Computación y Sistemas Vol. 1 No. 4 pp. 239-250
© 1998, CIC - IPN. *ISSN 1405-5546* Impreso en México

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

Nodari Vakhania
Facultad de Ciencias
Universidad Autonoma 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 relatively new field, traditionally belonging to Operations Research, now also it significantly intersects with Computer Science. The combinatorial optimization problems are discrete optimization problems with 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 minimal (maximal) value of a goal function is found. Though the set of feasible solutions is finite, it might turn out that it is "too big", so that the complete enumeration would be practically impossible, since it would take inadmissible amount of machine time and memory. The dependence of the number of feasible solutions of a problem P on the length of input (the size of P) might be polynomial, as well as exponential. In the latter case we are not able to find an algorithm for P with polynomial dependence. The problems with polynomial dependence are typically much easier to solve than the problems with exponential dependence, since they take significantly less computational time. For a polynomial problem P, there may exist several algorithms with different degrees of the 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 problem of scheduling n equal-length jobs (or tasks) on m identical processors to minimize the total completion time of all jobs. Our algorithm has a smaller degree of the polynomial than the earlier ones under a realistic restriction which we impose on the problem data.

The sequencing problem $P1$ we conseder, can be stated as follows: There are given a set $I = \{1, 2, ..., n\}$ of *jobs* and a set $M = \{1, 2, ..., m\}$ of *machines* (or *processors*). Each job has to be performed on any of the given m ma-

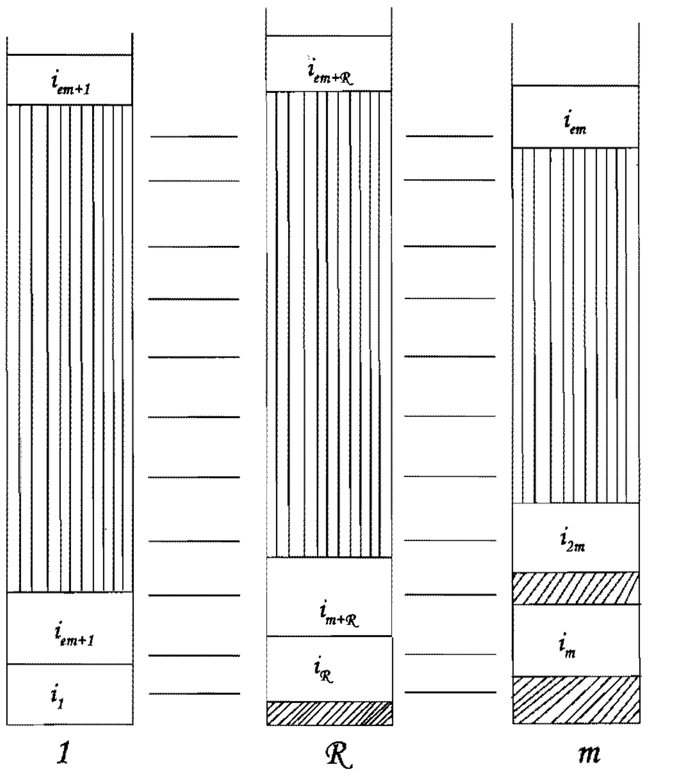

Figure 1: An ESS S.

chines; 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).

An alternate formulation of the above problem is the one with *due dates* (abbreviated as $P2$): instead of the tail q_i an integer due date d_i is given for each job i (d_i is the desirable time for completion of job i). The lateness L_i^S of job i in a schedule S is defined as:

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

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

If we allow in $P1$ or in $P2$ different processing times we get strongly NP-complete problem even in the single-machine case [Baker & Zaw, 1974; Bratley *et al.*, 1973; Carlier, 1982; Garey & Johnson, 1979; McMahon & Florian, 1975].

If we replace in $P2$ due dates with deadlines and look for a feasible schedule, we get the corresponding feasibility problem PF (by $PF1$ we abbreviate the one-machine version of PF). In a feasible schedule S of PF no job can be delayed, that is, $c_i^S \leq d_i$, for $i = 1, 2, ..., n$ (in a feasible schedule of $P2$ we allow the existence of such jobs and we look for a schedule which minimizes the maximum delay).

It has been proved that PF is solvable in polynomial time. An $O(n^2 \log n)$ algorithm for $PF1$ presented in [Carlier, 1981] is improved to an $O(n \log n)$ algorithm in [Garey *et al.*], 1981]. This algorithm applies a concept of the so called forbidden regions (that is a region in a schedule in which it is forbidden to start a job). Ones 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 of 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 instances. 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 a 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 $\mathbf{T}$ (we call it the *solution* or the *search* tree). We iteratively generate new feasible schedules which are represented by the nodes in $\mathbf{T}$. Each of the new generated schedule is obtained from the previously generated one by some specific rearrangement. Each feasible schedule we construct using the modification of the heuristic suggested by Luis Schrage for problem $P2$. According to this heuristic, the next scheduled job is a one which has the smallest due date (or, equivalently the greatest tail, for problem $P1$) among all available jobs:

Procedure Extended Schrage.

```
begin{extended schrage}

(0)  $t := \min\{a_i | i \in I\}$; $A := I$;
     $R(k) := 0$, $k = 1, 2, ..., m$;
     { $R_k$ is the release time of machine $k$ }

(1)  Among the unscheduled jobs $l \in A$ with $a_l \leq t$
     schedule next job $j$ with the greatest tail on machine
     $k = \text{ord}(j) \bmod m$ {break ties arbitrarily};
     {ord($j$) is the ordinal number of job $j$ in the current
     partial schedule}
     $t_j := \max\{t, R_k\}$; $R_k := t_j + p_j$; $A := A \setminus \{j\}$;
     $k' := (\text{ord}(j) + 1) \bmod m$; { $k'$ is the next available
     machine}
     if $A \neq \emptyset$ then $t := \max\{R_{k'}, \min\{a_i | i \in A\}\}$; go to
     (1)
     else Extended Schrage:= $\{t_j\}$ $(j = 1, 2, ..., n)$;

return;

end.{extended schrage}
```

Thus the above algorithm repeatedly determines the next scheduled job using the Extended Schrage heuristic and assigns it to the next machine (next to machine k, $k = 1, 2, ..., m-1$, is machine $k+1$ and next to machine m is machine 1). The schedules, generated by the application of this algorithm are called *extended Schrage schedules* (abbreviated ESS). An example of an ESS S is given in Figure 1. The ordinal number of job i in this schedule is denoted by ord(i, S) (i.e. i is ord(i, S)st scheduled job in S); we say that job i is occupying ord(i, S)st *slot* in S. We denote by $t(s, S)$ the starting time of sth slot in S, that is, the starting time of the job scheduled in this slot in S. Later we frequently use a short form $i > j$ for ord$(i, S) >$ ord(j, S), assuming for the simplicity that jobs in S are numbered consequently from 1 to n.

With the root of our solution tree $\mathbf{T}$ the feasible schedule, obtained by the application of the Estended Schrage algorithm to the initially given problem instance is associated (we call the schedule, thus obtained, the *initial* extended Schrage schedule). Each subsequent schedule we also obtain by the Extended Schrage algorithm, but 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* $\mathrm{ord}(j,S) > \mathrm{ord}(i,S)$, $t_j^S \geq t_i^S$.

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 $*$, represents a unique job with the same number. We have in G_S the set of (initial) arcs $(0,i)$, $(i,*)$, $i = 1,2,...,n$. With an arc $(0,i)$ ($(i,*)$, 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) ($i,j \notin \{0,*\}$) 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.

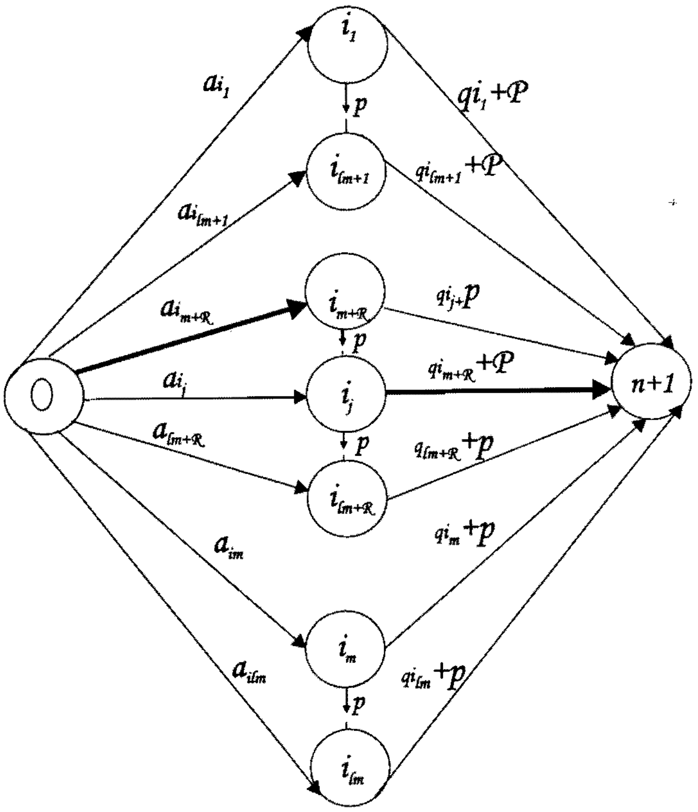

Figure 2: The Graph G_S.

Figure 2 shows that G_S contains m unconnected "gchains", each of them consisting of jobs scheduled successively on one particular machine. First such a chain consists of jobs scheduled on machine 1, the second such a chain consists of jobs scheduled on machine 2 and so on; the last chain consists of jobs scheduled on the last machine m. Each of these chains may contain one or more critical paths. Among all critical paths in G_S we distinguish the ones associated with the machine (or equivalently, with the chain) with the greatest index and call them the *rightmost* critical paths. Among critical paths of one particular chain we distinguish critical paths with the maximal number of jobs. The first such a path in a chain we call the *maximal* path. We will be further interested mainly in rightmost maximal paths (notice that for any schedule this path is defined uniquely).

A *gap* in a schedule is a time interval which is not occupied by any job. The greatest tail schedules consist of the sequence of one or more *blocks*. Intuitively, a block is an "isolated" part of a schedule. Any block, different from the last block (or equivalently, the first one, if they are the same) in any schedule contains the number of jobs which is multiple of m. Formally, a block is the maximal sequence of successively scheduled jobs on adjacent machines such that first m jobs in it are preceded by gaps or are earliest scheduled jobs on their respective machines, and the last m scheduled jobs ($k < m$ jobs, correspondingly) are succeeded by gaps (are latest scheduled jobs on their respective machines, correspondingly). For any two blocks B_1, $B_2 \in S$ either $B_1 > B_2$ or $B_1 < B_2$ holds, that is, either B_1 precedes B_2 in S or vise versa. In a given schedule, the *critical block* is the block containing the rightmost maximal path.

In Figure 3, the rightmost maximal path in G_S is represented. We call the last scheduled job of the rightmost maximal path in S the *overflow job* and usually denote it by r. Let B be the critical block in the ESS S. A job $l \in B$ such that $\mathrm{ord}(l) < \mathrm{ord}(r)$ is called an *emerge job* in S if $q_l < q_r$. We denote by $\mathbf{K}_{S,\mu}$ the set of all emerge jobs in S, where μ stands for the respective rightmost maximal path.

The sequence of jobs scheduled in S between l, the emerge job with the maximal ordinal number and the overflow job r (including this job) is called the *emerge sequence* and is denoted by $\mathbf{C}_{S,\mu}$. Notice that jobs of $\mathbf{C}_{S,\mu}$ 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 maximal) critical path in G_S and by $L(S,j)$ the length of a longest path to node j in G_S.

Let S be any Extended Schrage schedule and let $l \in \mathbf{K}_{S,\mu}$. The schedule obtained from S by rescheduling job l after all jobs of the emerge sequence $\mathbf{C}_{S,\mu}$ we call

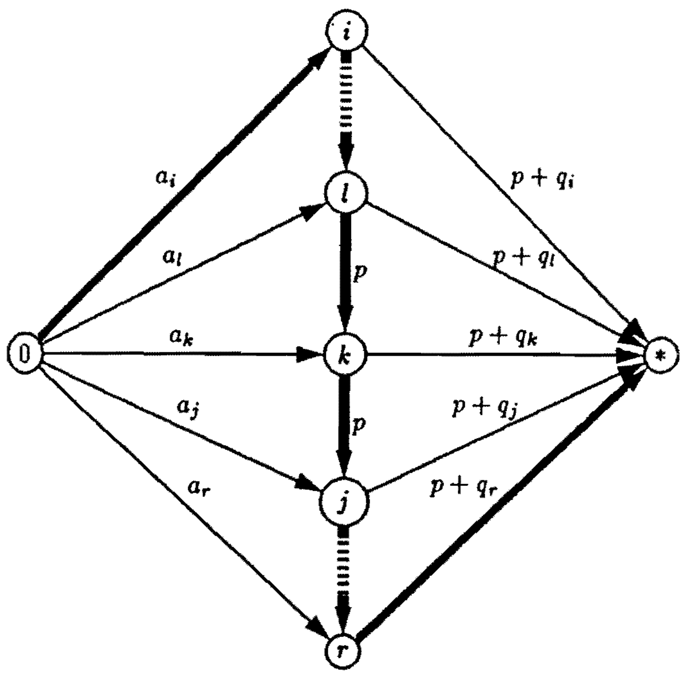

Figure 3: The fragment with the rightmost maximal path in G_S.

the *complementary schedule* and denote it by S_l. Complementary schedule S_l we obtain by application of the heuristic of ESS to the specially modified problem instance. We increase the readiness time of job l as well as some other jobs scheduled in S after the sequence $\mathbf{C}_{S,\mu}$ so that these jobs will be "forced" and scheduled after all jobs of $\mathbf{C}_{S,\mu}$ by the heuristic of the ESS. So, we leave an additional free space in S_l 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 $\mathbf{C}_{S,\mu}$ and the jobs scheduled before $\mathbf{C}_{S,\mu}$ in S, excluding job $l \in \mathbf{K}_{S,\mu}$; 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: $a_i := \max\{t_r^{S'}, a_i\}$ ($t_r^{S'}$ is said to be the *threshold value* for S_l). Now the complementary schedule S_l 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 schedules, that is, the consequences of rescheduling of an emerge job after the emerge sequence might be different. We distinguish five *behaviour alternatives* in a complementary schedule S_l. Let $r(S)$ denotes the overflow job in S. Then the critical path in S_l is said to be:

(a) unmoved, if $r(S_l) = r(S)$,

(b) rested on l, if $r(S_l) = l$,

(c) shifted forward,

(d) shifted backward, if $r(S_l)$ and $r(S)$ are in the same block ($r(S_l) \neq r(S)$, $r(S_l) \neq l$) and $\text{ord}(r(S_l), S_l) > \text{ord}(r(S), S_l)$ (respectively, $\text{ord}(r(S_l), S_l) < \text{ord}(r(S), S_l)$),

(e) otherwise, the critical path is said to be *relocated*, that is, $r(S_l)$ and $r(S)$ belong to different blocks.

All the alternatives except the last one are "local" for the current block: in the case of the instances of the first four alternatives we "stay" in the current block (making further the necessary rearrangement) while with the instance of the alternative five we "move" to another block (and make necessary rearrangement there). We again analyze the behaviour of the critical path in the newly generated complementary schedule and repeat the process. As it will be evident later, the impact of different alternatives on the complexity of our algorithm differs (alternatives (a), (b) will cause less computational efforts than instances of the other alternatives).

It can be easily seen that all five alternatives are attainable (see the Appendix for the examples). Clearly, the five alternatives are exhaustive (we can refer to one of them in any S_l): the overflow job in S_l may remain the same as in S (the alternative (a)) or change to l (the alternative (b)). Otherwise, either it can move to another block (the alternative (e)) or stay in the current block. For the latter case we have two possibilities: either $r(S_l)$ is scheduled after $r(S)$ in S_l (the alternative (c)) or it is scheduled before $r(S)$ (the alternative (d)). Thus, we have the following

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

3 Study of the Complementary Schedules

In this section we give the basic properties of the complementary schedules which we use later in our algorithm.

Lemma 1. *There arises at least one gap in any complementary schedule S_l between the* $(\text{ord}(l, S) - 1)$*st scheduled job and the overflow job r.*

Proof. Consider job j, $a_j = \min\{a_i | l < i \leq r\}$. Assume first that j is not an emerge job. Then $q_j > q_l$. This yields $a_j > t_l^S$ since otherwise job j would be scheduled at the moment t_l^S 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 $\mathbf{C}_{S,\mu}$ in S can occupy any interval before $\mathbf{C}_{S,\mu}$ in S_l. Therefore we will have a gap $[t_l^S, a_j)$ in S_l.

Now suppose j is an emerge job with $q_j \leq q_l$ and $a_j \leq t_l^S$ (if $a_j > t_l^S$ then we have a gap $[t_l^S, a_j)$). Again by the heuristic of the extended Schrage schedule and the definition of the complementary schedule, j will be ord(l, S)st scheduled job in S_l and (due to the equal processing times) we will have a gap in S_l strictly before (ord$(j, S) + 1$)th scheduled job if j is the only remained emerge job. If not, the next slot might be occupied by next emerge job. It is easy to see that there will be a gap in S_l 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 re-ordering jobs of the emerge sequence* $\mathbf{C}_{S,\mu}$.

Proof. Suppose that in the emerge sequence $\mathbf{C}_{S,\mu}$ 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: $a_l \leq t_m^S$ and $a_l > t_m^S$.

If $a_l \leq t_m^S$ 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 $a_l > t_m^S$ then we have a gap in $\mathbf{C}_{S,\mu}$ in the schedule S'. Again, job m can be scheduled before or after the overflow job r. In the first case we obviously have $L(S', r) \geq L(S, r)$. In the second case, $L(S', m) \geq L(S, r)$ (since $q_m \geq q_r$).◇

Let S be an extended Schrage schedule and $\delta_S = c_l^S - a_j$, where $l = \max\{i | i \in \mathbf{K}_{S,\mu}\}$, $a_j = \min\{a_i | i \in \mathbf{C}_{S,\mu}\}$.

Lemma 4. *The lower bound on the value of an optimal schedule is* $L(S) - \delta_S$.

Proof. $L = t_r^S + p + q_r$ is the makespan of S. We cannot improve this value by reordering jobs of the emerge sequence (Lemma 3). Therefore, the only possible way to improve it, is to reschedule some other jobs in such a way that jobs from $\mathbf{C}_{S,\mu}$ could start their processing earlier. Suppose we released some slots in S. Then none of the job $i \in \mathbf{C}_{S,\mu}$ can start its processing earlier than at time $t_i^S - \delta_S$ (by the definition of δ_S and the ESS). Thus the value $L(S) = L(S, r)$ can be decreased at most by δ_S and therefore $L(S) - \delta_S$ is the resulting lower bound.◇

Theorem 1. *An optimal schedule is amongst the complementary schedules.*

Proof. Consider any ESS S. We claim that if this schedule is not optimal then we can improve it only by generating complementary schedules. Then, coming out from the definition of a complementary schedule, we have to show that S cannot be improved by:

1. Rescheduling of any non-emerge job. This we have from Lemma 2;
2. Reordering the jobs of $\mathbf{C}_{S,\mu}$. We have this from Lemma 3;
3. Rescheduling an emerge job inside the emerge sequence. If we reschedule an emerge job inside the emerge sequence $\mathbf{C}_{S,\mu}$ then we can decrease $L(S)$ at most by δ_S (Lemma 4) while we increase it by p ($\delta_S < p$);
4. Reordering the jobs of a block different from the critical block. This case is obvious.◇

Let S be a complementary schedule with the rightmost maximal path μ. Consider the set of the complementary schedules S_l, $l \in \mathbf{K}_{S,\mu}$ and the magnitude, by which the length of μ is reduced in each of these schedules. As the following lemma shows, this magnitude may only decrease while we apply an emerge job which has an ordinal number, less than that of already applied emerge job.

Lemma 5. $L(S_l, r) \leq L(S_k, r)$ *if* $l > k$ $(l, k \in \mathbf{K}_{S,\mu})$.

Proof. Consider the complementary schedule S_k and the job, say j, which occupied ord(k, S)th slot in it. By the heuristic of the ESS we have $t_j^{S_l} \geq t_j^S$. The similar condition holds for all jobs which scheduled between job k and an overflow job r in S (in other words, the starting time of a job, scheduled in lth (ord$(k, S) \leq l \leq$ ord(r, S)) slot in S_k is more than or equal to that of in S). Analogously, the first late slot in S_l will be ord(l, S)th slot and we have ord$(l, S) >$ ord(k, S). This easily implies inequality of the form $t(s, S_l) \leq t(s, S_k)$, $s =$ ord(l, S), ord$(l, S) + 1, ..., r$ Now we get the lemma since in both S_l and S_k job r is scheduled in (ord$(r, S) - 1$)th slot.◇

Intuitively it should be clear that if a critical path in some complementary schedule is rested on the rescheduled 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 $S_l, l \in \mathbf{K}_{S,\mu}$, 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 is also emerge in S since $q_{l'} < q_l$. If $\text{ord}(l', S) > \text{ord}(l, S)$ then S_l can be neglected (this lemma, part 2). Let now $\text{ord}(l', S) < \text{ord}(l, S)$. Consider the nested complementary schedule $(S_l)_{l'}$. If $L((S_l)_{l'}, l') \geq L(S_l, l)$ 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 $q_l \leq q_k$ $(l, k \in \mathbf{K}_{S,\mu})$, $S \in T$ 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

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

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 $(S_k)_{k'}$, $k' \in \mathbf{K}_{S,\mu}$, $k' > k$ to improve the value $L(S_k, r)$. We have $(S_k)_{k'} = (S_{k'})_k$. If job k' is such that there is no $k'' \in \mathbf{K}_{S,\mu}$, $k'' > k'$ with $q_{k''} \leq q_{k'}$ then according the this theorem $(S_{k'})_k$ will be generated and $(S_k)_{k'}$ can be neglected. Otherwise, we recursively apply the similar reasoning to job k'' until we find a complementary schedule satisfying the conditions of the theorem.

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

Let now, in both S_l and S_k, a critical path be shifted forward. Consider the complementary schedules $(..(S_l)_l \ldots)_l$, $(..(S_k)_k..)_k$, obtained from the schedules S_l, S_k by rescheduling repeatedly jobs l, k respectively (as an emerge jobs). Observe that, if k is emerge in $(..(S_k)_k..)_k$, than l is also emerge in $(..(S_l)_l..)_l$ since $q_l \leq q_k$. Besides, the ordinal number of l in $(..(S_l)_l \ldots)_l$ is greater than or equal to the ordinal number of k in $(..(S_k)_k..)_k$ (again, because $q_l \leq q_k$). This again implies inequality of the form (*). The lengths of a critical paths in the considered schedules are decreasing step–by–step and the number of such schedules is bounded by the maximal tail (for details we refer to our proof of Theorem 2). As a result, we are brought either to the situation when the job k, or both l and k become non–emerge (these jobs cannot be further used for a schedule improvement), or to one of the situations considered above while for all intermediate complementary schedules inequalities of the form Eq. (*) are satisfied.

Suppose now that a critical path in S_l is shifted backward. A new arisen gap forces an order change of a couple of jobs in S_l; let j_l, j'_l be the corresponding couple of jobs in S_l. If the processing order of jobs j and j' in S_k is the same as in S_l (i.e. job j' precedes job j) then obviously the starting time of job j in S_k cannot be 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 $\text{ord}(j', S_l)$th slot is greater than that in S_l (by the heuristic of the ESS). Therefore the starting time of all consequent slots in S_k is greater than that in S_l (see Proposition 1) and again $L(S_k) \geq L(S_l)$. For any successor schedule of S_k we apply the reasoning which we already used for the different behaviour alternatives.

The alternative (e) obviously reduces to one of the alternatives (a) to (c). The lemma is proved.◇

We use Lemma 7 to reduce the set of emerge jobs. The subset of the emerge jobs $\mathbf{Kr}_{S,\mu}$ we will call a *reduced set* of emerge jobs, if for any pair $l, k \in \mathbf{Kr}_{S,\mu}$ such that $l < k$, we have $q_l < q_k$.

From Lemma 7 we obviously get the following:

Lemma 8. *Let $L(S_{l_1}, r) = L(S_{l_2}, r) = \ldots =$*

$L(S_{l_p}, r)$, $l_1, l_2, \dots, l_p \in \mathbf{K}_{S,\mu}$, and $q_{l_k} = \min\{q_{l_j} | j = 1, 2, \dots, p\}$. Then complementary schedules $S_{l_j}, j = 1, 2, \dots, p$ and $j \neq k$, can be neglected if the complementary schedule S_{l_k} is generated.

Lemma 9. *The number of complementary schedules, the direct successors of a particular complementary schedule S is bounded by δ_S.*

Proof. There are no more than δ_S possibilities to reduce the length of a critical path in S (by $1, 2, \dots, \delta_S$, 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 δ_S complementary schedules.◇

4 The Algorithm

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

Let $b_{\mathbf{T}}$ be any branch in $\mathbf{T}$ and let S be the first complementary schedule in $b_{\mathbf{T}}$ such that $r \in I$ is the overflow job in it. A complementary schedule $S' \in \mathbf{T}$, a successor of S, we call *lth 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 $b_{\mathbf{T}}$ 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 $\pi = \min\{|\mathbf{C}_{S,\mu}|, m\}$. We say that the nested complementary schedule of S, S' is *well-defined*, if any of the first π jobs of $\mathbf{C}_{S,\mu}$ is preceded by a gap in it.

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

Proof. From the definition of S' we have that first π jobs of $\mathbf{C}_{S,\mu}$ in this schedule are starting at their earliest starting times. Therefore the value $L(S', r)$ cannot be further improved (if $\pi \leq m$, this claim is obvious; if $\pi > m$, we apply Lemma 3). Thus we can close the schedule S'.◇

Lemma 11. *A nested complementary schedule of a level at most m is well-defined. Consequently, it might be closed.*

Proof. Consider S', a πth level nested complementary schedule of S. We claim that this schedule is well-defined. We get this claim from the definition of the extended Schrage and complementary schedules. Indeed, let j denotes a job from $\mathbf{C}_{S,\mu}$ with the minimal readiness time. In a nested complementary schedule of S of the first level job j will occupy the slot $\text{ord}(j, S) - 1$ and this job will be preceded by a gap (by the definition of a complementary schedule and Lemma 1). Analogously, in the next, nested complementary schedule of S of the level two, job j will occupy the slot $\text{ord}(j, S) - 2$ while the next job from $\mathbf{C}_{S,\mu}$ will occupy the slot $\text{ord}(j, S) - 1$ and both jobs will be preceded by a gap. Now, in S', a πth level nested complementary schedule of S, the first π jobs of $\mathbf{C}_{S,\mu}$ will be preceded by gaps. Thus, S' is a well-defined nested complementary schedule. Consequently, it can be closed (Lemma 10).◇

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

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

Proof. Consider schedules S^* and (S'). The schedule S^* should be generated before the schedule S' on the level $le(S^*)$ ($le(S)$ denotes the level of S in T) since otherwise the schedule (S') would have successors (remind that in $\mathbf{T}$ we continue search from the nearest leftmost open schedule). From this we get that $\text{ord}(A(S^*), S) > \text{ord}(A(S'), S')$ and therefore $L(S^*, r) \leq L(S', r)$ (Lemma 5). Since S'' is a successor of S^* we should have $L(S'', r) = L(S^*, r)$ unless job r is shifted 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 then shifted forward (here $B \in S'$ is the block containing the overflow job r of S). If in S' we decrease "enough" the length of a critical path then a critical path will be relocated to the same block B' and similarly in some successor S^c of S' job r would be delayed as much as in S''. So we apply one of the inequalities $L(S'', r) \leq L(S', r)$ or $L(S'', r) \leq L(S^c, r)$ with the inequality in the condition and use a reasoning analogous to that from the proof of Lemma 7 and complete this proof.◇

Now we are ready to give the description of our algorithm. It constructs the solution tree $\mathbf{T}$ containing 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, $l \in \mathbf{Kr}_{S,\mu}$. 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 $k \in \mathbf{Kr}_{S_l,\mu}$, 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 $q_{A(S')} \geq q^c$ 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 $\mathbf{C}_{S,\mu}$ and the reduced set
    of emerge jobs $\mathbf{Kr}_{S,\mu}$; $K := \mathbf{Kr}_{S,\mu}$;
    if $K = \emptyset$ then (close the schedule $S$; Backtrack);
    {Lemma 2}
    (2) $S^c := \emptyset$;
    while $K \neq \emptyset$ do
    begin{while}
    $l := \max\{j | j \in K\}$;  $K := K \setminus \{l\}$;

Construct the complementary schedule $S_l$;

  if $S^c \neq \emptyset$ & $L(S_l, r) = L(S^c, r)$ then close $S^c$ { Lemma 8}

  if $L(S_l) < L(S_{opt})$ then $S_{opt} := S_l$;

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

  if $S_l$ is a well-defined nested complementary schedule
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 is $O(mn \log n)$ (under the assumption that the maximal job tail is bounded by the sufficiently large constant C).*

Proof. Suppose that $B \in \mathbf{B}$ is a critical block in the initial extended Schrage schedule S and r is the overflow job in it. First we estimate the number of nodes in the solution tree **T** under the assumption that in all generated complementary schedules a critical path is shifted forward (the alternative (c)).

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

For the next complementary schedule $S' = (S_l)_{l'}$ ($l' \in \mathbf{Kr}_{S_l,\mu}$, $le(S') = 2$) in which a critical path is again shifted forward to job j' ($j' > j$) we use the analogous reasoning and get that $q_{j'} \leq q_r - 2$, for the next one we get $q_{j''} \leq q_r - 3$ and so on. Thus the number of levels in **T** will not exceed q_r.

Furthermore, the reduced set of emerge jobs $\mathbf{Kr}_{S,\mu}$ can contain no more than q_r jobs (by the definition). This implies that the number of complementary schedules of the first level of **T** cannot be more than q_r (see Lemma 7). Since an emerge job of any schedule of the first level cannot have tail greater than $q_r - 1$, analogously, we get that none of the schedules of the first level can have more than $q_r - 1$ successors, none of the schedules of the second level can have more than $q_r - 2$ successors and so on. Now we claim that the total number of schedules on 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 $\mathbf{T}$ we get the bound

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

$$q_{\max} = \max\{q_i | i = 1, 2, ..., n\}.$$

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 $m-1$ nested complementary schedules of r, the successors of S. For the number of complementary schedules on each level of $\mathbf{T}$ 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 $S \in \mathbf{T}$ 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 δ_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 $B'' \in S'$, $S' \in \mathbf{T}$ (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 $O(q_{\max})$ 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 from a block to its preceding block cannot be more than $\bar{q} = q_{\max} - q_{\min}$, $q_{\min} = \min\{q_i | i = 1, 2, ..., n\}$ (if a critical path is moving from job j to job j', $j' < j$ we should have $q_{j'} \geq q_j + 1$; from this inequality we can easily get the above bound).

The number of repeated relocations of a critical path to any 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 creation of $O(q_{\max}^2)$ nodes in $\mathbf{T}$. Instances of the alternatives (a),(d) cause an additional factors of the order $O(m)$ and $O(p)$, respectively. With each instance of the alternative (b) we close the corresponding schedule (Lemma 6). Each instance of the alternative (e, case 1) is covered by the bound of the alternative (c). The instances of the alternative (e, case 2) cause the additional factor of order $O(\bar{q})$.

So, the resulting bound on the total number of nodes in $\mathbf{T}$ is:

$$O(m).O(q_{max}^2).O(p).O(\bar{q})$$

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

For each node of $\mathbf{T}$ we construct an extended Schrage schedule (the time complexity is $O(n \log n)$) and spend time $O(n)$ to find an overflow job. We spend the same amount of time to find the sets $\mathbf{Kr}_{S,\mu}$. Altogether, we have the time complexity:

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

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

5 Concluding Remarks

The algorithm proposed improves the running time of the previously known best algorithms for the feasibility problem PF [Simons, 1983; Simons & Warmuth, 1989] under the made assumption about the maximal job tail, and solves an extended minimization problem $P1$. Algorithms from [Simons, 1983; Simons & Warmuth, 1989], as well as the one from [Garey *et al.*, 1981], are based on the concept of forbidden regions. In fact, we showed that an approach without the preliminary construction of the forbidden regions, what takes time $O(n^2)$, can be more efficient. In the solution tree $\mathbf{T}$, the number of nodes is bounded by the polynomial on the maximum tail and the number of machines and does not depend on the number of jobs, although we apply the greatest tail heuristic with the time complexity $O(nlogn)$ 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):

job	readiness	duration	tail
1	0	3	1
2	1	3	6

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):

job	readiness	duration	tail
1	0	3	4
2	1	3	6

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):

job	readiness	duration	tail
1	0	3	1
2	1	3	8
3	6	3	3

The initial schedule $S = (1,2,3)$, $L(S) = L(S,2) = 0+3+3+8 = 14$; job 1 is the emerging job; $S_1 = (2,1,3)$; $L(S_1) = L(S_1,3) = 1+3+3+3+3 = 13$ and the critical path is shifted forward to job 3.

The alternative (d):

job	readiness	duration	tail
1	0	3	0
2	2	3	6
3	3	3	9
4	7	3	4

The initial schedule: $S = (1,3,2,4)$; $L(S) = L(S,4) = 0+3+3+3+3+4 = 16$; job 1 is the only emerging job; $S_1 = (2,3,4,1)$; $L(S_1) = L(S_1,3) = 2+3+3+9 = 17$. Job 3 is scheduled before job 2 in S since at the time $t = 3$ of completion of job 1, both, jobs 2 and 3 are ready and $q_3 > q_2$. After rescheduling job 1, we get the gap $[0,2)$ in S_1, job 2 is scheduled at the moment $t = 2$ since job 3 is not ready at this moment; so, the critical path is shifted backward to job 3.

The alternative (e):

job	readiness	duration	tail
1	0	3	1
2	1	3	8
3	8	3	2

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

References

K.R. Baker and Zaw–Sing Su 1974."Sequencing with due dates and early start times to minimize maximum tardiness". *Naval Res. logist. Quart* 21, 171–177.

P. Bratley, M. Florian and P. Robillard 1973. "On sequencing with earliest start times and due–dates with application to computing bounds for $(n/m/G/F_{max})$ problem" *Naval Res. logist. Quart*20, 57–67.

J. Carlier (1981). "Problèmes d'ordonnancement à durées égales". Technical report, *Institut de Programmàtion, Université Paris*, IV–75012 Paris, France.

J. Carlier 1982. "The one–machine sequencing problem" *European J. of Operational Research.* 11, 42–47.

M.R. Garey, D.S. Johnson 1979. "Computers and

Intractability": A Guide to the Theory of NP–completeness, *Freeman, San Francisco.*

M.R. Garey, D.S., B.B. Johnson and R.E. Tarjan 1981. "Scheduling unit–time tasks with arbitrary release times and deadlines". *SIAM J. Comput.* 10, 256–269.

G. McMahon and M. Florian 1975. "On scheduling with ready times and due dates to minimize maximum lateness". *Operations Research.* 23, 475–482.

B. Simons 1983. "Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines". *SIAM J. Comput.* 12, 294-299.

B. Simons, M. Warmuth 1989. "A fast algorithm for multiprocessor scheduling of unit-length jobs". *SIAM J. Comput.* 18, 690-710.

N. Vakhania 1997. "Sequencing with readiness times and tails on parallel machines". *Proc. of the Twelfth ACM Conference on Applied Computing*, 438-446.

__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 recived his Ph.D. degree in Mathematical Cybernetics in 1991 from the Russian Academy of Sciences. He has worked as an assitent 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 algorthms, scheduling algorithms.