for I^1 by enumeration.

Step 3: Treat the optimal schedule for I^1 by greedy algorithm and output a schedule for I.

From Lemma 1, we obtain a PTAS for problem $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$.

Algorithm 2

Step 1: Apply the FBLPT rule to an instance I of $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$. Regard each batch as a job and obtain an instance I_1 of $P_m \| \sum_{i=1}^{m} L_i^2$.

Step 2: Apply Algorithm 1 to the instance I_1 and output a schedule for I_1.

The running time of Algorithm 2 is $O(nlogn + m^{\sqrt{m}/\varepsilon})$.

4. An FPTAS for $P_m \mid p\text{-}batch, B<n \mid \sum_{i=1}^{m} L_i^2$

Due to the close relationship between $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$ and $P_m \| \sum_{i=1}^{m} L_i^2$, we first give an FPTAS for the latter.

In an instance I of $P_m \| \sum_{i=1}^{m} L_i^2$, there are n jobs $J_j(j=1,2,\cdots,n)$ with processing time p_j, and the goal is to find a schedule that minimizes the sum of squared machine loads. Denote by $p_{sum} = \sum_{j=1}^{n} p_j$. Observe that the size $|I|$ of the input I satisfies $|I| \ge \log(p_{sum}) = const.\ln(p_{sum})$.

We encode a feasible schedule σ with load L_i for machine M_i by a m-dimensional vector

$[L_1, L_2, \cdots, L_m]$. We first give a pseudo-polynomial time algorithm for problem $P_m \| \sum_{i=1}^{m} L_i^2$.

Algorithm A

Initialization: Set

$VS_1 = \{[p_1, 0, \cdots, 0], [0, p_1, \cdots, 0], \cdots, [0, 0, \cdots, p_1]\}$.

Phase *k*. For every vector $[x_1, x_2, \cdots, x_m]$ in VS_{k-1}, put the m vectors

$$[x_1 + p_k, x_2, \cdots, x_m], [x_1, x_2 + p_k, \cdots, x_m], \cdots, [x_1, x_2, \cdots, x_m + p_k]$$

in VS_k.

Output. Output the vector $[x_1, x_2, \cdots, x_m] \in VS_n$ that minimizes the value $\sum_{i=1}^{m} x_i^2$.

Since the coordinates of all vectors in all sets VS_k are integers in the range from 0 to p_{sum}, the cardinality of every vector set VS_k is bounded from above by $O(p_{sum}^m)$. Since the time complexity of the algorithm is proportional to $\sum_{k=1}^{n} |VS_k|$, the algorithm has a pseudo-polynomial time complexity of $O(np_{sum}^m)$.

To reduce the running time for the algorithm, we take the following steps: Each feasible schedule corresponds to a geometric point in the *m*-dimensional geometric body $[0, p_{sum}] \times [0, p_{sum}] \times \cdots \times [0, p_{sum}]$. We subdivide this geometric body with cuts into lots of boxes. In each direction these cuts are made at the coordinates Δ^i for $i = 1, 2, \cdots, L$ where $\Delta = 1 + \frac{\varepsilon}{2n}$, and

$$L = \lceil \log_{\Delta}(p_{sum}) \rceil = \lceil \ln(p_{sum}) / \ln(\Delta) \rceil \le \lceil (1 + \tfrac{2n}{\varepsilon}) \ln(p_{sum}) \rceil.$$

The last inequality holds since for all $z \ge 1$ we have $\ln z \ge (z-1)/z$ (which may be seen from the Taylor expansion of $\ln z$).

Note that for any two vectors $[x_1, \cdots, x_m]$ and $[y_1, \cdots, y_m]$ falling into the same box, their coordinates satisfy $x_i / \Delta \le y_i \le x_i \Delta$ $(i = 1, 2, \cdots, m)$.

We now give the main idea for the trimmed algorithm: Out of every box that has nonempty intersection with VS_k we select a single vector and put it into the so-called trimmed vector set VS_k^*. All remaining vectors from the vector set VS_k that have not been selected are lost for the further computations. And in phase $k+1$, the so-called trimmed algorithm generates its new vector from the smaller set VS_k^*, and not from the set VS_k.

What is the time complexity of the trimmed algorithm? The trimmed vector set VS_k^* contains at most one vector from each box in the subdivision. Altogether there are $O(L^m)$ boxes, the cardinality of VS_k^* is polynomial in the input size $|I|$ and also polynomial in $1/\varepsilon$. And since the time complexity of the trimmed algorithm is proportional to $\sum_{k=1}^{n} |VS_k^*|$, the trimmed algorithm has a time complexity that is polynomial in the input size and in $1/\varepsilon$.

We can prove by induction that for each vector $[x_1, \cdots, x_m] \in VS_k$, there is a vector $[x_1^*, \cdots, x_m^*] \in VS_k^*$ whose coordinates are at most a factor of Δ^k above the corresponding coordinates of $[x_1, \cdots, x_m]$. (The coordinates of the new vectors are non-negative linear combinations of the coordinates of the old vectors, and it is the crucial property that makes the inductive argument go through.)

We claim that the trimmed algorithm outputs a near-optimal schedule for problem $P_m \| \sum_{i=1}^{m} L_i^2$. At the end of its execution, the untrimmed algorithm outputs a vector $[x_1, \cdots, x_m] \in VS_n$ that minimizes the value $m_{i=1}^{m} x_i^2$. By the above argument, there exists a $[x_1^*, \cdots, x_m^*] \in VS_n^*$ satisfying $x_i^* \le \Delta^n x_i = (1 + \frac{\varepsilon}{2n})^n x_i$ $\le (1+\varepsilon) x_i (i = 1, 2, \cdots, m)$.

The last inequality holds from the well-known inequality $(1 + z/n)^n \le (1 + 2z)$ for $0 \le z \le 1$.

So we have $\sum_{i=1}^{m} (x_i^*)^2 \le (1+\varepsilon)^2 \sum_{i=1}^{m} x_i^2 \le (1+3\varepsilon) OPT$.

We are ready to give an FPTAS for problem $P_m \| \sum_{i=1}^{m} L_i^2$.

Trimmed Algorithm B

Step 1: Set $VS_0^* = \{[0,0,\cdots,0]\}$.

Step 2: For $k=1$ to n do

Step 3: Let $VS_k = \varnothing$

Step 4: For every $[x_1,\cdots,x_m] \in VS_{k-1}^*$ do

Step 5: Put the m vectors $[x_1+p_k, x_2,\cdots,x_m]$, $\cdots,[x_1,x_2,\cdots,x_m+p_k]$ in VS_k.

Step 6: Endfor

Step 7: Compute a trimmed copy VS_k^* of VS_k

Step 8: Endfor

Step 9: Output the vector $[x_1,x_2,\cdots,x_m] \in VS_n^*$ that minimizes the value $\sum_{i=1}^{m} x_i^2$.

Following we give an FPTAS for $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$.

Algorithm Final

Step 1: Apply the FBLPT rule to an instance I of $P_m \mid p-batch, B<n \mid \sum_{i=1}^{m} L_i^2$. Regard each batch as a job and obtain an instance I_1 of $P_m \| \sum_{i=1}^{m} L_i^2$.

Step 2: Apply Trimmed Algorithm B to the instance I_1 and output a schedule for I_1.

5. Concluding Remarks

In this paper we study the problem of scheduling n jobs on m parallel bounded batch machines to minimize the sum of squared machine loads. We prove this problem to be NP-hard. Furthermore, we present a polynomial time approximation scheme (PTAS) and afully polynomial time approximation scheme (FPTAS) for this problem. From this sense we declare that this problem has been solved completely.

References

[1] Avidor, A., Azar, Y., Sgall, J., Ancient and new algorithms for load balancing in the l_p norm, Algorithmica 29 (2001) 422-441.

[2] Chandra, A.K., Wong, C.K., Worst-case analysis of a placement algorithm related to storage allocation, SIAM Journal on Computing 4 (1975) 249-263.

[3] Cody, R.A., Coffman, E.G., Record allocation for minimizing expected retrieval costs on drum-like storage devices, Journal of ACM 23 (1976) 103-115.

[4] Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Optimization and approximation in deterministic sequencing and scheduling: A survey, Annals of Discrete Mathematics 5 (1979) 287-326.

[5] Lee, C.Y., Uzsoy, R., Martin Vega, L.A., Efficient algorithms for scheduling semiconductor burn-in operations, Operations Research 40 (1992) 764-775.

[6] Leung, J.Y.T., Wei, W.D., Tighter bounds on a heuristic for a partition problem, Information Processing Letters 56 (1995) 51-57.