J Sched manuscript No.
(will be inserted by the editor)

Single processor scheduling with time restrictions

O. Braun · F. Chung · R. Graham

Received: date / Accepted: date

Abstract We consider the following list scheduling problem. We are given a set S of jobs which are to be scheduled sequentially on a single processor. Each job has an associated processing time which is required for its processing. Given a particular permutation of the jobs in S, the jobs are processed in that order with each job started as soon as possible, subject only to the following constraint: For a fixed integer $B \geq 2$, no unit time interval $[x, x+1)$ is allowed to intersect more than B jobs for any real x. It is not surprising that this problem is NP-hard when the value B is variable (which is typical of many scheduling problems).

There are several real world situations for which this restriction is natural. For example, suppose in addition to our jobs being executed sequentially on a single main processor, each job also requires the use of one of B identical subprocessors during its execution. Each time a job is completed, the subprocessor it was using requires one unit of time in order to reset itself. In this way, it is never possible for more than B jobs to be worked on during any unit interval.

In this paper we carry out a classical worst-case analysis for this situation. In particular, we show that any permutation of the jobs can be processed within a factor of $2 - 1/(B-1)$ of the optimum (plus an additional small constant) when $B \geq 3$ and this factor is best possible. For the case $B = 2$, the situation is rather different, and in this case the corresponding factor we establish is $4/3$ (plus an additional small constant), which is also best possible. It is fairly rare that best possible bounds can be obtained for the competitive ratios of list scheduling problems of this general type.

Keywords Single Processor Scheduling · Time Restrictions · Worst-Case Analysis

1 Introduction.

We are initially given a set $S = \{S_1, S_2, \ldots, S_n\}$ of *jobs* and an integer $B \geq 2$. Each job S_i has associated with it a *length* $s_i \geq 0$. The jobs are all processed sequentially on a single processor. A job S_i will be processed during a (semi-open) time interval $[\alpha, \alpha + s_i)$ for some $\alpha \geq 0$. A special type of job S_i, called a *zero-job* Z_i, has length $s_i = 0$. By convention, each zero-job is processed at some particular point in time. Furthermore, only one job can be worked on at any point in time except in the case of zero-jobs, where it is allowed for several zero-jobs to be processed at the same point in time, provided the constraint (1) below is not violated.

Given some permutation π of S, say $\pi(S) = T = (T_1, T_2, \ldots, T_n)$ with corresponding job lengths $(t_1, t_2, \ldots, t_n)$, the jobs are placed sequentially on the real line

Research supported in part by grants DFG BR 2272/7-1 (O. Braun), ONR MURI N000140810747, and FA9550-09-1-0900 (F. Chung), NSF 10050729 (R. Graham)

O. Braun
Environmental Campus Birkenfeld, Trier University of Applied Sciences, E-mail: o.braun@umwelt-campus.de

F. Chung
University of California, San Diego
E-mail: fan@ucsd.edu

R. Graham
University of California, San Diego
E-mail: graham@ucsd.edu

as follows. The initial job T_1 in the list T begins at time 0 and finishes at time t_1. In general, T_{i+1} begins as soon as T_i is completed, provided the following B-constraint is always observed. The constraint that differentiates our problem from many similar ones in the literature (e.g., see [4]) is this:

$$\text{For every real } x \geq 0, \text{ the unit interval } [x, x+1) \text{ can intersect at most } B \text{ jobs.} \quad (1)$$

Thus, if it happens that some job T_i finishes at time t and $\sum_{j=1}^{B-1} t_{i+j} \leq 1$, then the job T_{i+B} cannot start until time $t+1$ (since if it started strictly before this time, then for a suitable $\delta > 0$, the interval $[t-\delta, t+1-\delta)$ would intersect more than B jobs). Constraint (1) reflects the condition that each job needs one of B additional ressources for being processed and that a ressource has to be renewed after the processing of a job has been finished.

The preceding procedure results in a unique placement (or *schedule*) of the jobs on the real line. We define the *finishing time* $\tau(T)$ to be the time at which the last job T_n is finished. A natural goal might be for a given job set S, to find those permutations $T = \pi(S)$ which minimize the finishing time $\tau(T)$. In Section 2 and Section 3, our focus will be on a worst-case analysis of the problem. That is, how much worse can an arbitrary schedule be compared to an optimal schedule for a given set S of jobs. We will show in Section 4 that the problem is NP-hard in general.

Although this simple scheduling model has potential applicability to a number of application areas such as crew scheduling, flexible manufacturing systems, printed circuit board assembly, and cardinality constrained scheduling, for example, this is the first time that a precise worst-case analysis of it has been carried out (e.g., see [4]).

Let us denote by $\tau_w(S)$ the largest possible finishing time for any permutation of S, and let $\tau_o(S)$ denote the optimal (i.e., the shortest possible) finishing time for any permutation of S. We are interested in seeing how much worse $\tau_w(S)$ can be compared to $\tau_o(S)$. In what follows, we will prove the following worst-case bounds:

Theorem 1 *For $B = 2$ and any set S,*

$$\tau_w(S) - \frac{4}{3}\tau_o(S) \leq 1. \quad (2)$$

The factor $\frac{4}{3}$ is best possible.

Theorem 2 *For $B \geq 3$ and any set S,*

$$\tau_w(S) - \left(2 - \frac{1}{B-1}\right)\tau_o(S) \leq 3. \quad (3)$$

The factor $2 - \frac{1}{B-1}$ is best possible.

It is interesting that for this problem, there is a significant difference between the cases $B = 2$ and $B \geq 3$. We first make some preliminary remarks that will apply in the remainder of the paper. We are interested in upper-bounding the expression

$$\Delta_B(S) = \{\tau_w(S) - c_B\tau_o(S)\}$$

where

$$c_B = \begin{cases} \frac{4}{3} & \text{if } B = 2, \\ 2 - \frac{1}{B-1} & \text{if } B \geq 3, \end{cases}$$

and $S = \{S_1, S_2, \ldots, S_n\}$ is some arbitrary set of jobs. Thus, we can assume that all the lengths s_i of the S_i satisfy $s_i \leq 1$ for all i, since if any S_i had $s_i = 1+\epsilon > 1$, then by decreasing s_i to 1, we decrease both $\tau_w(S)$ and $\tau_0(S)$ by ϵ, thereby increasing $\Delta_B(S)$.

We should also keep in mind that in a schedule, it is possible that a job S_i finishes at some time t, and then j zero-jobs are processed at this time t for some $j \leq B-2$ (which takes time 0), and then another job S_k immediately begins its processing at time t.

2 The case $B = 2$.

Our goal in this section will be to prove (2). Let us examine the schedule for the optimal permutation for S, which we will assume (by relabeling if necessary) to be $(S_1, S_2, \ldots, S_n)$. Since $B = 2$, then by (1) we have:

$$\tau_o(S) \geq s_1 + 1 + s_3 + 1 + \cdots + s_{2k-1} + 1 + \cdots,$$

and also

$$\tau_o(S) \geq s_1 + s_2 + 1 + s_4 + 1 \cdots + s_{2k} + 1 + \cdots.$$

Consequently, it is not hard to check that this implies

$$2\tau_o(S) \geq s_1 + s_n + \sum_{i=1}^{n} s_i + (n-2),$$

which in turn implies

$$\tau_o(S) \geq \frac{1}{2}\left(\sum_{i=1}^{n} s_i + (n-2)\right). \quad (4)$$

Case 1: $\sum_{i=1}^{n} s_i \geq \frac{n+1}{2}$. Because of $\tau_w(S) \leq n$ (note that we only have jobs with lengths ≤ 1) and with (4) we come to

$$\Delta_2(S) = \tau_w(S) - \frac{4}{3}\tau_o(S) \leq n - \frac{2}{3}\left(\sum_{i=1}^{n} s_i \ + n - 2\right)$$
$$\leq n - \frac{2}{3}\left(\frac{n+1}{2} \ + n - 2\right) = 1.$$

Case 2: $\sum_{i=1}^{n} s_i < \frac{n+1}{2}$. In the schedule for S, let $s(S_i)$ and $f(S_i)$ denote the starting times and finishing times, respectively, of S_i (so that $f(S_i) - s(S_i) = t_i = |S_i|$). In particular, because of the constraint 1), we must have $s(T_3) - f(T_1) = 1$ (we say that there is a *jump* from T_1 to T_3), $s(T_5) - f(T_3) = 1$, and so on. Also we must have $s(T_4) - f(T_2) = 1, s(T_6) - f(T_4) = 1$, and so on. In any case, we can see that there are at least $\frac{n-1}{2}$ *jumps* from a job i to a job $i+B = i+2$. Therefore we have for the makespan of any schedule

$$\tau_w(S) \leq \sum_{i=1}^{n} s_i \ + \frac{n-1}{2}. \tag{5}$$

With (4) and (5) we come to

$$\Delta_2(S) = \tau_w(S) - \frac{4}{3}\tau_o(S)$$
$$\leq \sum_{i=1}^{n} s_i \ + \frac{n-1}{2} - \frac{2}{3}\left(\sum_{i=1}^{n} s_i \ + n - 2\right)$$
$$= \frac{1}{3}\sum_{i=1}^{n} s_i \ + \frac{5-n}{6} < \frac{1}{3}\frac{n+1}{2} \ + \frac{5-n}{6} = 1.$$

This proves Theorem 1. □

To see that this bound is best possible, consider the set S consisting of $2t+1$ jobs S_i of length 1 and $2t$ zero-jobs Z_j, for some positive integer t. The (optimal) permutation $(Z_1, S_1, S_2, \ldots, S_{2t+1}, Z_2, Z_3, \ldots Z_{2t})$ has a finishing time of $\tau_o(S) = 3t$. On the other hand, the (worst) permutation $(S_1, Z_1, S_2, Z_2, \ldots, S_{2t}, Z_{2t}, S_{2t+1})$ has a finishing time of $\tau_w(S) = 4t+1$. Thus,

$$\tau_w(S) - \frac{4}{3}\tau_o(S) = 4t + 1 - \frac{4}{3}\cdot 3t = 1,$$

achieving equality in (2).

3 The case $B \geq 3$.

Our next goal will be to establish (3). As before, we will assume that the permutation $(S_1, S_2, \ldots, S_n)$ is the optimal permutation for the set of jobs $S = \{S_1, S_2, \ldots, S_n\}$, so that this permutation has finishing time $\tau_o(S)$. We now want to create an auxiliary job set $T = \{T_1, T_2, \ldots, T_n\}$ as follows. The size of $t_i = |T_i|$ will be defined by:

$$t_i = \begin{cases} \frac{1}{B-1} & \text{if } s_i \leq \frac{1}{B-1}, \\ s_i & \text{if } s_i > \frac{1}{B-1}. \end{cases}$$

Observe, that for *any* permutation T' of T,

$$\tau(T') = \tau(T) = \sum_{i=1}^{n} t_i \geq \tau_w(S). \tag{6}$$

We next examine the schedule for S. Let us replace each job S_i that has $s_i \leq \frac{1}{B-1}$ by a zero-job S_i' placed on the time axis at the starting time of S_i (the lengths of the jobs with $s_i > \frac{1}{B-1}$ remain unchanged). This certainly causes no violations of the B-constraint (1). We will now assign a *weight* of size $\frac{1}{B-1}$ to each zero-job in this modified job set S'. Thus, the schedule for S' consists of the original "large" jobs S_i of length $s_i > \frac{1}{B-1}$ and a number of "point masses" of weight $\frac{1}{B-1}$, all placed so that condition (1) still holds, i.e., no unit interval intersects more than B of these jobs. The sum of all the weights of the jobs in S' is just equal to the sum of all the lengths of the jobs in T, which by (6) is an upper bound on $\tau_w(S)$. Also (since some of the jobs have been replaced by zero-jobs), all of the jobs in S' still fit in the interval $[0, \tau(S))$.

So we have reduced our problem to that of finding an upper bound W on the total weight of an arbitrary assignment of (semi-open) intervals of length $s_i > \frac{1}{B-1}$ and point masses of weight $\frac{1}{B-1}$ so that condition (1) is satisfied (and, of course, so that positive length intervals are disjoint, and point masses can only intersect a positive interval at its starting point). In particular, we can conclude that $\tau_w(S) \leq \tau(T) \leq W$.

Let us now define the *blocks* $U_i = [i, i+1)$ for $0 \leq i \leq N-1$, where $N = \lceil \tau_o(S) \rceil$.

Also, for each U_i we define its $B-1$ *subblocks* $U_{i,j} = [i + \frac{j-1}{B-1}, i + \frac{j}{B-1})$ for $1 \leq j \leq B-1$. We want to study how the weight from the jobs in S' is distributed in the U_i. Define the *content* $c(U_i)$ to be the sum of all the weight (or "mass") in U_i. In other words, we add up all the contributions of the point masses in U_i together with all the portions of those S_k' that happen to lie within U_i.

We will use the abbreviation $\theta := \frac{1}{B-1}$ since this quantity will occur so frequently in what follows.

Our next task will be to analyze various possibilities for the U_i. Let us say that the interval U_i is *bad* if $c(U_i) > 2 - \theta$. Otherwise, we will say that U_i is *good*.

First, suppose that U_i contains B zero-jobs, each contributing θ to $c(U_i)$. Then U_i cannot intersect any other jobs in S' and we have

$$c(U_i) = B\,\theta \le 2 - \theta$$

when $B \ge 3$, so that U_i is good in this case.

Next, suppose that U_i contains j zero-jobs for some j with $0 \le j \le B-2$. Then

$$c(U_i) \le j\,\theta + 1 \le (B-2)\,\theta + 1 \le 2 - \theta$$

and, again, U_i is good.

Hence, any bad U_i must contain exactly $B-1$ zero-jobs. In addition, it must also intersect exactly one other job in S' in an interval I_i (called its S-interval) with weight (= length) greater than $1-\theta$, in order that its total mass (i.e., content) is greater than $2-\theta$.

The plan is now to show that if we add up the contents of all of the U_i, the sum will be at most $(2-\theta)\tau(S)+B\theta$. We will do this by keeping a running total of the total mass accumulated as we proceed from the beginning block U_0 and move to U_1, U_2, etc., until we reach the final block U_{N-1}. We would like to show that the average mass in the blocks is at most $2-\theta$ (plus a small increment). While any good block has mass (= content) at most $2-\theta$, every now and then we may have a bad block, which could have content as much as 2. Our goal is to show that having too many bad blocks will always force there to be a compensating number of *very good* blocks, i.e., blocks have content a lot less than $2-\theta$, and this will keep the accumulated average mass total close to $2-\theta$.

First, observe that for any bad U_i, since its S-interval I_i has length exceeding $1-\theta$, then it must intersect all of the $B-1$ subblocks $U_{i,j}$ of U_i. Hence, each of the $B-1$ zero-jobs in U_i must lie in either the first or last subblocks, i.e., in $U_{i,1} \cup U_{i,B-1}$. Let us say that a block U_i has *rank* r if it intersects exactly r jobs of S' in its last subblock $U_{i,B-1}$. We will denote this by writing $\rho(U_i) = r$.

Let us now examine the relationship between two consecutive blocks U_i and U_{i+1} that have $\rho(U_i) = s$ and $\rho(U_{i+1}) = t$, respectively, where we assume that $s > t$. Since $\rho(U_i) = s$ then because of condition (1), the number of jobs which can intersect any of the first $B-2$ subblocks of U_{i+1} is at most $B-s$. However, since $\rho(U_{i+1}) = t$, then it is not hard to see that the content $c(U_{i+1})$ of U_{i+1} satisfies

$$\begin{aligned} c(U_{i+1}) &\le (B-s-1)\theta + (t-1)\theta + 1 \\ &= 2 - \theta - (s-t)\theta. \end{aligned}$$

(This can be seen by considering the possible contributions C of the non-zero-jobs to the content of U_{i+1}. If $C = 0$, then $c(U_{i+1}) \le (B-s+t)\theta \le 2-\theta-(s-t)\theta$ for $B \ge 3$. On the other hand, if $0 < C \le 1-\theta$, then either $c(U_{i+1}) \le 1-\theta+(B-s-1)\theta+t\theta$ if the non-zero-jobs in U_{i+1} intersect the first $B-2$ subblocks of U_{i+1}, or $c(U_{i+1}) \le 1-\theta+(B-s)\theta+(t-1)\theta$ if the non-zero-jobs in U_{i+1} intersect only the last subblock of U_{i+1}. Finally, if $C > 1-\theta$, then the last subblock of U_{i+1} can contain at most $t-1$ zero-jobs, so that $c(U_{i+1}) \le 1+(B-s-1)\theta+(t-1)\theta$, as claimed.)

In other words, if the ranks of two consecutive blocks drop by $s-t > 0$, then the second block has content at least $(s-t)\theta$ below the target value $2-\theta$. On the other hand, if $s \le t$, then the content of U_{i+1} might well be 2 (which is θ above the desired average value of $2-\theta$).

Let us examine the situation for a bad block $U_i, i > 0$, with rank s (so, in particular, $s > 0$). Thus, there are $s-1$ point masses in the last subblock of U_i (since its S-interval I_i intersects *all* the subblocks of U_i). Consequently, the first subblock of U_i must have $B-s$ point masses in it (since U_i is bad), and so, intersects $B-s+1$ jobs of S'. This means that the *preceding* block U_{i-1} can have rank at most $s-1$.

We now consider the sequence of ranks $\rho(U_i), i = 0, 1, 2, \ldots$. We will also consider the corresponding loss or gain with respect to our "average" value $2-\theta$ we make as we encounter each new block. As we have seen, if the rank in going from one block to the next one decreases by m then the content of the second block is at least $m\theta$ below the value $2-\theta$. On the other hand, if it increases, then the content of the second block might only be as much as θ above the value $2-\theta$. If the rank doesn't change, then we can't conclude that the content of the second block is "deficient", i.e., below $2-\theta$.

However, suppose that we have a sequence of blocks $U_j, U_{j+1}, \ldots, U_k$, where the only bad blocks in the sequence are the first and last blocks, and both blocks have the same rank s. Then by the preceding remark, the block U_{k-1} preceding U_k has rank at most $s-1$. Hence, in the sequence of ranks $\rho(U_j), \rho(U_{j+1}), \ldots, \rho(U_k)$, there will be a pair of consecutive blocks for which the rank drops by at least 1.

Putting all these facts together, we can conclude that the total gain over the average value $2-\theta$ as we sum up the cumulative content of the complete sequence of blocks is at most $B\theta$. That is, the total weight

$\sum_{i=0}^{N-1} c(U_i)$ of the blocks of S' satisfies

$$\begin{aligned}\tau_w(S) \le \tau(T) &= \sum_{i=1}^{n} t_i = \sum_{i=0}^{N-1} c(U_i) \\ &\le N(2-\theta) + B\theta \\ &\le \lceil \tau_o(S) \rceil (2-\theta) + B\theta \\ &< (\tau_o(S)+1)(2-\theta) + B\theta \\ &\le \tau_o(S)\left(2 - \frac{1}{B-1}\right) + 3.\end{aligned}$$

This proves Theorem 2. □

To see that the constant factor $2-\frac{1}{B-1}$ is best possible, we consider the job set S consisting of $(B-1)t+1$ jobs of length 1 and $(B-2)((B-1)t+1)+B$ zero-jobs for a positive integer t (t must be large enough). It is then not hard to see that the optimal permutation for S is $B-1$ jobs of size 0, followed by a job of length 1 followed by $B-2$ zero-jobs which is repeated for $(B-1)t+1$ times, followed by a zero-job. We denote this by

$$0^{B-1}\ 1\ 0^{B-2} \ldots 1\ 0^{B-2}\ 0 = 0^{B-1}\ [1\ 0^{B-2}]^{(B-1)t+1}\ 0$$

that has a finishing time $\tau_o(S) = (B-1)t+1$. On the other hand, the permutation

$$1\ 0^{B-1} \ldots 1\ 0^{B-1}\ 1 \ldots 1 = [1\ 0^{B-1}]^{(B-2)t+2}\ 1^{t-1}$$

has finishing time $(2B-3)t+3 \le \tau_w(S)$. Hence,

$$\begin{aligned}&\tau_w(S) - \tau_o(S)\left(2 - \frac{1}{B-1}\right) \\ \ge\ &(2B-3)t+3 - ((B-1)t+1)\left(2 - \frac{1}{B-1}\right) \\ &= \frac{B}{B-1}.\end{aligned}$$

This shows that the constant 3 in (3) is not too far from the truth. In fact, we believe that the theorem should be true when the constant 3 is replaced by $\frac{B}{B-1}$, which because of the example above, would be the best possible value for the constant.

4 NP-hardness of the general problem

In this section we will show that the determining whether there is a "perfect" schedule for S, (i.e., so that $\tau(S)$ is equal to the sum of all the job lengths) is an NP-hard problem, when the bound B is a variable parameter of the problem. We will polynomially reduce the NP-hard problem PARTITION (see [1]) to a special case of our scheduling problem. Recall that for PARTITION, we are given a set $T = \{t_1, t_2, \ldots, t_r\}$ of positive integers with an even sum $2R$ and we are asked to decide if there is a partition of $T = T_1 \cup T_2$ such that each subset T_i has a sum R.

Let us first choose an integer X slightly larger than $\sum_{i=1}^{r} t_i$. Now define a set of $2r$ jobs $S = \{S_1, S_2, \ldots S_{2r}\}$ where S_i has length $s_i = |S_i| = 10X$ for $1 \le i \le r$, and $s_{i+r} = |S_{i+r}| = 10(X + t_i)$ for $r+1 \le i \le 2r$. Further, define a set $E = \{E_1, E_2, E_3, E_4\}$ of four jobs each having length $|E_i| = 1$.

It follows from the choice of X that for any t, the sum of any $t+1$ s_i's is greater than the sum of any t s_i's. In particular, any partition of $\{s_1, s_2, \ldots, s_{2r}\}$ into two subsets with equal sums can only happen if the two subsets each have r elements.

We are going to choose $B = r+2$ and scale up our problem so that we will require that no interval $[0, L)$ of length $L = \frac{1}{2}\sum_{i=1}^{2r} s_i = 10rX + 5\sum_{i=1}^{r} t_i$ can intersect more than B jobs.

Suppose now that $S \cup E$ has a "perfect" schedule, say $S_{i_1}, S_{i_2}, \ldots, S_{i_{2r}}$, where the E_i are also intermixed in with the S_i. Assume without loss of generality that the indices of the E_i increase from 1 to 4 as we go from left to right. Since the schedule is perfect, the finishing time will be $2L+4 := \sum_{i=1}^{2r} s_i + 4 = 20rX + 10\sum_{i=1}^{r} t_i + 4$. Define $\sigma_1 = \sum_{k=1}^{r} s_{i_k}$ and $\sigma_2 = \sum_{k=r+1}^{2r} s_{i_k}$.

First, suppose that $\sigma_1 < \sigma_2$. Then, in fact, $\sigma_1 \le \sigma_2 - 20$. Since $\sigma_1 + \sigma_2 = 2L$ then we see that $\sigma_1 \le L - 10$. Consequently, if E_1 and E_2 came *before* $S_{i_{r+1}}$ then the sum of the lengths of the first r jobs together with the two (or more) E_i is still less than L, so that the interval $[0, L)$ intersects more than $B = r+2$ jobs (since it also intersects $S_{i_{r+1}}$). On the other hand, if E_2, E_3 and E_4 come *after* $S_{i_{r+1}}$ then it is easy to see that there is again an interval of length L which intersects more than B jobs (since the sum of the last $r-1$ s_i's is less than the sum σ_1 of the first r s_i's). A similar argument applies if $\sigma_1 > \sigma_2$.

Thus, if there is to be a perfect schedule for $S \cup E$, then we must have $\sigma_1 = \sigma_2$. But as we have seen, this implies that we have exactly r terms in each sum. This in turn implies that each sum has a contribution of $10rX$ in addition to the sum of various $10t_i$ tems. Canceling out the $10rX$ terms from each side, we have deduced the existence of a valid solution to the original PARTITION problem. This completes the reduction of PARTITION to a special case of our scheduling problem.

5 Concluding remarks.

It is interesting that our bounds are similar in spirit to some of the bounds in the very early list scheduling literature ([2], [3]). One would suspect that there are polynomial-time algorithms with which one can improve the bounds in our theorems. However, we have not looked at this problem yet. There are also a number of variations and generalizations which might be interesting to investigate. For example, what if there is more than one processor, and/or more than one type of constraint. An example would be a setting where not only no unit interval can intersect more than B jobs, but also no set of jobs that together use more than a certain amount of some resource (such as power) can be processed at the same time (in the multi-processor case). It would also be interesting to know if our problem remains NP-hard if $B \geq 2$ is fixed.

Acknowledgements

The authors would like to thank David Johnson and the referees for helpful suggestions.

References

1. Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W. H. Freeman.
2. Graham, R. L. (1966). Bounds on certain multiprocessing anomalies. *Bell System Technical Journal, 45*, 1563–1581.
3. Graham, R. L. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics, 17*, 416–429.
4. Leung, J. Y. - T. (Ed.) (2004). *Handbook of Scheduling.* Chapman and Hall.