Duplicating and its Applications in Batch Scheduling

Yuzhong Zhang[1] Chunsong Bai[1] Shouyang Wang[2]

[1] College of Operations Research and Management Sciences
Qufu Normal University, Shandong 276826, China
[2] Academy of Mathematics and Systems Science, CAS, Beijing 100080, China

Abstract *In this paper, we firstly describe a technique named* **Duplicating**, *which duplicates machines in batch scheduling environment. Then we discuss several applications of* **Duplicating** *in solving batch scheduling problems, for the batch scheduling problem on unrelated parallel machines to minimize makespan, we give an* $(4-\frac{2}{B})$*-approximation algorithm and an* $(2-\frac{1}{B}+\epsilon)$ *algorithm when* m *is fixed. We also present an* $4(2-\frac{1}{B}+\epsilon)$*-competitive algorithm for the on-line scheduling problem on identical machines to minimize total weighted completion time.*

Keywords Duplicating, Integer programming, Batch processing, Parallel machines

1 Introduction

Model. Scheduling a batch processing system has been extensively studied in the last decade. In batch processing system, each job j of a given instance has a positive processing time $p_j > 0$ and a nonnegative weight $w_j \geq 0$. Each of the m machines can simultaneously process B jobs of the n jobs as a batch. The processing time of a batch is given by the longest job in the batch, and the weight of the batch is the sum of job weight in the batch. No preemption is allowed. Our goal is to assign all n jobs to batches and determine the sequence of these batches in such a way that the objective function C_{max} or $\sum w_j C_j$ is minimized.

Related Work. Regarding to scheduling problem of batch processing machine, Ikura and Gimple are the first researchers, they proposed an $O(n^2)$ algorithm to find a due date feasible schedule minimizing the makespan under the assumption that release times and due dates are agreeable and that the processing times of all jobs are the same(see [11]). Glassey and Weng(1991) provide a dynamic heuristic for single batch processing machine problem with the consideration of future job arrivals(see [9]). In [8], Dobson et al. consider the problem of minimizing total weighted completion time on single batch processing machine by modelling an integer programming formulation.

As to minimize the number of tardy jobs and maximum tardiness and to minimize the makespan on parallel identical batch processing machines, Lee et al.(1992) provide efficient algorithms under some assumptions, for some heuristics they give worst case performance analysis(see [15]). Later they improve their algorithms to solve problems $1|r_j, B|T_{max}$ and $1|r_j, B|\sum U_j$ where releasing times, processing times and due dates are all agreeable. Furthermore, they prove that, when releasing times and due dates are agreeable and all jobs have identical processing time, these problems are NP-hard(see [17]). Under the same assumptions in [17], Chandru et al.(1993) provide the optimal and a heuristic algorithm for the problems to minimize total completion times(see [2]), they solve the problem polynomially in the case where for single machine there are only a fixed number of job families while jobs in the same family have the same processing time(see [3]).

Uzsoy(1994, 1995) considers a number of problems involving single and parallel batch processing machines. He prove that, when jobs have different capacity requirements, minimizing $\sum C_j$ and C_{max} are both NP-hard. He provides a series of heuristics(see [20]). Having partitioned all jobs into some disjoint families, where jobs in the same family have the same processing times, while jobs from different families can not be processed in the same batch, he gives some optimal and heuristic algorithms for some batch scheduling problems(see [21]).

Even in the case of the capacity of the stove is large enough, i.e. $B = \infty$, Deng and Zhang proved that the problem to minimize the total of the completion times on single batch machine, jobs release over time, is also NP-hard. They design polynomial and pseudopolynomial time algorithms in different cases respectively. For the problem of its on-line case they propose a 1.618-competitive algorithm(see [7]).

Lee C.Y and Uzsoy R.(1996) address the problem of minimizing makespan on single batch processing machine in the presence of dynamic job arrivals. They solve the problems polynomially for several special cases, one is that the capacity of machine is infinite, i.e. $B = \infty$, another is that jobs must be processed in a predetermined sequence and the other two cases are release times and processing times are agreeable and disagreeable respectively(see [14]) .

They provide a pseudopolynomial time algorithm when there is only two distinct release times. For the general problem $1|r_j, B|C_{max}$, they present five heuristics and give computational experiments for some of them. At last Deng and Zhang ([7]) derive a $PTAS$ algorithm for the original problem.

As to minimize the total weighted completion time on single batch machine, Chen et al.[4] provided a $\frac{10}{3}$-competitive on-line algorithm for $1|B \geq n, r_j|\sum w_j C_j$, and an $(4+\epsilon)$-competitive on-line algorithm for $1|B < n, r_j|\sum w_j C_j$.

Our Contributions. In this paper, we firstly present a new technique named **Duplicating** and then consider its applications in batch scheduling problems. In section 2, we describe the framework **Duplicating**, in section 3 and 4, we deal with the scheduling problems $R|B|C_{max}$ and $P|$*on-line*$,B|\sum w_j C_j$, by using **Duplicating**.

2 Framework Duplicating

Now we describe the general framework **Duplicating** which is presented for the first time. We denote $\mathbf{\Pi}$ the batch scheduling problem we are going to deal with, and $\mathbf{\Pi}'$ the scheduling problem which has the same scheduling environment like $\mathbf{\Pi}$ but no batch. For dealing with batch scheduling problem $\mathbf{\Pi}$, we construct problem $\mathbf{\Pi}'$ by duplicating B identical machines for each machine in problem $\mathbf{\Pi}$. By using algorithm for problem$\mathbf{\Pi}'$, we can get a schedule, then we batch and schedule jobs to be scheduled on B same machines on the machine which duplicating the B same machines by some rules, and determine an feasible schedule for the problem $\mathbf{\Pi}$.

However, many efficient algorithms have been presented for the general scheduling problems by combining with **Duplicating**.We will do them in the paper.

3 Algorithm for $R|B|C_{max}$

In this section, we will design an approximation algorithm for problem $R|B|C_{max}$ by combining **FBLPT** and the approximation algorithm for $R||C_{max}$ derived in [16], give an $(4 - \frac{2}{B})$-approximation algorithm.

3.1 Terminology and Rounding Theorem

For the convenience of the statement, we would like to cite a basic algorithm — Full Batch Largest Processing Time (**FBLPT**) given by Bartholdi ([1]), which can solve $1|B|C_{max}$ problem optimally and we have proved this in [25].

Algorithm FBLPT

Step 1 Sort the jobs according to their processing times and assume that $p_1 \geq p_2 \geq \cdots \geq p_n$

Step 2 Form batches by placing jobs $iB + 1$ through $(i + 1)B$ together in the same batch for $i = 0, 1, \cdots, \lfloor n/B \rfloor$, where $\lfloor x \rfloor$ is the largest integer smaller than x

Step 3 Schedule these batches in arbitrary order

The schedule contains at most $\lfloor n/B \rfloor + 1$ batches and all batches are full except possibly the last one.

Now we introduce *Rounding Theorem* by Lenstra([16]). Let $J_i(t)$ denote the set of jobs that require no more than t time units on machine i, and let $M_j(t)$ denote the set of machines that can process job j no more than t time units. We next consider a decision version of a scheduling problem, where for each machine i there is a deadline d_i and where we are further constrained to schedule jobs such that each uses processing time at most t.

Theorem 3.1. *(Rounding Theorem) [16] Let* $P = (P_{ij}) \in \mathcal{F}_+^{m \times n}$,

$$\vec{d} = (d_1, \cdots, d_m) \in \mathcal{F}_+^m,$$

and $t \in \mathcal{F}_+$. *If the linear program* $LP(P, \vec{d}, t)$ *given by*

$$\sum_{i \in M_i(t)} x_{ij} = 1, j = 1, \cdots, n,$$

$$\sum_{j \in J_i(t)} P_{ij} x_{ij} \leq d_i, i = 1, \cdots, m,$$

$$x_{ij} \geq 0 \quad \textit{for } j \in J_i(t), i = 1, \cdots, m,$$

has a feasible solution, then any vertex $\tilde{x}$*of the polytope can be rounded to a feasible solution* $\tilde{x}$ *of the integer linear programming* $IP(P, \vec{d}, t)$, *given by*

$$\sum_{i \in M_i(t)} x_{ij} = 1, j = 1, \cdots, n$$

$$\sum_{j \in J_i(t)} P_{ij} x_{ij} \leq d_i + t, i = 1, \cdots, m$$

$$x_{ij} \in \{0, 1\} \quad \textit{for } j \in J_i(t), i = 1, \cdots, m,$$

and this rounding can be done in polynomial time.

A polynomial 2−approximation algorithm for the problem $R||C_{max}$ is derived by the theorem above. We denote the algorithm as **Rounding Algorithm** in this paper.

3.2 Polynomial Approximation Algorithm

We present the main approximation algorithm in the section, construct a schedule that is guaranteed to be no longer than 4 time the optimum by making use of **Rounding Algorithm** and **FBLPT** agilely.

Algorithm $\mathcal{A}$

Step 1. Duplicate B identical machines for each unrelated machine in our problem

Step 2. Determine jobs to be scheduled on each of mB machines by Rounding Algorithm

Step 3. Batch and schedule all jobs to be scheduled on B machines in Step 2 on the machine which duplicating B machines by algorithm **FBLPT**.

The algorithm is a polynomial hybrid, which combines **Rounding Algorithm** and **FBLPT**.

3.3 The Performance Guarantee of $\mathcal{A}$

We will analyze the performance guarantee of the algorithm above. Let $\mathcal{A}(I)$ and $OPT(I)$denote the makespans of the algorithm and the optimal schedule for an instance I of our problem $R|B|C_{max}$ respectively. We denote the optimal makespan for instance I on machines mB machines duplicated in the algorithm $\mathcal{A}$ as $OPT_{mB}(I)$.

Lemma 3.2.

$$OPT_{mB}(I) \leq OPT(I)$$

Proof. Let σ be the optimal schedule for the problem $R|B|C_{max}$. Lengthening the processing time of each job to the largest one in the same batch, we get a new instance I'. Obviously,

$$OPT(I) = OPT(I')$$

Assigning B jobs in the same batch on B machines duplicated, we obtain a schedule on mB machines for instance I'. The optimal makespan remains unchanged,

$$OPT(I') = OPT_{mB}(I')$$

which is larger or equal to $OPT_{mB}(I)$, i.e.

$$OPT_{mB}(I') \geq OPT_{mB}(I)$$

which implies the result. □

Let I_i denote all jobs for the instance I assigned on machine M_i by $\mathcal{A}$. $\mathcal{A}(I_i)$ denotes the completion time of machine M_i after scheduled by $\mathcal{A}$, and $OPT_{mB}(I_i)$ denote the makespans yielded by the optimal schedule on the machines duplicated by machine M_i.

Lemma 3.3.

$$\mathcal{A}(I_i) \leq (2 - \frac{1}{B})OPT_{mB}(I_i)$$

Proof. We suppose that the jobs assigned on machine M_i is grouped k batches and the batches are indexed by the non-increasing order in **FBLPT**. P^j is the largest processing time of the jobs in batch j. $P_{(l)j}$ is the processing time of job j in batch l. It is not hard to see that

$$\begin{aligned} B\mathcal{A}(I_i) = BC_i(\mathcal{A}) &= \sum_{j=1}^{k} BP^j \\ &= \sum_{j \in I_i} P_j + BP^1 - \sum_j P_{(1)j} + BP^2 - \sum_j P_{(2)j} + \cdots + BP^k - \sum_j P_{(k)j}. \end{aligned}$$

Therefore

$$\begin{aligned} B\mathcal{A}(I_i) &\leq \sum_{i,j} P_{(i)j} + (B-1)(P^1 - P^2) + (B-1)(P^2 - P^3) + \cdots + (B-1)P^k \\ &= \sum_{i,j} P_{(i)j} + (B-1)P^1, \end{aligned}$$

then

$$\mathcal{A}(I_i) \leq \frac{\sum_{i,j} P_{(i)j}}{B} + (1 - \frac{1}{B})P^1 \leq (2 - \frac{1}{B})OPT_{mB}(I_i).$$ □

Lemma 3.4.

$$OPT_{mB}(I_i) \leq \mathcal{A}(I_i)$$

for each i.

Proof. This is trivial. □

Lemma 3.5.

$$\mathcal{A}(I) \leq (2 - \frac{1}{B})A_{mB}(I)$$

Proof. Suppose that the makespan corresponding to the algorithm $\mathcal{A}$ is reached on machine M_i. Consider the B machines duplicated by the machine M_i,

$$\mathcal{A}(I) = \mathcal{A}(I_i) \leq (2 - \frac{1}{B})OPT_{mB}(I_i) \leq (2 - \frac{1}{B})\mathcal{A}_{mB}(I)$$

□

Theorem 3.6.

$$\frac{\mathcal{A}(I)}{OPT(I)} \leq 4 - \frac{2}{B}$$

Proof. By lemma 2 and lemma 5 we get

$$\frac{\mathcal{A}(I)}{OPT(I)} \leq \frac{\mathcal{A}(I)}{OPT_{mB}(I)} \leq (2 - \frac{1}{B})\frac{\mathcal{A}_{mB}(I)}{OPT_{mB}(I)}$$

At last, noticing the worst case performance of the **Rounding Algorithm**, we complete the proof. □

However, there is an FPTAS for problem $Rm||C_{max}$ presented by Jansen and Porkolab([12]) in 2000, hence, by Theorem 6, we get an improved $(2 - \frac{1}{B} + \epsilon)$-approximation for the problem when m is fixed.

4 Algorithm for $P|online, B|\sum w_j C_j$

In this section, we consider the on-line batch scheduling problem $P|B|\sum w_j C_j$. Our algorithm is motivated by a general on-line framework of Hall et al.([13]), by using **Duplicating** , we give an $4(2 - \frac{1}{B} + \epsilon)$-competitive on-line algorithm for this problem.

4.1 Framework Greedy-Interval (see [13])

The framework **Greedy-Interval** is based on partitioning the time horizon of possible completion time at geometrically increasing points. Let τ_0=1 and $\tau_i = 2^{i-1}$. The algorithm constructs the schedule iteratively: in iteration $i = 1, 2, \ldots,$ we wait until time τ_i, and then focus on the set of jobs that have been released by this time, but not yet scheduled, which we denote by J_i. We invoke the dual ρ-approximation algorithm for the set of jobs J_i and these jobs are scheduled to run from time $\rho\tau_i$ to $\rho\tau_{i+1}$. According to Hall et al.(1997), we have the following.

Lemma 4.1. *Given a dual ρ-approximation algorithm for the maximum scheduled weight problem(MSWP), the framework* **Greedy-Interval** *yields an on-line 4ρ-approximation algorithm to minimize the total weighted completion time.*

For the on-line problem $P||\sum w_jC_j$, there is an important result by Hall et al.(1997), which we will use later.

Lemma 4.2. *There is a dual $(1+\epsilon)$-approximation algorithm for the maximum scheduled weight problem(MSWP) in identical parallel machine scheduling environment.*

4.2 An Algorithm for MSWP

Now we design an approximation algorithm for MSWP in identical batch machine scheduling environment, determine a subset of J_i(and an associated schedule) that can be batched and scheduled on m identical parallel machines to complete by time $(2-\frac{1}{B}+\epsilon)D$, whose total weight is *superoptimal*, that is , at least as large as that of any subset of jobs that can be scheduled to complete by time D.
Algorithm $\mathcal{A}'$

Step 1. Duplicate B identical machines for each machine in our problem

Step 2. Determine jobs to be scheduled on each of mB machines by the algorithm for MSWP of $P|online|\sum w_jC_j$

Step 3. Batch and schedule all jobs to be scheduled on B machines in Step 2 on the machine which duplicating B machines by algorithm **FBLPT**.

Like the algorithm $\mathcal{A}$, we construct algorithm $\mathcal{A}'$ by using **Duplicating**. It is obvious that the algorithm is polynomial in the size of the input.

4.3 Analysis of the Algorithm

Let $w(S^*_{mB})$ and $w(S^*_m)$ be the optimal value for the MSWP of $P_{mB}|r_j|\sum w_jC_j$ and $P_m|r_j, B|\sum w_jC_j$ respectively, where $w(S) = \sum_{j\in S} w_j$ for each subset $S \subseteq J_i$. In fact, the subsets S^*_{mB} and S^*_m of J_i are the optimal solution for the two problems respectively.

Lemma 4.3.

$$w(S^*_{mB}) \geq w(S^*_m)$$

Proof. For the optimal job set S^*_m, we assign B jobs in the same batch on B machines duplicated, and obtain a schedule on mB machines for the job set S^*_m. Obviously, these jobs can be completed by D, the total weight of these jobs remains unchanged, means that any optimal set S^*_m is a feasible set for the MSWP of $P_{mB}|r_j|\sum w_jC_j$, which implies the result. □

Let S denote all jobs of J_i assigned on mB machines by $\mathcal{A}'$, we denote the optimal makespan as $OPT_{mB}(S)$, by Lemma 4.2, we know $OPT_{mB}(S) \leq (1+\epsilon)D$. Let $\mathcal{A}'(S)$ and $OPT(S)$ denote the makespan of the algorithm and the optimal value for the jobs set S of our problem MSWP respectively. If we use notation as in section 3, by Lemma 3.3, we have the following result.

Lemma 4.4.

$$\mathcal{A}'(S_i) \leq (2-\frac{1}{B})OPT_{mB}(S_i)$$

Theorem 4.5. *There is a dual $(2-\frac{1}{B}+\epsilon)$-approximation algorithm for the maximum scheduled weight problem(MSWP) in identical batch machine scheduling environment.*

Proof. By Lemma 4.3, we know that the total weight of S is superoptimal. Suppose that the makespan $\mathcal{A}'(S)$ is reached on machine M_i, together with Lemma4.2, we have

$$\mathcal{A}'(S) = \mathcal{A}'(S_i) \leq (2-\tfrac{1}{B})OPT_{mB}(S_i)$$

$$\leq (2-\tfrac{1}{B})(1+\epsilon)D.$$

Therefore, S is a dual $(2-\frac{1}{B}+\epsilon)$-approximation solution. □

From Theorem 4.5, we know that there is a dual $(2-\frac{1}{B}+\epsilon)$-approximation algorithm, together with on-line framework **Greedy-Interval** yields an $4(2-\frac{1}{B}+\epsilon)$-competitive algorithm for our problem.

5 Conclusion

In this paper, we presented framework **Duplicating** for scheduling on parallel batch machines and constructed two approximation algorithms by using the method. However, this method maybe useful for other problems, such as, when scheduling on unrelated batch machines, the method lead to a $8(2-\frac{1}{B})$-competitive algorithm.

References

[1] Bartholdi J.J, Unpublished, 1988.

[2] Chandru V., Lee C.Y. and Uzsoy R., 1993, Minimizing total completion time on batch processing machines, *International Journal of Production Research*, **31**, 2097-2121.

[3] Chandru V., Lee C.Y. and Uzsoy R., 1993, Minimizing total completion time on a batch processing machine with job families, *Operations Research Letters*, **13**, 61-65.

[4] Chen B., Deng X. and Zang W., 2004, On-line scheduling a batch processing system to minimize total weighted job completion time, *Journal of Combinatorial Optimization*, **8**, 85-95.

[5] Cheng T.C.E. and Gordon V.S., 1994, Batch delivery scheduling on single machine, *Journal of the Operational Research Society*, **45**, 1211-1215.

[6] Deng X., Poon C.K. and Zhang Y., 1999, Approximation algorithms in batch processing, *Lecture Notes in Computer Science*, **1741**, 153-162.

[7] Deng X. and Zhang Y., 1999, Minimizing mean response time in batch processing system, *Lecture Notes in Computer Science*, **1627**.

[8] Dobson G. and Nambinadom R.S., 2001, The batch loading and scheduling problem, *Operations Research*, **49**, 52-65.

[9] Glassey C.R. and Weng W.W., 1991, Dynamic batching heuristics for simultaneous processing, *IEEE Transactions on Semiconductor Manufacturing*, **4**, 77-82.

[10] Graham R.L., Lawler, Lenstra J.K. and Rinnooy Kan, A.H.G., 1979, Optimization and approximation in deterministic sequencing and scheduling, *Annals of Discrete Mathematics*, **5**, 387-326.

[11] Ikura Y. and Gimple M., 1986, Scheduling algorithm for a single batch processing machine, *Operations Research Letters*, **5**, 61-65.

[12] Klaus Jansen and Lorant Porkolab, 2001, Improved approximation schemes for scheduling unrelated parallel machines, *Mathematics of operations research*, **26**, 324-338.

[13] L.A. Hall, A.S.Schulz, D.B. Shmoys, and J. Wein, 1997, Scheduling to minimize average completion time: off-line and on-line approximation algorithms, *Mathematics of Operations Research*, **22**, 513-544.

[14] Lee C.Y., and Uzsoy R., Minimizing makespan on a single batch processing machine with dynamic job arrivals, *Research Report, *Department of Industrial and System Engineering, University of Florida, January*, 1996.

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

[16] Lenstra J.K.,Shmoys D.B. and Tardos E., 1990, Approximation algorithms for unrelated parallel machines, *Mathematical Programming*, **46**, 259-271.

[17] Li C.L. and Lee C.Y., 1997, Scheduling with agreeable release times and due dates on a batch processing machine, *European Journal of Operations Research*, **96**, 564-569.

[18] Papadimitriou C.H. and Steiglitz K., Combinatorial optimization: Algorithms and complexity. Prentice-Hall, Inc., New Jersey, 1982.

[19] S. Sahni, 1976, Algorithms for scheduling independent tasks, *Journal of the association for computing machinery*, **23**, 116-127.

[20] Uzsoy R., 1994, Scheduling a single batch processing machine with nonidentical job sizes, *International Journal of Production Research*, **32**, 1615-1635.

[21] Uzsoy R., 1995, Scheduling batch processing machines with incompatible job families, *International Journal of Production Research*, **33**, 2605-2708.

[22] Uzsoy R., Lee C.Y. and Martin Vega L.A., 1990, A review of production planning and scheduling models in the semiconductor industry. Part I: System characteristics performance evaluation and production planning, *IIE Transactions on Scheduling and Logistics*, **24**, 47-61.

[23] W.E. Smith, 1956, Various optimizers for single-stage production, *Naval Research and Logistics Quarterly*, **3**, 59-66.

[24] X.Deng, H.Feng, P.Zhang, Y.Zhang, 2004, Minimizing mean completion time in a batch processing system, *New York: Springer*, Algorithmica , 49-106.

[25] Y.Zhang, 1997, Scheduling problem with batch set-up, *Ph.D Thesis, Institute of Applied Mathematics, Chinese Academy of Sciences*.

[26] Y.Zhang, S. Wang, 2002,