Parallel Machine Batch Scheduling with Deadlines and Sequence Independent Set-Ups

Peter Brucker [1]
Universität Osnabrück, 49069 Osnabrück, Germany
Mikhail Y. Kovalyov [2]
Institute of Engineering Cybernetics, Surganova 6, 220012 Minsk, Belarus
Frank Werner [3]
Otto-von-Guericke Universität, Fakultät für Mathematik, PSF 4120, 39106 Magdeburg, Germany

Abstract: The problem of scheduling G groups of jobs on m unrelated parallel machines is considered. Each group consists of several identical jobs. Implicit in a schedule is a partition of a group into batches; all jobs of a batch are processed jointly. It is possible for different batches of the same group to be processed concurrently on different machines. However, at any time, a batch can be processed on at most one machine. A sequence independent set-up time is required immediately before a batch of a group is processed. A deadline is associated with each group. The objective is to find a schedule which is feasible with respect to deadlines. The problem is shown to be NP-hard even for the case of two identical machines, unit processing times, unit set-up times and a common deadline. It is strongly NP-hard if machines are uniform and processing times, set-up times and deadlines are unit. Special cases which are polynomially solvable are discussed. For the general problem, a dynamic programming algorithm and a family $\{DP_\varepsilon\}$ of approximation algorithms is constructed. For any $\varepsilon > 0$, algorithm DP_ε delivers a schedule in which the completion time of each group is at most $(1+\varepsilon)$ times the value

[1]Supported by Deutsche Forschungsgemeinschaft, Project JoBTAG, and by Alexander von Humboldt-Stiftung
[2]Supported by Deutsche Forschungsgemeinschaft, Project ScheMA
[3]Supported by Deutsche Forschungsgemeinschaft, Project ScheMA, and by Alexander von Humboldt-Stiftung

of its deadline if there exists a schedule which is feasible with respect to the deadlines. The time complexity of DP_ε is $O(G^{2m+1}/\varepsilon^{2m})$.
Key words: Parallel machine scheduling, batching, fully polynomial approximation scheme.

1 Introduction

We consider the following parallel machine batch scheduling problem. There are G groups of jobs to be scheduled on $m \geq 2$ unrelated parallel machines. All jobs are available for processing at time zero. Each group j has a deadline $d_j > 0$ and consists of $q_j > 0$ identical jobs. If a job a group j is processed on machine l then its processing time is $p_{jl} \geq 0$.
It is possible to partition the groups into batches of jobs; all jobs of a batch are processed jointly. Different batches of the same group can be processed concurrently on different machines. However, at any time, a batch cannot be processed on more than one machine. Machine set-ups are necessary as follows. A set-up time $t_{jl} \geq 0$ is needed immediately before a batch of group j is processed on machine l.
A schedule specifies batch sizes, indicates which batches are scheduled on which machine and defines a processing order for the relevant batches on each machine. The completion time C_j of group j is the completion time of the job processed last in this group. The objective is to find a schedule such that $C_j \leq d_j$ for all groups.
The relevant schedules can be restricted in the following way. If there are two or more batches of the same group on the same machine, then, without violating the feasibility, these batches may be combined to form a single batch on the machine. Thus, we may consider only schedules in which there is at most one batch for each group on the same machine. By the *Earliest Due Date (EDD)* rule of Jackson [3], batches on each machine may be sequenced in non-decreasing order of the deadlines. Thus, the problem reduces to finding a partition of each group into at most m batches processed on different machines and an assignment of these batches to the machines.
Grouping of similar jobs is done in many production environments where the principles of group technology [2] are adopted. Based on these principles, the factory layout is such that machines are grouped into cells. Each cell produces several groups of jobs with similar production requirements. For

jobs scheduled consecutively on the same machine, a machine set-up is only required if the jobs belong to different groups.
In connection with group technologies all jobs from the same group are scheduled contiguously. However, this is not necessarily the best strategy. For example, in a parallel machine environment, the creation of batches permits the overlapping of operations of the same group and may therefore reduce the throughput time. Preemptive schedules may also reduce throughput time but, in the literature on preemptive scheduling, there is no penalty when a job is preempted. In practice, it is rather unusual for a machine to be able to operate a job immediately after the processing of the previous job: it is more likely that some machine set-up time is incurred. This observation motivates the consideration of our parallel machine batch scheduling problem with set-up times.
For the batch scheduling problems, a review of algorithms and complexity was recently given by Potts and Van Wassenhove [8]. However, as far as we know our problem is not yet considered in the literature.
The remainder of the paper is organized as follows. In the next section we study the computational complexity of various special cases. We prove that our problem is NP-hard even for the case of two identical machines, unit processing times, unit set-up times and a common deadline (but a variable number of jobs for each group). If there are identical machines, a single job for each group, a common deadline and zero set-up times or zero processing times, then the problem is evidently equivalent to the known NP-complete problem MULTIPROCESSOR SCHEDULING [1]. We prove that the problem is strongly NP-hard if there is a variable number of uniform machines and all processing times, set-up times and deadlines are unit. We also present two special cases which are polynomially solvable. In Section 3, a family of approximation algorithms $\{DP_\varepsilon\}$ for the general problem is presented. For any $\varepsilon > 0$, algorithm DP_ε delivers a schedule, for which $C_j \leq (1 + \varepsilon)d_j,\ j = 1, \ldots, G$ if there exists a schedule which is feasible with respect to the deadlines. The time requirement of DP_ε is $O(G^{2m+1}/\varepsilon^{2m})$. Some concluding remarks are presented in Section 4.

2 Special cases

It is convenient to adopt the three-field descriptor $\alpha/\beta/\gamma$ of Lawler et al. [6] to denote our problem.

The first field $\alpha = \alpha_1\alpha_2$ specifies the machine environment. In this paper, we have $\alpha_1 \in \{P, Q, R\}$ where P, Q and R denote identical, uniform and unrelated machines, respectively. If $\alpha_1 = P$ then $p_{jl} = p_j$, $t_{jl} = t_j$; if $\alpha_1 = Q$, then $p_{jl} = p_j/s_l$, $t_{jl} = t_j/s_l$ and if $\alpha_1 = R$, then p_{jl} and t_{jl} are arbitrary nonnegative integers. In the case of uniform machines the values s_l represent machine speeds. If α_2 is a positive integer, then the number m of machines is a constant equal to α_2 and it is specified as part of the problem type. If α_2 is empty, then m is a variable, the value of which is specified as part of the problem instance. The second field $\beta \subset \{\emptyset, p_j = p,\ t_j = t,\ q_j = q,\ d_j = d, t_j = cp\}$ indicates a number of group characteristics. Here $t_j = t$ and $q_j = q$ indicate a common set-up time and a common number of jobs for each group, respectively. Furthermore, $p_j = p,\ t_j = cp$ indicates that the set-up time is common for all groups and it is divisible by the common processing time for all jobs. Other characteristics are traditionally used in the notation for scheduling problems. The third field, $\gamma = \{C_j \leq d_j\}$, refers to the objective of the problem. Our original problem is represented by $R//C_j \leq d_j$.

In this section, we study the computational complexity of various special cases of the original problem.

We first note that an evident reduction shows that the problems $P/t_j = 0,\ q_j = 1,\ d_j = d/C_j \leq d_j$ and $P/p_j = 0,\ q_j = 1,\ d_j = d/C_j \leq d_j$ are equivalent to the known strongly NP-complete problem MULTIPROCESSOR SCHEDULING [1]. Next, we show that $P2/p_j = 1,\ t_j = 1,\ d_j = d/C_j \leq d_j$ is NP-hard and $P/p_j = 1,\ t_j = 1,\ d_j = d/C_j \leq d_j$ is strongly NP-hard.

Theorem 1 *The problem $P2/p_j = 1,\ t_j = 1,\ d_j = d/C_j \leq d_j$ is NP-hard.*

Proof: We show that the problem $P2/p_j = 1,\ t_j = 1,\ d_j = d/C_j \leq d_j$ is NP-hard by a transformation from the known NP-complete problem PARTITION [1]: given positive integers $a_1, \ldots, a_G$, is there a set $S \subset \{1, \ldots, G\}$ such that $\sum_{j \in S} a_j = A/2$, where $A = \sum_{j=1}^{G} a_j$?

Given any instance of the problem PARTITION, we construct the following instance of the problem $P2/p_j = 1,\ t_j = 1,\ d_j = d/C_j \leq d_j$. There are two identical machines and G groups with $p_j = t_j = 1,\ d_j = d = A$ and

$q_j = 2a_j - 1$ for $j = 1, \ldots, G$. We show that there exists a set S for which $\sum_{j=1}^{G} a_j = A/2$ if and only if there is a schedule for which $C_j \leq d$ for $j = 1, \ldots, G$.
If there is a set S for which $\sum_{j=1}^{G} a_j = A/2$, then we combine all jobs of each group to form a single batch and schedule all groups of S non-splitted on the first machine and all other groups non-splitted on the second machine. Then, the maximum group completion time is exactly $A = d$.
Conversely, suppose there is a schedule for which $C_j \leq d,\ j = 1, \ldots, G$. We note that there are exactly G batches, since otherwise the sum of the processing times and set-up times is at least $2A + 1$ which implies that the maximum group completion time cannot be less than $A + 1/2 > d$. Therefore, all groups are non-split and set S corresponds to the set of groups that are scheduled on one of the machines. □
Similar transformation from the strongly NP-complete problem 3-PARTITION [1] shows that the problem $P/p_j = 1,\ t_j = 1,\ d_j = d/C_j \leq d_j$ is strongly NP-hard. However, if there are equal numbers of jobs in each group, then this problem can be polynomially solved even if $p_j = p,\ t_j = cp,\ j = 1, \ldots, G$, by applying McNaughton's *Wrapping Around Rule (WAR)*: schedule the groups one after the other in $[0, d]$ by filling this interval for each machine, one machine after the other. Continue to schedule a group on the next machine if the group cannot be completed on the current machine. The following theorem holds.

Theorem 2 *WAR solves the problem* $P/p_j = p,\ t_j = cp,\ q_j = q,\ d_j = d/C_j \leq d_j$.

Proof: In the problem considered, all groups are identical and all machines are identical. Assume without loss of generality that schedules which differ only in the numeration of the groups or machines are the same. Then only one schedule can be constructed by applying WAR. To prove the theorem, it remains to show that if there is a feasible schedule, then a feasible schedule is constructed by WAR.
Consider a feasible schedule. Assume that jobs are scheduled according to WAR on the first $k - 1$ machines, $1 \leq k \leq m - 1$ and jobs of a certain group i are scheduled last on machine $k - 1$. Consider machine k. Since machines are identical, assume without loss of generality that q_{ik} jobs of group i are scheduled first on machine k, q_{jk} jobs of a certain group j are

scheduled second on machine k and q_{il} jobs of group i are scheduled on a certain machine $l > k$. The following three cases are only possible.
Case 1: $q_{il} < q_{jk}$.
In this case, interchange q_{il} jobs of group i on machine l with the same number of jobs of group j on machine k.
Case 2: $q_{jk} < q_{il} \leq q_{jk} + c$.
In this case, interchange q_{il} jobs of group i on machine l with all q_{jk} jobs of group j on machine k.
Case 3: $q_{il} > q_{jk} + c$.
In this case, interchange $q_{jk} + c < q_{il}$ jobs of group i on machine l with all q_{jk} jobs of group j on machine k.
It is easy to see that the above interchange procedure leads to a feasible schedule in which the number of jobs of group i on machine k is increased and the situation is equivalent to the initial one. Repeat this interchange procedure until we obtain a schedule in which machine k is completely filled with jobs of group i or there is no jobs of group i on machines $l > k$. In the later case assume without loss of generality that all non-split groups scheduled on machine k are scheduled immediately after the batch of group i. Let r be the first group not completely scheduled on machine k. By applying the above interchange procedure, construct a schedule in which as many jobs of group r are scheduled on machine k as posssible. Thus, we show that there is a feasible schedule in which time interval $[0, d]$ on machine k is filled according to WAR. Repeating this argumentation for $k = 1, \ldots, m$ completes the proof. □
Using the WAR a feasible schedule can be constructed in $O(m)$ time if such a schedule exists. We now begin to study the complexity of the problem with uniform machines. The following theorem is proved by Kovalyov and Shafransky [5]. For the completeness, we present the proof.

Theorem 3 *(Kovalyov and Shafransky [5]) The problem* $Q/p_j = 1,\ t_j = 1\ , q_j = q\ , d_j = 1/C_j \leq d_j$ *is strongly* NP*-hard.*

Proof: We show that the problem $Q/p_j = 1,\ t_j = 1,\ q_j = q,\ d_j = 1/C_j \leq d_j$ is strongly NP-hard by a transformation from the known strongly NP-complete problem 3-PARTITION [1]: given positive integers $a_1, \ldots, a_{3n}$ and A such that $A/4 < a_j < A/2$ for $j = 1, \ldots, 3n$, and $\sum_{j=1}^{3n} a_j = An$, is there a partition of the set $\{1, \ldots, 3n\}$ into n disjoint sets $X_1, \ldots, X_n$ such

that, for $l = 1, \ldots, n$, $\sum_{j \in X_l} a_j = A$? We assume that $a_j > 1$ for $j = 1, \ldots, 3n$. Otherwise, it is easy to see that $A = 3, a_j = 1$ for all j and 3-PARTITION has a trivial solution. Given any instance of 3-PARTITION, we construct an instance of our problem in which there are $3n$ groups with $p_j = t_j = d_j = 1, q_j = q = 3A - 1$ for $j = 1, \ldots, 3n$, and $4n$ machines with $s_l = 3A - a_l + 1$ for *high-speed* machines $l = 1, \ldots, 3n$, and $s_l = A$ for *slow* machines $l = 3n + 1, \ldots, 4n$. We show that there exists a solution to 3-PARTITION if and only if there exists a feasible schedule to the above instance of our problem.

If 3-PARTITION has a solution, then we construct a schedule in which $s_l - 1 = 3A - a_l$ jobs of group l are assigned to high-speed machine l for $l = 1, \ldots, 3n$. Remaining $a_l - 1 > 0$ jobs of group l are scheduled in a single batch on one of the slow machines, $l = 1, \ldots, 3n$. Since 3-PARTITION has a solution, there is an assignment of these $3n$ batches to n slow machines such that the sum of the job processing times and set-up times is exactly A for each slow machine. The speed of any slow machine is also A. Therefore, the deadline $d = 1$ will be satisfied.

Conversely, suppose there is a feasible schedule to the constructed instance of the problem $Q/p_j = 1,\ t_j = 1,\ q_j = q,\ d_j = 1/C_j \leq d_j$. We first consider the case when the following condition is satisfied: time interval $[0, 1]$ on each high-speed machine l is filled only by the jobs of group l for $l = 1, \ldots, 3n$. If so, then remaining $a_l - 1$ jobs of group l are scheduled in a single batch on one of the slow machines for each l. If these jobs are scheduled in more than one batch for a certain group, then the sum of the job processing times and set-up times on all slow machines is more than $\sum_{j=1}^{3n} a_j = An$ and the deadline $d = 1$ cannot be satisfied. Thus, if the above condition is valid, then 3-PARTITION has a solution corresponding to the assignment of the batches to the slow machines. We now show that any feasible schedule can be tranformed into a feasible schedule satisfying this condition.

We first prove that, for any feasible schedule, each group is divided into exactly two batches. Indeed, there cannot be less than two batches, since each group cannot be completely processed by any of the machines. Furthermore, there cannot be more than two batches, since otherwise the sum of the job processing times and set-up times on all $4n$ machines is at least $\sum_{j=1}^{3n}(3A - 1 + 2) + 1 = 9An + 3n + 1$. For any feasible schedule, the sum of set-up times and job processing times on machine l cannot exceed s_l. For all machines, this time cannot exceed $\sum_{l=1}^{4n} s_l = \sum_{l=1}^{3n}(3A - a_l + 1) + An = 9An + 3n$. Thus,

there are exactly two batches for each group. We note that the larger of these two batches is scheduled on one of the high-speed machines, since it includes at least $(3A-1)/2 > A$ jobs which is more than any slow machine can process. Moreover, only one large batch can be scheduled on the same machine, since two large batches include at least $3A-1$ jobs which is more than any machine can process. Assume without loss of generality that the larger batch of group l is scheduled first on high-speed machine $l, l = 1, \ldots, 3n$.
We show that there exists a feasible schedule for which time interval $[0, 1]$ on machine l is completely filled by the jobs of group l for $l = 1, \ldots, 3n$. Let this statement be satisfied for first $k-1$ machines, $1 \leq k \leq 3n$, and on machine k, except the batch of group k, batches of other groups are scheduled. Assume that $b < s_k - 1 = 3A - a_k$ jobs of group k are scheduled first on machine k and the remaining time interval is filled by the batches of other groups and corresponding set-ups. Assume that the smaller batch of group k including $3A - 1 - b$ jobs is scheduled on machine $k' > k$. Move $3A - a_k - b$ jobs of this batch to machine k and move all the batches of other groups from machine k to machine k'. Since $p_j = t_j = 1$ for all j, it is easily checked that this transposition does not change the feasibility of the schedule. Repeat such a tranposition until we obtain a feasible schedule for which time interval $[0, 1]$ on machine l is completely filled by the jobs of group l for $l = 1, \ldots, 3n$. In this case, we have shown that 3-PARTITION has a solution. □
We claim that the problem $Qm/p_j = p,\ t_j = cp,\ q_j = q\ , d_j = d/C_j \leq d_j$ with constant number of machines is polynomially solvable by applying the following *Modified Wrapping Around Rule (MWAR)*: for each sequence $(M_1, \ldots, M_m)$ of machines, apply WAR where machines are considered in order $M_1, \ldots, M_m$. If one of the $m!$ constructed schedules is feasible, then it is a solution to the problem $Qm/p_j = p,\ t_j = cp,\ q_j = q\ , d_j = d/C_j \leq d_j$. Otherwise, there is no feasible solution to this problem. The following theorem holds.

Theorem 4 *MWAR solves the problem* $Qm/p_j = p,\ t_j = cp,\ q_j = q,\ d_j = d/C_j \leq d_j$.

Proof: This theorem is easily proved by applying an interchange argumentation similar to that used in Theorem 2. □
The only open questions for the case of a common deadline and equal $p_j,\ t_j$ or q_j are the complexities of the problems $P/p_j = p,\ t_j = t,\ q_j = q,\ d_j = d/C_j \leq d_j$ and $Qm/p_j = p,\ t_j = t,\ q_j = q,\ d_j = d/C_j \leq d_j$ where $t \neq cp$.

3 Dynamic programming and a fully polynomial approximation scheme

In this section, we derive a dynamic programming algorithm and a fully polynomial approximation scheme for the general problem $Rm//C_j \leq d_j$.
Let C_j^B denote the completion time of group j obtained using a certain $(1+\varepsilon)$-approximation algorithm A_ε. We define A_ε as a $(1+\varepsilon)$-approximation algorithm if $C_j^B \leq (1+\varepsilon)d_j, j = 1, \ldots, G$ for all problem instances in which there exists a feasible solution with respect to deadlines. The family of algorithms $\{A_\varepsilon\}$ defines a fully polynomial approximation scheme if, for any $\varepsilon > 0, A_\varepsilon$ is a $(1+\varepsilon)$-approximation algorithm with a time requirement which is polynomial in G and $1/\varepsilon$.
To facilitate the discussion we formulate problem $Rm//C_j \leq d_j$ as a system of integer inequalities.
Assume that the groups are numbered in EDD order so that $d_1 \leq \ldots \leq d_G$ and there is at most one batch for each group on the same machine. Introduce variables x_{jl} to denote that x_{jl} jobs of group j are assigned to machine l. The processing time of a batch of group j on machine l including set-up time is presented by

$$f_{jl}(x_{jl}) = t_{jl}\varrho(x_{jl}) + p_{jl}x_{jl}$$

where $\varrho(x_{jl}) = 1$ if $x_{jl} > 0$ and $\varrho(x_{jl}) = 0$ if $x_{jl} = 0$. Then, the system is written as follows:

$$\sum_{i=1}^{j} f_{il}(x_{il}) \leq d_j, \qquad l = 1, \ldots, m, \quad j = 1, \ldots, G, \tag{1}$$

$$x_{jl} \in \{0, 1, \ldots, q_j\}, \qquad l = 1, \ldots, m, \quad j = 1, \ldots, G, \tag{2}$$

$$\sum_{l=1}^{m} x_{jl} = q_j, \qquad j = 1, \ldots, G. \tag{3}$$

Let $x_{jl}^*(j = 1, \ldots, G, l = 1, \ldots, m)$ be a solution of this system if it exists.
We now describe a dynamic programming algorithm DP to solve system (1)-(3). This algorithm is based on a general dynamic programming technique

of Rothkopf [9] and Lawler and Moore [7] presented for the classical parallel machine scheduling problem.
The idea is to construct feasible schedules by distributing the jobs of each group to the machines. This is done group by group in an order of non-decreasing group due dates. A schedule for the first j groups $1,\ldots,j$ is given by variables $x_{il},\ i=1,\ldots,j,\ l=1,\ldots,m$ satisfying (2) and (3). We associate with such a partial schedule an m-tuple $(H_{j1},\ldots,H_{jm})$ where $H_{jl}:=\sum_{i=1}^{j} f_{il}(x_{il})$ is the completion time of the last job scheduled on machine l. If $(H_{j-1,1},\ldots,H_{j-1,m})$ corresponds with a feasible schedule (x_{il}) for groups $i=1,\ldots,j-1$ then the extension of this schedule by $x_{jl},\ l=1,\ldots,m$ is feasible too if for $l=1,\ldots,m$ we have $H_{j-1,l}+f_{jl}(x_{jl})\le d_j$. $X^{(j)}$ denotes the set of m-tuples $(H_{j1},\ldots,H_{jm})$ corresponding with all feasible schedules for the groups $1,\ldots,j$. A formal statement of Algorithm DP which calculates the sets $X^{(0)},\ldots,X^{(G)}$ and checks whether $X^{(G)}$ is nonempty is given below.

Algorithm *DP*

Step 1 (Initialization) Set $X^{(0)}=\{(0,\ldots,0)\}$ and $X^{(j)}=\emptyset$ for $j=1,\ldots,G$. Set $j=1$.

Step 2 (Generation of $X^{(1)},\ldots,X^{(G)}$) For each tuple $(H_{j-1,1},\ldots,H_{j-1,m})\in X^{(j-1)}$ and for each tuple $(x_{j1},\ldots,x_{jm})$ satisfying (2),(3) evaluate $H_{jl}=H_{j-1,l}+f_{jl}(x_{jl})$ for $l=1,\ldots,m$. If $H_{jl}\le d_j$ for $l=1,\ldots,m$, then add the tuple $(H_{j1},\ldots,H_{jm})$ to $X^{(j)}$. If the same tuples appear in $X^{(j)}$, store only one of them. If $j=G$, then go to Step 3, otherwise set $j=j+1$ and repeat Step 2.

Step 3 (Solution) If $X^{(G)}$ is not empty, select an arbitrary tuple $(H_{G1},\ldots,H_{Gm})\in X^{(G)}$ and use backtracking to find the corresponding solution $x^*_{jl}(j=1,\ldots,G,l=1,\ldots,m)$. If $X^{(G)}$ is empty, then there is no solution for the system (1)-(3).

The general dynamic programming justification shows that Algorithm DP solves system (1)-(3) in $O(\sum_{j=1}^{G}(q_j+1)^{m-1}d_{j-1}^{m})$ time. Here $d_0=1$. To derive this complexity bound notice that in the jth step we have to combine $(q_j+1)^{m-1}$ $(m-1)$-tuples $(x_{j1},\ldots,x_{j,m-1})$ with at most d_{j-1}^{m} m-tuples $(H_{j-1,1},\ldots,H_{j-1,m})$.

We now describe an idea to reduce the time complexity of Algorithm DP. This idea is also used in our fully polynomial approximation scheme.
Consider a certain iteration j of Step 2 of Algorithm DP. It is apparent that in this iteration we do not need to handle all tuples $(x_{j1}, \ldots, x_{jm})$ which deliver the same set of values $H_{j1}, \ldots, H_{jm}$. It is sufficient to take only one of them. Let $R^{(j)}$ be a set of tuples $(x_{j1}, \ldots, x_{jm})$ which satisfy (2),(3) and deliver all different m-tuples $(H_{j1}, \ldots, H_{jm})$. Algorithm DP can be modified in such a way that in Step 2 "each tuple $(x_{j1}, \ldots, x_{jm})$ satisfying (2),(3)" is replaced by "each tuple $(x_{j1}, \ldots, x_{jm}) \in R^{(j)}$". The time complexity of this modified algorithm is $O(\sum_{j=1}^{G} |R^{(j)}| d_{j-1}^m)$. Thus, if $R^{(j)}$ can be efficiently determined and $|R^{(j)}| < (q_j + 1)^{m-1}$ then the modified algorithm works more efficiently than the original Algorithm DP.
We now modify Algorithm DP to be a $(1 + \varepsilon)$-approximation algorithm for $Rm//C_j \leq d_j$. Our approximation algorithm A_ε consists of three steps. Step 1 and Step 3 are the same as in Algorithm DP. It Step 2, m-tuples $(F_{j1}, \ldots, F_{jm})$ are recursively constructed for $j = 1, \ldots, G$. At each iteration j of Step 2, sets $R^{(j)}(F_{j-1,1}, \ldots, F_{j-1,m})$ are formed. For a given m-tuple $(F_{j-1,1}, \ldots, F_{j-1,m})$, the set $R^{(j)}(F_{j-1}, \ldots, F_{j-1,m})$ is a set of tuples $(x_{j1}, \ldots, x_{jm})$ satisfying (2),(3) which delivers all different m-tuples $(F_{j1}, \ldots, F_{jm})$ such that

$$F_{jl} = \delta_j \lceil (F_{j-1,l} + f_{jl}(x_{jl}))/\delta_j \rceil \leq (1 + \varepsilon) d_j, l = 1, \ldots, m.$$

Here scaling factors $\delta_j = \varepsilon d_j / G$, $j = 1, \ldots, G$, are used.
A formal description of Algorithm DP_ε can be given as follows:

Algorithm DP_ε

Step 1 (Initialization) Set $Y^{(0)} = \{(0, \ldots, 0)\}$ and $Y^{(j)} = \emptyset$ for $j = 1, \ldots, G$. Set $j = 1$.

Step 2 (Generation of $Y^{(1)}, \ldots, Y^{(G)}$) For each tuple $(F_{j-1,1}, \ldots, F_{j-1,m}) \in Y^{(j-1)}$ calculate the set $R^{(j)}(F_{j-1,1}, \ldots, F_{j-1,m})$ and for each tuple $(x_{j1}, \ldots, x_{jm}) \in R^{(j)}(F_{j-1,1}, \ldots, F_{j-1,m})$ evaluate

$$F_{jl} = \delta_j \lceil (F_{j-1,l} + f_{jl}(x_{jl}))/\delta_j \rceil \text{ for } l = 1, \ldots, m. \quad (4)$$

If $F_{jl} \leq (1+\varepsilon)d_j$ for $l = 1, \ldots, m$, then add the tuple $(F_{j1}, \ldots, F_{jm})$ to $Y^{(j)}$. If the same tuples appear in $Y^{(j)}$, store only one of them. If $j = G$, then go to Step 3, otherwise set $j = j + 1$ and repeat Step 2.

Step 3 (Solution) If $Y^{(G)}$ is not empty, select an arbitrary tuple $(F_{G1}, \ldots, F_{Gm}) \in Y^{(G)}$ and use backtracking to find the corresponding solution $x^A_{jl}(j = 1, \ldots, G, l = 1, \ldots, m)$. If $Y^{(G)}$ is empty, then there is no solution for the system (1)-(3).

It is apparent that each $R^{(j)}(F_{j-1,1}, \ldots, F_{j-1,m})$ can be defined as a set of solutions of the system (2)-(4) for all different tuples $(F_{j,1}, \ldots, F_{j,m})$ such that $F_{jl} \leq (1+\varepsilon)d_j, l = 1, \ldots, m$. Since

$$\lfloor \frac{G}{\varepsilon} + G \rfloor \delta_j = \lfloor \frac{G}{\varepsilon}(1+\varepsilon) \rfloor \frac{\varepsilon}{G} d_j \leq (1+\varepsilon)d_j \text{ and } (\lfloor \frac{G}{\varepsilon} + G \rfloor + 1)\delta_j > (1+\varepsilon)d_j$$

we have $F_{jl} = k_{jl}\delta_j$ where $k_{jl} \in \{0, 1, \ldots, \lfloor G/\varepsilon + G \rfloor\}$ for all l. This implies $|R^{(j)}(F_{j-1,1}, \ldots, F_{j-1,m})| \leq O(G^m/\varepsilon^m)$.
We now show that system (2)-(4) can be solved in $O(m)$ time for a fixed pair of tuples $(F_{j1}, \ldots, F_{jm})$ and $(F_{j-1,1}, \ldots, F_{j-1,m})$.
Using the inequality $\lceil u \rceil - 1 < u \leq \lceil u \rceil$, we rewrite (4) as follows:

$$F_{jl} - \delta_j = \delta_j(\lceil (F_{j-1,l} + f_{jl}(x_{jl})/\delta_j \rceil - 1) < \delta_j(F_{j-1,l} + f_{jl}(x_{jl})/\delta_j$$
$$\leq \delta_j \lceil (F_{j-1,l} + f_{jl}(x_{jl})/\delta_j \rceil = F_{jl}$$

or, equivalently,

$$F_{jl} - F_{j-1,l} - \varrho < f_{jl}(x_{jl}) \leq F_{jl} - F_{j-1,l}.$$

For fixed indices j and l, the last inequalities can be written as follows:

$$A < f(x) = t\varrho(x) + px \leq B$$

where $x = x_{jl}$, $t = t_{jl}$, $p = p_{jl}$, $B = F_{jl} - F_{j-1,l}$, and $A = B - \varrho$. The graph of the function $f(x)$ is given in Figure 3.

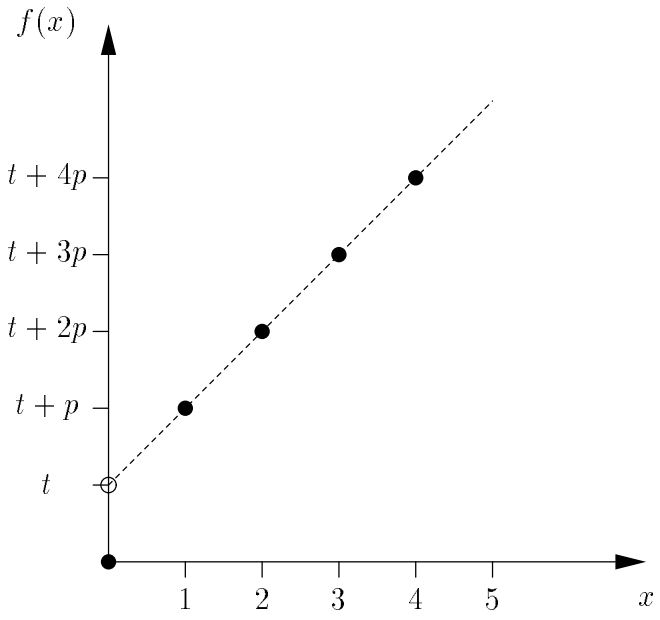

Figure 3

An analysis of the function $f(x)$ shows that either $A < f(x) \leq B$ has no solution $x \geq 0$ or it is equivalent to an inequality $a \leq x \leq b$.
In the first case (2)-(4) has no solution. This case applies if and only if one of the following conditions holds:

1. $B < 0$
2. $A \geq 0$ and $B < t + p$

Otherwise we have the following values for a and b.

3. If $A \geq 0$ and $B \geq t + p$ then

$$a = \left\lceil \frac{A - t}{p} \right\rceil^* \text{ and } b = \left\lfloor \frac{B - t}{p} \right\rfloor.$$

   Here $\lceil u \rceil^*$ denotes the lowest integer greater than u.

4. If $A < 0$ and $B \geq t + p$ then

$$a = 0 \text{ and } b = \left\lfloor \frac{B - t}{p} \right\rfloor.$$

5. If $A < 0$ and $B < t + p$ then $a = b = 0$.

Thus, if the system (2)-(4) has an integer solution then the relations (4) can be written as

$$a_{jl} \le x_{jl} \le b_{jl}, \quad l = 1, \ldots, m \tag{5}$$

where a_{jl} and b_{jl} are the values of a and b, respectively, belonging to the corresponding pair of indices (j, l).
It is evident that systems (2)-(4) and (2),(3),(5) are equivalent. System (2), (3),(5) can be solved in the following way.
If $a_{jl} > b_{jl}$ for a certain l or $\sum_{l=1}^{m} a_{jl} > q_j$ or $\sum_{l=1}^{m} b_{jl} < q_j$ then there is no solution. Assume that $a_{jl} \le b_{jl}$ for $l = 1, \ldots, m$ and

$$\sum_{l=1}^{m} a_{jl} \le q_j \le \sum_{l=1}^{m} b_{jl}. \tag{6}$$

Find such index $k, 1 \le k \le m$ that

$$q_j \le \sum_{l=1}^{k-1} a_{jl} + \sum_{l=k}^{m} b_{jl} \text{ and } q_j \ge \sum_{l=1}^{k} a_{jl} + \sum_{l=k+1}^{m} b_{jl}. \tag{7}$$

Here $\sum_{l=1}^{k-1} a_{jl} = 0$ if $k = 1$ and $\sum_{l=k+1}^{m} b_{jl} = 0$ if $k = m$.
Due to assumption (6), this index k exists. Define values x_{jl} as follows:

$$x_{jl} = a_{jl}, l = 1, \ldots, k-1,$$

$$x_{jl} = b_{jl}, l = k+1, \ldots, m,$$

$$x_{jk} = q_j - (\sum_{l=1}^{k-1} a_{jl} + \sum_{j=k+1}^{m} b_{jl}).$$

Clearly, $\sum_{l=1}^{m} x_{jl} = q_j$. Furthermore, relation $a_{jk} \le x_{jk} \le b_{jk}$ follows from (7). Thus, $(x_{j1}, \ldots, x_{jm})$ is a solution of system (2),(3),(5).
It is clear that the above procedure of solving system (2),(3),(5) runs in $O(m)$ time if the values $\sum_{l=1}^{k} a_{jl}$ and $\sum_{l=k}^{m} b_{jl}, k = 1, \ldots, m$ are calculated in advance.
Thus, if m is fixed, then the procedure of generating the set $R^{(j)}(F_{j-1,1}, \ldots, F_{j-1,m})$ can be done with at most $O(G^m/\varepsilon^m)$ operations for each tuple $(F_{j-1,1}, \ldots, F_{j-1,m}) \in Y^{(j-1)}$.

We now show that the family of algorithms $\{DP_\varepsilon\}$ forms a fully polynomial approximation scheme for the original problem if number m of machines is a constant.

Theorem 5 *The family of algorithms $\{DP_\varepsilon\}$ forms a fully polynomial approximation scheme for the problem $Rm//C_j \le d_j$ and DP_ε has a time complexity of $O(G^{2m+1}/\varepsilon^{2m})$.*

Proof: Clearly, if DP_ε finds a solution x^A_{jl} then this solution satisfies (2),(3). We prove that if Algorithm DP provides a solution $x^*_{jl}(j = 1,\dots,G, l = 1,\dots,m)$ then

$$\sum_{i=1}^{j} f_{il}(x^A_{il}) \le (1+\varepsilon)d_j, j = 1,\dots,G. \tag{8}$$

We first show that if a solution $x^*_{jl}(j = 1,\dots,G, l = 1,\dots,m)$ exists then $Y^{(G)} \neq \emptyset$. For

$$F_{jl} = \delta_j \lceil (F_{j-1,l} + f_{jl}(x^*_{jl}))/\delta_j \rceil$$

we have

$$F_{jl} \le F_{j-1,l} + f_{jl}(x^*_{jl}) + \delta_j \le \dots \le \sum_{i=1}^{j} f_{il}(x^*_{il}) + \sum_{i=1}^{j} \delta_i,$$

$$l = 1,\dots,m, j = 1,\dots,G.$$

Since

$$\sum_{i=1}^{j} \delta_i \le \varepsilon \sum_{i=1}^{j} d_i/G \le \varepsilon \frac{j}{G} d_j \le \varepsilon d_j \text{ and } \sum_{i=1}^{j} f_{il}(x^*_{il}) \le d_j,$$

we have

$$F_{jl} = \delta_j \lceil (F_{j-1,l} + f_{jl}(x^*_{jl}))/\delta_j \rceil \le (1+\varepsilon)d_j, l = 1,\dots,m, j = 1,\dots,G.$$

The latter relations yield $Y^{(j)} \neq \emptyset$ for each j.
We now show that if $Y^{(G)} \neq \emptyset$ then (8) is satisfied.
By definition, the solution $x^A_{jl}(j = 1,\dots,G, l = 1,\dots,m)$ provides values F_{jl} such that

$$F_{jl} \le (1+\varepsilon)d_j, l = 1,\dots,m, j = 1,\dots,G.$$

Moreover, for these values we have

$$F_{jl} \geq F_{j-1,l} + f_{jl}(x_{jl}^A) \geq \ldots \geq \sum_{i=1}^{j} f_{il}(x_{il}^A).$$

Thus, (8) is satisfied.
We now establish the time complexity of DP_ε assuming that m is a constant. The procedure of generating $Y^{(j)}$ can be done in $O(|Y^{(j-1)}|G^m/\varepsilon^m)$ time since $|R^{(j)}(F_{j-1,1}, \ldots, F_{j-1,m})| \leq O(G^m/\varepsilon^m)$. It is apparent that the number of different tuples in $Y^{(i)}$ is at most $O(G^m/\varepsilon^m)$ for each i. Thus, the procedure of generating $Y^{(j)}$ can be done in $O(G^{2m}/\varepsilon^{2m})$ time for each j and the overall time complexity is $O(G^{2m+1}/\varepsilon^{2m})$. □

4 Concluding remarks

The problem of scheduling groups of identical jobs on m (identical, uniform or unrelated) parallel machines $l = 1, \ldots, m$ has been investigated. Associated with each job group $j = 1, \ldots, G$ consisting of q_j identical jobs with processing time p_{jl} there is a due date d_j and a set-up time t_{jl} for starting to process a number of jobs of group j on machine l. We considered the problem of finding a schedule satisfying the conditions $C_j \leq d_j$ where C_j denotes the completion time of group j.
It was shown that already the following problems in which we have two identical machines are NP-hard:

$$\begin{array}{llll} P2/p_j = 1, & t_j = 1, & d_j = d/C_j \leq d_j \\ P2/t_j = 0, & q_j = 1, & d_j = d/C_j \leq d_j \\ P2/p_j = 0, & q_j = 1, & d_j = d/C_j \leq d_j. \end{array}$$

The following problems have been shown to be strongly NP-hard:

$$\begin{array}{llll} P/p_j = 1, & t_j = 1, & & d_j = d/C_j \leq d_j \\ Q/p_j = 1, & t_j = 1, q_j = q, & & d_j = 1/C_j \leq d_j. \end{array}$$

Problems

$$\begin{array}{llll} P/p_j = p, & t_j = cp, q_j = q, & d_j = d/C_j \leq d_j \\ Qm/p_j = p, & t_j = cp, q_j = q, & d_j = d/C_j \leq d_j \end{array}$$

are polynomially solvable.

Problems

$$P/p_j = p, \quad t_j = t, q_j = q, \quad d_j = d/C_j \leq d_j$$
$$Qm/p_j = p, \quad t_j = t, q_j = q, \quad d_j = d/C_j \leq d_j$$

where $t \neq p$ are still open.

A dynamic programming algorithm and a family of approximation algorithms $\{DP_\varepsilon\}$ was constructed for the general problem with unrelated machines. For any $\varepsilon > 0$, algorithm DP_ε delivers a schedule in which the completion time of each group is at most $(1+\varepsilon)$ times the value of its deadline if there exists a schedule which is feasible with respect to deadlines. The time complexity of DP_ε is $O(G^{2m+1}/\varepsilon^{2m})$ if the number m of machines is a constant.

Suppose that d_j is treated as a time to deliver group j to a customer after its completion and the objective is to minimize the maximum delivery time $D_{\max} = \max_{1\leq j\leq G}\{C_j + d_j\}$. It is possible to adopt our approximation scheme to handle this more general problem $Rm//D_{\max}$.

Firstly, we determine a lower bound LB and an upper bound UB for the minimal objective value $D^*_{\max}$. The simplest bounds are $LB = \max_{1\leq j\leq G}\{d_j\}$ and $UB = LB + \min_{1\leq l\leq m}\{\sum_{j=1}^{G}(t_{jl} + p_{jl}q_j)\}$. Then we apply a bisection search procedure, in which, for a trial value $D \in [LB, UB]$, we set $d'_j = D - d_j,\ j = 1, \ldots, G$ and apply algorithm DP_ε for the problem $Rm//C_j \leq d'_j$. If it finds a solution, then, for this solution, we have $C_j \leq d'_j(1+\varepsilon) = (D - d_j)(1+\varepsilon),\ j = 1, \ldots, G$, from which follows that $D^*_{\max} \leq D(1+\varepsilon)$. We store this solution, since if $D^*_{\max} \geq D$, then the value of this solution is at most $D^*_{\max}(1+\varepsilon)$. We define a new upper bound $UB = D$. If DP_ε does not find any solution then there is no such schedule that $C_j \leq d'_j,\ j = 1, \ldots, G$, i.e. $D^*_{\max} > D$. In this case, we define a new lower bound $LB = D$. In both cases, we define a new trial value $D = (LB + UB)/2$. The procedure is finished when $UB - LB \leq 1$. It is easy to check that the solution with the minimal value $D_{\max}$ among those found by the procedure provides such a value $D^0_{\max}$ of the maximum delivery time that $D^0_{\max} \leq (1+\varepsilon)D^*_{\max}$. The number of iterations of the procedure is $O(\log(UB - LB))$. Thus, it determines a fully polynomial approximation scheme for the problem $Rm//D_{\max}$.

References

[1] M.R. Garey and D.S. Johnson (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., New York.

[2] I. Ham, K. Hitomi and T. Yoshida (1985) *Group technology*, Kluwer-Nijhoff, Dordrecht.

[3] J.R. Jackson (1955) *Scheduling a Production Line to Minimize Maximum Tardiness*, Research Report 43, Management Science Research Project, University of California, Los Angeles.

[4] M.Y. Kovalyov (1993) *New rounding technique in combinatorial approximation*, Preprint N 1, Institute of Engineering Cybernetics, Belarus Academy of Sciences, Minsk.

[5] M.Y. Kovalyov and Y.M. Shafransky (1994) *Scheduling of Identical Jobs on Uniform Machines in Batches*, Working Paper, Institute of Engineering Cybernetics, Belarus Academy of Sciences, Minsk.

[6] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (1989) *Sequencing and scheduling: algorithms and complexity*, Report 8934/A, Econometric Institute, Erasmus University, Rotterdam.

[7] E.L. Lawler, J.M. Moore (1969) *A functional equation and its application to resource allocation and sequencing problems*, Management Science 16, 77-84.

[8] C.N. Potts and L. N. Van Wassenhove (1991) *Integrating Scheduling with Batching and Lot-Sizing: a Review of Algorithms and Complexity*, J. Opl. Res. Soc. 43, 395-406.

[9] M.H. Rothkopf (1966) *Scheduling independent tasks on parallel processors*, Management Science 12, 437-447.