Vakhania, Nodari

A Fast Algorithm for Scheduling Equal-Length Jobs on Identical Machines

Computación y Sistemas, vol. 1, núm. 4, abril-junio, 1998, pp. 239-250

Instituto Politécnico Nacional

Distrito Federal, México

Available in: http://www.redalyc.org/articulo.oa?id=61510406
A Fast Algorithm for Scheduling Equal-Length Jobs on Identical Machines

Nodari Vakhania
Facultad de Ciencias
Universidad Autónoma del Estado de Morelos
Cuernavaca, Mexico

Article Received on November, 1997; accepted on April, 1998.

Abstract

The problem of sequencing jobs of equal durations with available (readiness) times and the additional tails on a set of parallel identical processors is considered. The objective is to minimize the maximal completion time. We present a new polynomial algorithm which improves the running time of the previously known best algorithm under the realistic assumption that tails of all jobs are bounded by some sufficiently large constant.

Keywords: Scheduling, Identical Processors, Readiness Time, Tail, Computational Complexity.

1 Introduction

Scheduling problems constitute part of the combinatorial optimization problems. Combinatorial optimization itself is a relatively new field, traditionally belonging to Operations Research, now also it significantly interacts with Computer Science. The combinatorial optimization problems are discrete optimization problems, the finite set of feasible solutions and a goal function which has to be minimized (or maximized). A problem is (exactly) solved if a feasible solution with the (maximal) value of a goal function is found. The set of feasible solutions is finite, it might turn out to be "too big", so that the complete enumeration is practically impossible, since it would take an inordinate amount of machine time and memory. The dependence of the number of feasible solutions of a problem on the length of input (the size of P) might be polynomial as well as exponential. In the latter case we are forced to find an algorithm for P with polynomial dependence. The problems with polynomial dependence are much easier to solve than the problems with exponential dependence, since they take significantly less computational time. For a polynomial problem P, there are several algorithms with different degrees of polynomial. Then the algorithm with the smallest degree is preferable, since it takes less computational time. In this paper we propose an algorithm for a polynomial instance of scheduling $n$ equal-length jobs (or tasks) on identical processors to minimize the total completion time of all jobs. Our algorithm has a smaller degree of polynomial than the earlier ones under a realistic type of constraints which we impose on the problem data.
machines; the processing time of any job (on any machine) is a given integer number $p$. Job $i$ ($i = 1, 2, ..., n$) is available at its integer readiness time $a_i$ (this job cannot be started before the time $a_i$) and has an integer tail $q_i$ (interpreted as the additional amount of time needed for the termination of job $i$ once it is processed on a machine). A schedule is a function which assigns a machine to each job and a starting time (on that machine). An (integer) starting time $t_i^S$ of job $i$ (in the schedule $S$) is the time at which this job is scheduled to be performed on a machine. The completion time of job $i$ on a machine $c_i^S = t_i^S + p$. The full completion time of job $i$ in the schedule $S$ is $c_i^S + q_i$ (notice that $q_i$ doesn't take any machine time). Each machine can handle at most one job at a time, that is, if jobs $i$ and $j$ are scheduled on the same machine then either $c_i^S \leq t_j^S$ or $c_j^S \leq t_i^S$.

The preemption of jobs is not allowed, that is, each job is performed during the time interval $[t_i^S, t_i^S + p]$ on a machine. A feasible schedule is a schedule which satisfies the above restrictions. The objective is to find an optimal schedule, that is, a feasible schedule which minimizes the makespan (the maximum full job completion time).}

$$L_i^S = \begin{cases} 0, & \text{if } c_i^S \leq d_i; \\ c_i^S - d_i, & \text{otherwise}. \end{cases}$$

The objective is to find an optimal schedule $S$ of a feasible schedule which minimizes the maximum completion time $L_i^S = \max \{ L_i^S | i = 1, 2, ..., n \}$. The equivalence of the two problems $P1$ and $P2$ is established by a transformation [Bratley et al., 1973].

If we allow in $P1$ or in $P2$ different problem we get strongly NP-complete problem even in one machine case [Baker & Zaw, 1974; Bratley & Carlier, 1982; Garey & Johnson, 1979; McIlvain, 1975].

If we replace in $P2$ due dates with deadlines for a feasible schedule, we get the corresponding problem $PF$ (by $PF1$ we abbreviate the transformed version of $PF$). In a feasible schedule $S$ of $P2$, if a feasible schedule of $P2$ we allow the existence of jobs and we look for a schedule which minimizes.
of the so-called forbidden regions (that is, a region in a schedule in which it is forbidden to start a job). Once the algorithm declares the forbidden regions it applies the earliest due date heuristic and constructs the final feasible schedule. In [Simons, 1983] and [Simons & Warmuth, 1989] the concept of forbidden regions is generalized for a multiple-machine case and an \( O(n^3 \log \log n) \) and \( O(n^2 m) \) time algorithms, respectively, are presented for \( PF \).

The minimization problems \( P1 \) and \( P2 \) can be solved by the repeated application of an algorithm for the corresponding feasibility problem. We iteratively increase the due dates of all jobs by some constant until we find a feasible schedule for the feasibility problem with modified data. Since the maximum lateness will depend on the number of jobs, we need to apply such algorithm \( O(n) \) times.

The algorithm for the problem \( P1 \) which we present here has the time complexity \( O(mn \log n) \) under the assumption that the tails of all jobs are bounded by the sufficiently large constant. We notice that for many applications this assumption is realistic and imposes no additional restrictions. With each node of our search tree the so-called complementary schedule is associated. The complementary schedules are complete schedules which we generate iteratively by the application of the greatest tail heuristic to the specially modified problem instance. An overflow job is a job which realizes the value of the maximal completion time in a schedule. We introduce five behaviour alternatives in our algorithm which reflect five different ways of alteration of an overflow job when we generate a new complementary schedule. Our search for an optimal schedule is based on the analysis of the behaviour alternative in the generated complementary schedules.

In Section 2 we introduce the basic definitions and notations. In Section 3 we investigate the properties of the complementary schedules providing the basis for the algorithm construction. In Section 4 we describe the algorithm and indicate its computational complexity. In Section 5 we give the final remarks.

## 2 Basic Concepts

Our search for an optimal schedule can be conveniently represented by a rooted tree \( T \) (we call it the solution tree) which contains the following types of nodes:

- The root node represents the scheduled jobs.
- The leaf nodes represent the unscheduled jobs.
- The internal nodes represent the schedules.

The algorithm searches for the best schedule by exploring the tree. It starts at the root node and explores the tree by selecting the best child for each node. The algorithm terminates when it reaches a leaf node or when it finds a better schedule.

The algorithm follows the extended Schrage heuristic for scheduling jobs on identical machines. The extended Schrage heuristic is an improvement of the basic Schrage heuristic which is known to be optimal for scheduling jobs on identical machines.

To illustrate the extended Schrage heuristic, we consider the problem of scheduling jobs on \( m \) identical machines with the objective of minimizing the total completion time. The heuristic works as follows:

1. Sort the jobs in non-decreasing order of their due dates.
2. Assign the first job to the first machine.
3. For each subsequent job, select the machine with the smallest completion time for the current job.
4. Assign the job to the selected machine and update the completion time of the machine.

The extended Schrage heuristic improves the basic Schrage heuristic by selecting the machine with the smallest weight completion time for the current job, where the weight is the sum of the remaining due dates of the unscheduled jobs.

The extended Schrage heuristic is implemented in the following procedure:

\[ \text{Procedure Extended Schrage.} \]

\[ \begin{align*}
(0) & \quad t := \min\{a_i | i \in I\}; A := I; \\
& \quad \text{if } A \neq \emptyset \text{ then } t := \max\{R_k, \min\{a_i | i \in E\}\}; \\
(1) & \quad \text{return};
\end{align*} \]

Thus, the above algorithm repeatedly determines the next scheduled job using the Extended Schrage heuristic and assigns it to the next machine (next machine \( k \), \( k = 1, 2, \ldots, m - 1 \), is machine \( k + 1 \) and machine \( m \) is machine 1). The schedules, generated in the application of this algorithm are called extended schedules (abbreviated ESS). An example of an ESS is given in Figure 1. The ordinal number of each slot in this schedule is denoted by \( \text{ord}(i, S) \) (i.e., \( i \) is the \( \text{ord}(i, S) \)th scheduled job in \( S \)); we say that job \( i \) is \( \text{ord}(i, S) \)th scheduled job in \( S \).

With the root of our solution tree \( T \) the following notation is used:

- \( a_i \) is the tardiness of job \( i \).
- \( t(s, S) \) is the total completion time of \( S \) at slot \( s \).
- \( \text{ord}(i, S) \) is the ordinal number of job \( i \) in schedule \( S \).
we iteratively apply it to the specially modified problem instances (different from the initially given one). We discuss this in details later in this section.

**Proposition 1.** For any extended Schrage schedule $S$ and jobs $i, j$ such that $\text{ord}(j, S) > \text{ord}(i, S)$, $t_{ij}^S \geq t_{ij}^S$.

**Proof.** Immediately follows from the heuristic of the ESS.

With an extended Schrage schedule $S$ we associate a conjunctive graph $G_S$ (see Figure 1). Each node in this graph, except the (fictitious) source node 0, and the (fictitious) sink node $\ast$, represents a unique job with the same number. We have in $G_S$ the set of (initial) arcs $(0, i), (i, \ast)$, $i = 1, 2, \ldots, n$. With an arc $(0, i)$ ($(i, \ast)$, respectively) the weight $a_i$ ($p + q_i$, respectively) is associated. We complement the set of initial arcs as follows. We add an arc $(i, j)$ ($i, j \not\in \{0, \ast\}$) to $G_S$ with the associated weight $p$ when job $j$ is scheduled directly after job $i$ on the common machine. The makespan of $S$ is then determined by the length of a critical path in $G_S$.

A gap in a schedule is a time interval which is occupied by any job. The greatest tail schedule, i.e., the sequence of one or more blocks. Intuitively, it is an “isolation” part of a schedule. Any gap is not covered by gaps or are earliest scheduled jobs on consecutive machines, and the last $m$ scheduled jobs (correspondingly) are succeeded by gaps scheduled jobs on their respective machines. For any two blocks $B_1, B_2 \in S$ either $B_1 < B_2$ holds, that is, either $B_1$ precedes $B_2$ or $B_2$ precedes $B_1$. In a given schedule, the critical block containing the rightmost maximal path.

In Figure 3, the rightmost maximal path is represented. We call the last scheduled job of the maximal path in $S$ the overflow job and use it by $r$. Let $B$ be the critical block in the Boolean $l \in B$ such that $\text{ord}(l) < \text{ord}(r)$ is called a block in $S$ if $q_l < q_r$. We denote by $K_{S, \mu}$ the set of jobs in $S$, where $\mu$ stands for the respective maximal path.

The sequence of jobs scheduled in $S$ before each job with the maximal ordinal number of an overflow job $r$ (including this job) is called a sequence and is denoted by $C_{S, \mu}$. Notice that $C_{S, \mu}$ are scheduled successively on adjacent machines, and hence may belong to different paths in $G_S$.

We denote by $L(S)$ the length of the (rightmost) critical path in $G_S$ and by $L(S, j)$ the longest path to node $j$ in $G_S$. 

![Figure 2: The Graph $G_S$.](image-url)
the complementary schedule and denote it by \( S_i \). Complementary schedule \( S_i \) we obtain by application of the heuristic of ESS to the specially modified problem instance. We increase the readiness time of job \( i \) as well as some other jobs scheduled in \( S \) after the sequence \( C_{S,\mu} \) so that these jobs will be “forced” and scheduled after all jobs of \( C_{S,\mu} \) by the heuristic of the ESS. So, we leave an additional free space in \( S_i \) giving the possibility to the urgent jobs from the emerge sequence to start their processing earlier.

Let \( I' \) be the set consisting of all jobs from \( C_{S,\mu} \) and the jobs scheduled before \( C_{S,\mu} \) in \( S \), excluding job \( i \in K_{S,\mu} \); let \( S' \) denotes the partial schedule obtained by the application of the heuristic of the ESS to the set \( I' \). Let us redefine the readiness times of all jobs \( i \in I \setminus I' \), as follows: \( a_i := \max\{t_i^{S'}, a_i\} \) (\( t_i^{S'} \) is said to be the threshold value for \( S_i \)). Now the complementary schedule \( S_i \) is an extension of \( S' \) obtained by application of the heuristic of the ESS to the remained jobs (with the modified readiness times) from \( I \setminus I' \).

The rightmost maximal path with the respective overflow job might alternate in different ways in newly generated complementary schedule. Let us denote these conditions as follows:

(a) unmoved, if \( r(S_i) = r(S) \),
(b) rested on \( i \), if \( r(S_i) = i \),
(c) shifted forward,
(d) shifted backward, if \( r(S_i) \) and \( r(S) \) are in the same block (\( r(S_i) \neq r(S) \)),
(e) otherwise, the critical path is said to be different.

All the alternatives except the last one are possible in the current block: in the case of the instance of the four alternatives we “stay” in the current block and do not rearrange further the necessary rearrangement. In the instance of the alternative five we “move” the whole block (and make necessary rearrangement). The last alternative will again analyze the behavior of the critical path and newly generated complementary schedule according to the new process. As it will be evident later, the implementation of the alternative five (alternatives (a), (b)) will cause less computational effort than instances of the other alternatives.

It can be easily seen that all five alternatives are exhaustive and exhaustive (see the Appendix for the example). The five alternatives are exhaustive (we can implement any of them in any \( S_i \)): the overflow job in \( S_i \) may be in the same as in \( S \) (the alternative (a)) or change block (the alternative (d)) or stay in the current block (the alternative (e)). Otherwise, either it can move to another block (the alternative (d)) or stay in the current block (the alternative (e)). The latter case we have two possibilities: the overflow job is scheduled after \( r(S) \) in \( S_i \) (the alternative (e)) or is scheduled before \( r(S) \) (the alternative (d)).

Proposition 2. The alternatives (a) to (e) are exhaustive and exhaustive.

3 Study of the Complementary Schedules

In this section we give the basic properties of complementary schedules which we use later in our applications.
yields \( a_j > t^S_i \) since otherwise job \( j \) would be scheduled at the moment \( t^S_i \) in \( S \) (by the heuristic of the ESS). From the definition of the complementary schedule we have that no job from those which were scheduled after \( C_{S,\mu} \) in \( S \) can occupy any interval before \( C_{S,\mu} \) in \( S_t \). Therefore we will have a gap \([t^S_i, a_j)\) in \( S_t \).

Now suppose \( j \) is an emerge job with \( q_j \leq q_i \) and \( a_j \leq t^S_i \) (if \( a_j > t^S_i \) then we have a gap \([t^S_i, a_j)\) again by the heuristic of the extended Schrage schedule and the definition of the complementary schedule, \( j \) will be the \( l \)th scheduled job in \( S_t \) and (due to the equal processing times) we will have a gap in \( S_t \) strictly before \((\text{ord}(j, S) + 1)\)th scheduled job if \( j \) is the only remained emerge job. If not, the next slot might be occupied by another emerge job. It is easy to see that there will be a gap in \( S_t \) strictly before the job scheduled after the last such emerge job.

By the following lemmas we give other properties of the complementary schedules.

**Lemma 2.** An ESS cannot be improved by rescheduling of any non-emerge job.

**Proof.** Obviously follows from the definition of a non-emerge job and Proposition 1.

**Lemma 3.** An ESS \( S \) cannot be improved by the reordering of jobs of the emerge sequence \( C_{S,\mu} \).

**Proof.** Suppose that in the emerge sequence \( C_{S,\mu} \) job \( m \) precedes job \( l \) and that we have interchanged the order of these two jobs in the schedule \( S' \). Consider the two following possibilities: \( a_l \leq t^S_m \) and \( a_l > t^S_m \).

If \( a_l \leq t^S_m \) then \( q_m \geq q_l \) (by the heuristic of the ESS). Job \( m \) can be scheduled before or after the overflow job \( r \) in \( S' \). The first alternative is obvious (see Proposition 1). For the second one we easily obtain \( L(S', m) > L(S, r) \) since \( q_m \leq q_r \).

If \( a_l > t^S_m \) then we have a gap in \( C_{S,\mu} \) in the schedule \( S' \). Again, job \( m \) can be scheduled before or after the overflow job \( r \). In the first case we obviously have \( L(S', r) > L(S, r) \). In the second case, \( L(S', m) > L(S, r) \) (since \( q_m \geq q_r \)).

Let \( S \) be an extended Schrage schedule and \( \delta_S = c^S_i - a_j \), where \( l = \max\{i \mid i \in K_{S,\mu}\} \), \( a_j = \min\{a_i \mid i \in C_{S,\mu}\} \).

**Lemma 4.** The lower bound on the value of an optimal schedule in the ESS case is \( L(S_t, r) \).

Let \( S \) be a complementary schedule with the maximal path \( \mu \). Consider the set of the complementary schedules \( S_t \), \( l \in K_{S,\mu} \) and the magnitude of the length of \( \mu \) is reduced in each of these schedules. The following lemma shows, this magnitude may increase while we apply an emerge job which has a number, less than that of already applied emerge jobs.

**Lemma 5.** \( L(S_t, r) \leq L(S_k, r) \) if \( l > k \).

**Proof.** Consider the complementary schedule \( S \), which occupies \( \text{ord}(k, S) \)th slot in the heuristic of the ESS we have \( l^S_i \geq t^S_j \). Since this condition holds for all jobs which scheduled before \( k \) and an overflow job \( r \) in \( S \) (in other words, the time of a job, scheduled in \( l \)th \( (\text{ord}(k, S) \leq l) \) slot in \( S_k \) is more than or equal to that of in \( S_t \), obviously the first late slot in \( S_t \) will be orded \((k, S) \), and we have \( \text{ord}(l, S) > \text{ord}(k, S) \). This is a contradiction.
uled emerge job then it makes no sense to improve it. A **closed schedule** is a schedule without successors which cannot have successors, while an **open schedule** is a schedule which is not closed and has no successors.

**Lemma 6.** Suppose in the complementary schedule $S_l, l \in K_{S,l}$, a critical path is rested on $l$. Then:
1. $S_l$ can be closed;
2. Any complementary schedule $S_k$ such that $k < l$ and $q_k \geq q_l$ can be neglected.

**Proof.**

Part 1. Suppose $l'$ is any emerge job in $S_l$ (if there is no emerge job in $S_l$ then it can be closed, Lemma 2). This job also emerges in $S$ since $q_l < q_l$. If $ord(l', S) > ord(l, S)$ then $S_l$ can be neglected (this lemma, part 2). Let now $ord(l', S) < ord(l, S)$. Consider the nested complementary schedule $(S_l)_l$. If $L((S_l)_l, l') > L(S_l, l)$ then obviously $(S_l)_l$ can be neglected. Assume $L((S_l)_l, l') < L(S_l, l)$. Then also $L(S'_l, l') < L(S_l, l)$ since job $l'$ in $S'_l$ will be scheduled in an earlier slot than in $(S_l)_l$ (see Proposition 1), hence $L(S'_l) < L(S_l)$ and again $S_l$ can be closed.

Part 2. Obviously follows from Lemma 5.

The following lemma enables us to reduce the number of complementary schedules we generate:

**Lemma 7.** If $l > k$ and $q_l \leq q_k$ $(l, k \in K_{S,l})$, $S \in T$ then the complementary schedule $S_k$ can be neglected if the complementary schedule $S_l$ is generated.

**Proof.** Suppose that a critical path in $S_l$ is rested on $l$. Then it is rested on $k$ in $S_k$ and $L(S_l, l) \leq L(S_k, k)$ since $q_l \leq q_k$. If a critical path in $S_l$ is unmoved then from Lemma 5 we have

$$L(S_l, r) \leq L(S_k, r)$$

and obviously the schedule $S_k$ can be neglected.

Let a critical path in $S_l$ be shifted forward. If a critical path in $S_k$ is rested on $k$ then this schedule cannot be further improved (Lemma 6); also it cannot be better than $S_l$ since $q_l \leq q_k$ and $l > m$ (Lemma 5).

Suppose in $S_k$ a critical path is unmoved. Again from Lemma 5 we have $L(S_l) < L(S_l, r) \leq L(S_k, r) = L(S_k)$ and obviously we have to generate a complementary schedule of the form $(S_k)_{k'}$, $k' \in K_{S,k}$, $k' > k$ to improve the value $L(S_k, r)$. We have $(S_k)_{k'} = (S_k)_{k'}$. If $k'$ is not the optimal.

of the theorem.

Suppose a critical path in $S_k$ is shifted backward. From Lemma 5 and from the definition of $l$ and $q_l$ (for the equal processing times) we easily get that $L(S_l)$ and that none of the complementary schedules $S_k$ can have makespan better than the schedule $S_l$ if we succeed in the schedule $S_l$ be brought to the schedule which cannot be brought to the schedule $S_l$.

Let now, in both $S_l$ and $S_k$, a critical path be shifted forward. Consider the complementary schedules $(S_l)_l$, $(S_k)_k$, obtained from schedules $S_l$, $S_k$ by rescheduling repeatedly jobs $l$ and $k$ (as an emerge jobs). Observe that, if $k$ in $(S_k)_k$, then $l$ is also emerge in $(S_l)_l$. Besides, the ordinal number of $l$ in $(S_l)_l$ is greater than or equal to the ordinal number of $l$ in $(S_k)_k$ (again, because $q_l \leq q_k$). This again inequality of the form $(\ast)$. The lengths of a critical path in the considered schedules are decreasing step by step and the number of such schedules is bounded by a finite tail (for details we refer to our proof of Theorem 3). As a result, we are brought either to the situation when the job $k$, or both $l$ and $k$ become non-emerge and cannot be further used for a schedule improvement to one of the situations considered above while the intermediate complementary schedules inequalities form Eq. $(\ast)$ are satisfied.

Suppose now that a critical path in $S_l$ is shifted forward. A new arisen gap forces an order change of jobs in $S_l$; let $j, j'$ be the corresponding couple of jobs in $S_l$. If the processing order of $j$ and $j'$ in $S_k$ is the same as in $S_l$ (i.e., job $j'$ preceding $j$) then obviously the starting time of job $j$ is not less than that in $S_l$, that is, $S_k$ cannot be better than $S_l$. If job $j$ precedes job $j'$ in $S_k$ then the starting time of ord($j$, $S_l$)-th slot is greater than that of the heuristic of the ESS). Therefore the starting time of all consequent slots in $S_k$ is greater than that of $S_l$ (Proposition 1) and again $L(S_k) \geq L(S_l)$. For the successor schedule of $S_k$ we apply the reasoning already used for the different behaviour alterna-atives (a) to (c). The lemma is proved.

We use Lemma 7 to reduce the set of emerge jobs $K_{S,l}$ of the subset of the emerge jobs $K_{S,l}$, we will call $K_{S,l}$ a set of emerge jobs, if for any pair $l, k \in K_{S,l}$, we have $L(S_l) \leq L(S_k)$. The alternative (e) obviously reduces to one of the alternatives (a) to (c). The lemma is proved.

The following lemma enables us to reduce the number of complementary schedules we generate:

**Lemma 7.** If $l > k$ and $q_l \leq q_k$ $(l, k \in K_{S,l})$, $S \in T$ then the complementary schedule $S_k$ can be neglected if the complementary schedule $S_l$ is generated.

**Proof.** Suppose that a critical path in $S_l$ is rested on $l$. Then it is rested on $k$ in $S_k$ and $L(S_l, l) \leq L(S_k, k)$ since $q_l \leq q_k$. If a critical path in $S_l$ is unmoved then from Lemma 5 we have

$$L(S_l, r) \leq L(S_k, r)$$

and obviously the schedule $S_k$ can be neglected.

Let a critical path in $S_l$ be shifted forward. If a critical path in $S_k$ is rested on $k$ then this schedule cannot be further improved (Lemma 6); also it cannot be better than $S_l$ since $q_l \leq q_k$ and $l > m$ (Lemma 5).

Suppose in $S_k$ a critical path is unmoved. Again from Lemma 5 we have $L(S_l) < L(S_l, r) \leq L(S_k, r) = L(S_k)$ and obviously we have to generate a complementary schedule of the form $(S_k)_{k'}$, $k' \in K_{S,k}$, $k' > k$ to improve the value $L(S_k, r)$. We have $(S_k)_{k'} = (S_k)_{k'}$. If $k'$ is not the optimal.

The alternative (e) obviously reduces to one of the alternatives (a) to (c). The lemma is proved.
Lemma 9. The number of complementary schedules, the direct successors of a particular complementary schedule $S$ is bounded by $\delta_S$.

Proof. There are no more than $\delta_S$ possibilities to reduce the length of a critical path in $S$ (by 1, 2, $\ldots$, $\delta_S$, see Lemma 4). From all the complementary schedules, for which the critical path is reduced by a certain quantity $\delta, 1 \leq \delta \leq \delta_S$, we generate only one (Lemma 8). Thus, we generate no more than $\delta_S$ complementary schedules.

4 The Algorithm

In this section we finally give our algorithm. But before, we need some additional definitions and lemmas.

Let $b_T$ be any branch in $T$ and let $S$ be the first complementary schedule in $b_T$ such that $r \in I$ is the overflow job in it. A complementary schedule $S' \in T$, a successor of $S$, we call $l$th level nested complementary schedule of job $S$ if:

1. A critical path in $S'$ is unmoved with job $r$ being an overflow job in it (the instance of alternative (a)) or it is relocated to job $r$ (the instance of alternative (e));
2. $S'$ has exactly $l-1$ predecessors in $b_T$ satisfying the condition 1.

Thus the main characteristic feature of a nested complementary schedule of $S$ is that $r$ is the overflow job in it.

Let $\pi = \min \{|C_{S,u}|, m\}$. We say that the nested complementary schedule of $S$, $S'$ is well-defined, if any of the first $\pi$ jobs of $C_{S,u}$ is preceded by a gap in it.

Lemma 10. Any well-defined nested complementary schedule $S' \in T$ might be closed.

Proof. From the definition of $S'$ we have that first $\pi$ jobs of $C_{S,u}$ in this schedule are starting at their earliest starting times. Therefore the value $L(S', r)$ cannot be at most $m$ is well-defined. Consequently, closed.

Proof. Consider $S'$, a $\pi$th level nested complementary schedule of $S$. We claim that this schedule is defined. We get this claim from the definition of a nested Schrage and complementary schedule. $u$, let $j$ denote a job from $C_{S,u}$ with the minimum time. In a nested complementary schedule, the first job $j$ will occupy the slot $ord(j, S')$. This job will be preceded by a gap (by the definition of a complementary schedule and Lemma 1). Assume in the next, nested complementary schedule $S''$ of level two, job $j$ will occupy the slot $ord(j, S'')$ and the next job from $C_{S,u}$ will occupy the slot of $j$ and both jobs will be preceded by a gap. Next, the $\pi$th level nested complementary schedule of $S''$ must define nested complementary schedule. Consequently, it can be closed (Lemma 10).

Suppose $S^*$, $S'$ are the complementary schedule, the common parent schedule $S$, $S^*$ is the schedule of a non-empty set of successors and $S'$ is an open schedule. Let, further $S''$ be a successor of $S^*$ and let $A(S'')$ be the active job of the complementary schedule satisfying the following:

Lemma 12. The complementary schedule $S''$ is closed if the complementary schedule $S''$ with $q_{A(S'')}$ is generated.

Proof. Consider schedules $S^*$ and ($S'$). Schedule $S^*$ should be generated before the schedule ($S'$) since otherwise the schedule ($S'$) would have a conflict (remind that in $T$ we continue search from the first leftmost open schedule). From this we get: $ord(A(S^*), S) > ord(A(S'), S')$ and therefore $L(S', r)$ (Lemma 5). Since $S''$ is a successor it should have $L(S'', r) = L(S^*, r)$ unless it is the right in one of the schedules, generated between $S'$ and $S''$. Obviously, this might happen only if a critical path first is relocated to some block $B' < B$ and the job is forward (here $B \in S'$ is the block containing the overflow job $r$ of $S$). If in $S'$ we decrease “enough” of a critical path then a critical path will be added to the same block $B'$ and similarly in some $S^c$ of $S'$ job $r$ would be delayed as much as we can apply one of the inequalities $L(S'', r) \leq L(S^*, r)$, $L(S'', r) \leq L(S', r)$ with the inequality in the $S''$ and use a supercritical edge, then that fraction...
an optimal solution $S_{opt}$. With the root node of $T$ the extended Schrage schedule $S$, constructed for the initial data is associated. The successor nodes on the first level of $T$ represent the complementary schedules $S_l$, $l \in Kr_{S,u}$. The successor nodes on the second level of $T$ represent the complementary schedules $(S_l)_k$ where $S_l$ is a complementary schedule of the first level and $k \in Kr_{S_l,u}$, and so on. We test each generated schedule for the optimality (by Lemma 2) and close it if the conditions of one of the lemmas 6, 8, 10 are satisfied. Then we continue search from the leftmost open node, if an optimal schedule is not obtained, applying Lemma 12 (the Procedure Backtrack in the description below) and finally we stop when there is no more open node left in $T$.

Procedure Main;

Procedure Backtrack;

begin {backtrack}

Find the nearest open schedule $S' \in T$ (if there is some);

if $q_l(S') \geq q^c$ then (close $S'$; Backtrack) {Lemma 12}
else ($S := S'$; return)

if there is no open schedule in $T$ then stop

$S_{opt}$ is an optimal schedule

end {backtrack}

begin {main}

(0) $S :=$ Extended Schrage {Section 2}

$S_{opt} := S$; $q^c := +\infty$;

(1) Find the emerge sequence $C_{S,u}$ and the reduced set of emerge jobs $Kr_{S,u}$; $K := Kr_{S,u}$;

if $K = \emptyset$ then (close the schedule $S$; Backtrack); {Lemma 2}

(2) $S^c := \emptyset$;

while $K \neq \emptyset$ do

begin {while}

$l := \max \{ j | j \in K \}$; $K := K \setminus \{ l \}$;

if $q_l < q^c$ then $q^c := q_l$;

if $S_l$ is a well-defined nested complementary or a critical path in it is rested on $l$
then close $S_l$; {Lemmas 10, 6}

$S^c := S_l$;

end {while}

Backtrack;

end {main}.

Theorem 2. The time complexity of the algorithm $O(mn \log n)$ (under the assumption that the number of tails is bounded by the sufficiently large constant).

Proof. Suppose that $B \in B$ is a critical branch of the initial extended Schrage schedule $S$ and $r$ is the last job in it. First we estimate the number of nested complementary schedules a critical path can have (the alternative (c)).

Consider the complementary schedule $S_l$, $le(S_l) \in B_{S_l,r}$ in it. We claim that $q_j \leq q_r - 1$ (there can be scheduled no more than $m - 1$ number of machines) jobs in $S_l$ (different from which are started at time $t^*_l$ and have the tail equal to $q_r$. There can exist no job started in $S_l$ at a later and having the tail greater than $q_r$ since a critical path in $S$ would pass through this job). jobs with the tail equal to $q_r$ are scheduled job $r$ in $S_l$ since $r$ belongs to the rightmost critical path. Thus a critical path cannot be shifted forward among these jobs and we get that $q_j \leq q_r - 1$.

For the next complementary schedule $S' = Kr_{S,u}$, $le(S') = 2$ in which a critical path is shifted forward to job $j'$ ($j' > j$) we use the same reasoning and get that $q_{j'} \leq q_r - 2$, for the next job we get $q_{j''} \leq q_r - 3$ and so on. Thus the number of the jobs cannot exceed $q_r$.

Furthermore, the reduced set of emerge jobs do not contain no more than $q_r$ jobs (by the definition of the reduced set of emerge jobs implies that the number of complementary schedules in the first level of $T$ cannot be more than $q_r$ (Lemma 7). Since an emerge job of any schedule of the complementary set cannot have tail greater than $q_r - 1$, analogously to the proof of the previous section, we can conclude that the number of levels of the tree in which the $S_{opt}$ is not generated will not exceed $q_r$. Thus the number of levels will not exceed $O(mn \log n)$.
the level \( l \) will not be greater than \( q_r \) - \((l - 1)\). Suppose it exceeds \( q_r \) - \((l - 1)\). Then at least one schedule on the level \( l - 1 \), different from the leftmost schedule should have successors. Consider such a schedule, say \( S \). Since \( S \) was not closed, there is no schedule among those already generated, which active job has the tail equal to or less than that of the active job of the schedule \( S \) (Lemma 12). Consequently, there cannot exist a successor of the schedule \( S \) such that its active job has the tail equal to or more than that of the active job of any of the generated schedules (we remind that tails of active jobs are decreasing level by level). Thus, on level \( l \) we will have no schedules such that their active jobs have equal tails. Therefore, the number of schedules of level \( l \) will not exceed \( q_r \) - \((l - 1)\).

So, for the total number of schedules in \( T \) we get the bound

\[
 b = 1 + \sum_{i=0}^{q_{\text{max}}} (q_{\text{max}} - i) = O(q_{\text{max}}^2),
\]

\[
 q_{\text{max}} = \max \{q_i | i = 1, 2, ..., n\}.
\]

Now suppose that a critical path in a complementary schedule \( S \) is unmoved. From Lemma 11 we have that there can exist no more than \( mn \) nested complementary schedules of \( r \), the successors of \( S \). For the number of complementary schedules on each level of \( T \) we have the bound of the same order as for the alternative (c). Thus, an instances of the alternative (a) cause an additional factor of \( m \) in \( b \).

Let now in the complementary schedule \( S \in T \) a critical path be shifted backward. It is easy to see that a critical path cannot be shifted backward again in any of the direct successors of \( S \). The number of the direct successors of \( S \) is bounded by \( \delta_S \) (Lemma 9). So, the instances of the alternative (d) will cause an additional factor of \( p \) in \( b \).

Suppose that a critical path is relocated from the block \( B' \in S' \) to the block \( B'' \in S', S' \in T \) (the alternative (e)). Consider two different possibilities: \( B'' > B' \) and \( B'' < B' \). Case 1 \( (B'' > B') \). We apply a reasoning quite analogous to that which we used above in the case of the alternative (c) and get the bound \( O(q_{\text{max}}^2) \) on the number of relocations from one block to another successive block. Notice that this bound provides the instances of both alternatives (c) and (e, case 1) since these instances alternate. Case 2 \( (B'' < B') \). Again we can use a reasoning, similar to that which we used for the alternative (c) and obtain that the number of relocations easily get the above bound).

The number of repeated relocations of a critical job \( r \) cannot exceed \( m \) and these instances of the alternative (e) are covered by the bound of the alternative (a) (Lemma 11).

Thus, the alternative (c) may cause the number of \( O(q_{\text{max}}^2) \) nodes in \( T \). Instances of the alternative (e) cause an additional factors of the order \( O(mn) \) respectively. With each instance of the alternative (e) we close the corresponding schedule (Lemma 9). An instance of the alternative (e, case 2) causes the additional factor \( O(\bar{q}) \).

So, the resulting bound on the total number of \( T \) is:

\[
 O(m).O(q_{\text{max}}^2).O(p).O(\bar{q})
\]

(the constant factor \( O(p) \) can be excluded since the readiness times, tails and durations of \( p \) we obtain the equivalent problem with readiness times and tails and unit-length jobs).

For each node of \( T \) we construct an extended schedule (the time complexity is \( O(n \log n) \)) and the time \( O(n) \) to find an overflow job. We spend an amount of time to find the sets \( K_{r,S,p} \). Altogether we have the time complexity:

\[
 O(m).O(q_{\text{max}}^2).O(\bar{q}).(O(n \log n) + O(n)) = O(mn \log n)
\]

(from our assumption about the maximal job). Theorem is proved.

5 Concluding Remarks

The algorithm proposed improves the running time of the previously known best algorithms for the problem \( PF \) [Simons, 1983; Simons & Warm, 1987] under the assumption about the maximum and solves an extended minimization problem for the concept of forbidden regions. In fact, we solve...
number of machines and does not depend on the number of jobs, although we apply the greatest tail heuristic with the time complexity $O(n \log n)$ each time we generate a new ESS. Is it possible to generalize the presented algorithm to an algorithm with the same time complexity but without the restriction on the maximal job tail? This question is left open.

6 Appendix

We give five simple examples illustrating the behaviour alternatives (a) to (e).

The alternative (a):

<table>
<thead>
<tr>
<th>job</th>
<th>readiness</th>
<th>duration</th>
<th>tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

The initial extended Schrage schedule $S = (1, 2)$. Further, $L(S) = L(S, 2) = 0 + 3 + 3 + 6 = 12$; job 1 is the (only) emerging job; $S_1 = (2, 1)$; $L(S_1) = L(S_1, 2) = 1 + 3 + 6 = 10$ and the critical path is unmoved since $r(S) = r(S_1)$.

The alternative (b):

<table>
<thead>
<tr>
<th>job</th>
<th>readiness</th>
<th>duration</th>
<th>tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

The initial schedule $S = (1, 2)$, $L(S) = L(S, 2) = 0 + 3 + 3 + 6 = 12$; job 1 is the emerging job; $S_1 = (2, 1)$; $L(S_1) = L(S_1, 1) = 1 + 3 + 3 + 4 = 11$ an the critical path is rested on job 1 since $r(S_1) = 1$.

The alternative (c):

<table>
<thead>
<tr>
<th>job</th>
<th>readiness</th>
<th>duration</th>
<th>tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

The alternative (d):

<table>
<thead>
<tr>
<th>job</th>
<th>readiness</th>
<th>duration</th>
<th>tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

The initial schedule: $S = (1, 3, 2, 4)$; $L(S) = 0 + 3 + 3 + 3 + 4 = 16$; job 1 is the only emerging job; $S_1 = (2, 3, 4, 1)$; $L(S_1) = L(S_1, 3) = 2 + 3 + 3 = 8$. Job 3 is scheduled before job 2 in $S$ since at $t = 3$ of completion of job 1, both, jobs 2 and 3 are ready and $q_3 > q_2$. After rescheduling job 1, the gap $[0, 2]$ in $S_1$, job 2 is scheduled at the most critical path is shifted backward to job 3.

The alternative (e):

<table>
<thead>
<tr>
<th>job</th>
<th>readiness</th>
<th>duration</th>
<th>tail</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

The initial schedule $S = (1, 2, 3)$; $L(S) = 0 + 3 + 3 + 8 = 14$; job 1 is the only emerging job; $S_1 = (2, 1, 3)$; $L(S_1) = L(S_1, 3) = 8 + 3 + 2 = 13$. The critical path is relocated from the first block of jobs 1, 2 to the second block, containing

References


J. Cully 1980. "The two-machine"


*Nodari Vakhania* was born in Tbilisi, Republic of Georgia en 1961. He obtained his first scientific degree in Applied Mathematics at the Tbilisi State University in 1983. Later received his Ph.D. degree in Mathematical Cybernetics in 1991 from the Russian Academy of Sciences. He has worked as an assistant researcher at the department of Artificial Intelligence of the Computing Center of the Russian Academy of Sciences, Moscow. Since 1995 to 1996 he was an associate reseacher at IIMAS, UNAM. From 1996 he is an associate researcher at the Universidad Autónoma del Estado de Morelos, Cuernavaca. His main interest in research are discrete optimization problems, the design and analysis of algorithms, scheduling algorithms.