Semi-Online Preemptive Scheduling: Study of Special Cases

Tomáš Ebenlendr*

March 21, 2010

Abstract

We use the duality of linear programing to obtain exact formulas of competitive ratio for the semi-online preemptive scheduling on three and four machines. We use the linear programs from [5]. Namely we consider the online scheduling and the semi-online scheduling with known sum of processing times. We solve the linear programs symbolically by examining all basic solution candidates. All solutions are obtained by a computer but all are verifiable by hand.

1 Introduction

We study the scheduling on *uniformly related machines*. Every machine has its speed s, i.e., processing a job with processing time p in a machine with speed s takes p/s time. *Preemption* is allowed: each job may be divided into several pieces, these pieces can be assigned to different machines in disjoint time slots. The objective is to minimize the *makespan*, i.e., the length of a schedule. In the *online* problem, jobs arrive one-by-one and we need to assign each incoming job without any knowledge of the jobs that arrive later. When a job arrives, its assignment at all times must be given and we are not allowed to change this assignment later. In other words, the online nature of the problem is given by the ordering of the input sequence and it is not related to possible preemptions and the time in the schedule.

The online algorithms are evaluated by their competitive ratio, that is the worst-case ratio between the makespan of the output schedule of the algorithm and the optimal (offline) makespan. I.e., the r-competitive algorithm produces at most r times longer schedule than the best possible schedule for every input sequence. *Semi-online* problems are derived from the original *online* problem by providing some partial information in advance. The competitive ratio of studied problems is given by several linear programs, their dimension depends on number of machines quadratically. We deploy a method how to solve small

*Institute of Mathematics, AS CR, Žitná 25, CZ-11567 Praha 1, Czech Republic.Email: ebik@math.cas.cz.

parametrized linear programs. We obtain exact competitive ratios for small numbers of machines (up to 4) for various semi-online scheduling problems this way.

This method is based on duality of linear programing, which says that it suffices to examine all possible basic solutions. From a geometrical point of view, we take all intersections of dimension zero of hyperplanes defined by the linear conditions, and then we test if such an intersection is a feasible and optimal solution. Note that the feasibility and the optimality of such a solution also depends on the actual values of the parameters, so the result typically splits into several cases. Searching through all such intersections would be tedious work as it requires solving a system of linear equations and then examining the feasibility of the result. Most of this work can be automated nowadays as there is various algebraic software available.

2 Definitions of the Problem and Previous Results

Here we describe the problem in more detail, to clarify which linear programs we focus on.

We have m machines with speeds $s_1 \geq s_2 \geq \cdots \geq s_m > 0$. We use a shorthand for the total speed of all machines: $S = s_1 + s_2 + \cdots + s_m$. We use special notation for the ratio $\alpha = \frac{S-s_1}{S} = \frac{s_2+s_3+\cdots+s_m}{s_1+s_2+s_3+\cdots+s_m}$ also, as it occurs in resulting formulas. The *input sequence* contains n jobs with processing times $p_1, p_2, \ldots, p_n$. Note that n is unknown to the online algorithm. Again, we use a shorthand $P = p_1 + p_2 + \cdots + p_n$ for the total processing time. The optimal makespan can be computed simply as a maximum of m numbers [15, 8, 3]:

$$C^*_{\max} = \max\left\{p_1^{\max}/s_1, \ldots, (p_1^{\max} + \cdots + p_{m-1}^{\max})/(s_1 + \cdots + s_{m-1}), P/S\right\}\,, \quad (1)$$

where $p_j^{\max}$ is j-th maximal job in the input sequence.

We view every semi-online problem as a *restriction* of set of possible input sequences of the online problem. Then we define a general semi-online input restriction to be simply a set Ψ of allowed input sequences. We call an input sequence a *partial input* if it is a prefix of some input sequence; the set of all partial inputs is denoted pref(Ψ). Thus the partial inputs are exactly the sequences that the algorithm can see at some point.

We restrict the definition of $C^*_{\max}$ only to Ψ-valid sequences of jobs as in [5], i.e., $C^{*,\Psi}_{\max}[\mathcal{J}] = C^*_{\max}[\mathcal{J}]$ for $\mathcal{J} \in \Psi$. Then we extend it to the partial inputs in a natural way, i.e., denoting the least achievable makespan over all valid continuations of such a partial input:

$$C^{*,\Psi}_{\max}[\mathcal{I}] = \inf\{C^*_{\max}[\mathcal{J}] \mid \mathcal{J} \in \Psi \;\&\; \mathcal{I} \text{ is a prefix of } \mathcal{J}\}\,. \quad (2)$$

This simplifies to the similar formula as in (1) for our restrictions. That is crucial, because then we are able to define $C^{*,\Psi}_{\max}$ as the minimum value satisfying several linear inequalities. (We consider the speeds as fixed parameters, because

they are given at the start of the algorithm, while the processing times are the variables as the algorithm does not know them in advance.)

We measure how bad are (semi-)online algorithms compared to optimal solution by the competitive ratio. We say that an algorithm is r-competitive if it manages to generate valid schedule with at most r times greater makespan than the makespan of the optimal (offline) schedule. For randomized algorithms we use expectation of the makespan over the random bits of the algorithm for each input sequence.

The exact analysis of scheduling on two machines was given in [6, 1, 16] for various semi-online problems, and in many more cases for non-preemptive scheduling. The paper [5] provides the framework to construct the algorithm with best possible competitive ratio for arbitrary number of machines and gives linear programs computing the optimal competitive ratio for several semi-online problems. We solve these linear programs for the cases of up to four machines. Below we list the restrictions studied in our paper.

Online scheduling. Here Ψ contains all sequences. In [2] is designed an optimal online algorithm for all speed vectors, but the competitive ratio is given implicitly by the linear program, which is solved there up to three machines. Here we analyze the competitive ratio of four machines.

Known sum of processing times, $\sum p_j = \bar{P}$. For a given value $\bar{P}$, Ψ contains all sequences with $P = \bar{P}$. The algorithm from [2] is extended in [5] to all semi-online problems studied in this paper. There is also noted that the overall ratio is surprisingly the same as in the general online case, but for $m = 2$, 1-approximation exists. We analyze the cases of $m = 3, 4$.

Known maximal job size, $p_{\max} = \bar{p}$. For a given value $\bar{p}$, Ψ contains all sequences with $\max p_j = \bar{p}$.

It is easy to see that this restriction is equivalent to the case when the first job is maximal, as any algorithm for that special case can be modified also for the case when the maximal job arrives later. (It just reserves space by virtually scheduling job of size $p_{\max}$, and uses this space as soon as first job of size $p_{\max}$ occurs.) Thus this restriction also includes non-increasing jobs. In [18] it is shown that for identical machines, the approximation ratio is the same as when the jobs are non-increasing. This is not the case for general speeds [5]. This restriction was introduced in [14] for non-preemptive scheduling on 2 identical machines. We analyze the cases of $m = 2, 3, 4$.

Known maximal job size and total processing time, $\sum p_j = \bar{P}$ & $p_{\max} = \bar{p}$. Here the algorithm knows both informations listed above. Scheduling on two machines is trivial as in $\sum p_j = \bar{P}$. We analyze the cases of $m = 3, 4$.

Inexact partial information. In this case, some of the previously considered values (optimal makespan, sum of job sizes, maximal job size) is not known

exactly but only up to a certain factor. These variants were studied first in [20] without preemption. The complete analysis of the preemptive version with approximately known optimum and maximal processing times on two machines and approximately known optimum on identical machines was given in [16]. We give a complete analysis for an approximately known optimum for three machines, denoted $T \leq C^*_{\max} \leq \beta T$.

Tightly grouped processing times, denoted $p \leq p_j \leq \beta p$. For given values $\bar{p}$ and β, Ψ contains all sequences with $p_j \in [\bar{p}, \beta\bar{p}]$ for each j. This restriction was introduced in [14] for non-preemptive scheduling on 2 identical machines. The complete analysis of the preemptive version on two machines was given independently in [1, 12]. The case of three identical machines was analyzed in [13]. We sketch a complete analysis for two machines, reproving the results of [1, 12] using our technique.

The paper [17] is probably the first paper which studied and compared several notions of semi-online algorithms, including the known sum of the processing times. Some combination of the previous restrictions were studied in [19] for non-preemptive scheduling on identical machines.

The lower bound, as well as matching algorithm can be found in [5]. We consider only nice restrictions, and for these is there proved that the best possible competitive ratio (even for randomized algorithms) can be computed as $r^\Psi(\mathbf{s}) = \sup_{\mathcal{J}} \bar{r}^\Psi(\mathbf{s}, \mathcal{J})$, where:

$$\bar{r}^\Psi(\mathbf{s}, \mathcal{J}) = \frac{\sum_{j=1}^{n} p_j}{\sum_{j=1}^{n} s_{n+1-j} \cdot C^{*,\Psi}_{\max}[\mathcal{J}_{[j]}]}\,, \tag{3}$$

where $s_{m+1} = s_{m+2} = \cdots = 0$, $\mathcal{J}$ is a prefix of Ψ-valid input sequence, $\mathcal{J}_{[j]}$ is a sequence containing first j jobs of the input sequence $\mathcal{J}$ and $n = |\mathcal{J}|$. The lower bound uses the argument of the total processing time available to (semi-)online algorithm processing n jobs: after the time $rC^{*,\Psi}_{\max}[J_{[j]}]$ only $n - j$ machines can be used as the algorithm is r-competitive semi-online, thus it has all jobs from $J_{[j]}$ finished, and there are only $n - j$ jobs in $J \setminus J_{[j]}$.

Following simplifications hold for the restrictions studied in this paper: It suffices to consider only $\mathcal{J}$ where jobs are sorted from the smallest to the largest. Also if $n > m$, it suffices to consider sequences with first $n - m$ jobs tiny where only their total processing time is interesting. Then it is easy to construct linear conditions exactly bounding $C^{*,\Psi}_{\max}[J_{[n-m+1]}], \ldots, C^{*,\Psi}_{\max}[J_{[n]}]$, and construct a linear program with the objective function $\bar{r}^\Psi(\mathbf{s}, \mathcal{J})$, where the job sizes and the optimal makespans are variables. These programs are already constructed in [5].

3 Online scheduling

The linear program in variables $q_1, \ldots, q_m$ and $O_1, \ldots, O_m$, follows, each inequality labeled with corresponding dual variable ($z_?$). The value of the optimal

solution is the competitive ratio of the problem for m machines. This program is already solved for $m \leq 3$ in [2].

$$\begin{array}{rcll} \textbf{maximize} & r = q_1 + \cdots + q_m & & \\ \textbf{subject to} & & & \\ 1 & = & s_1 O_m + s_2 O_{m-1} + \cdots + s_m O_1 & (z_{norm}) \\ q_1 + \cdots + q_k & \leq & (s_1 + \cdots + s_m) O_k & (z_k) \quad 1 \leq k \leq m \\ q_j + \cdots + q_k & \leq & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) \quad 2 \leq j \leq k \leq m \\ q_j & \leq & q_{j+1} & (z_{\leq,j}) \quad 2 \leq j \leq m-1 \\ 0 & \leq & q_1 & (z_{0,1}) \\ 0 & \leq & q_2 & (z_{0,2}) \end{array} \tag{4}$$

So we have the linear program and we solve it for all speed combinations of four machines ($m = 4$).

The list of the cases follows. We list not only the resulting competitive ratio and the validity domain of the case, (i.e., the conditions that define this domain), but also the sizes of the jobs in the input sequence proving the lower bound and also the dual coefficients proving that no better bound can be obtained from (3). Note that the algorithm from [2] matches exactly this lower bound, i.e., if it fails to achieve some competitive ratio, then it is due to the bound (3). (Recall the definition $\alpha = \frac{S-s_1}{S}$.)

Case I Ratio: $r = \dfrac{S}{s_1 + \alpha s_2 + \alpha^2 s_3 + \alpha^3 s_4}$

Conditions:

(A+: II) $s_2 \geq \alpha s_1$

(B+: IV) $s_2 + s_3 \geq (\alpha + \alpha^2) s_1$

Common denominator: $D = s_1 + \alpha s_2 + \alpha^2 s_3 + \alpha^3 s_4$

Nonbasic dual vars: $z_{0,1}, z_{0,2}, z_{\leq,2}, z_{\leq,3}, z_{2,3}, z_{2,4}, z_{3,4}$

Jobs:

$q_1 = \alpha^2(S - s_1)/D$

$q_2 = \alpha^2 s_1/D$

$q_3 = \alpha s_1/D$

$q_4 = s_1/D$

U.b. coefficients:

$z_{norm} = -r$

$z_1 = z_{2,2} = s_4/D$

$z_2 = (s_3 - (1-\alpha)s_4)/D$

$z_3 = (s_2 - (1-\alpha)(s_3 + \alpha s_4))/D$

$z_4 = (s_1 - (1-\alpha)(s_2 + \alpha s_3 + \alpha^2 s_4))/D$

$z_{3,3} = (s_3 + \alpha s_4)/D$

$z_{4,4} = (s_2 + \alpha s_3 + \alpha^2 s_4)/D$

We use a shorthand D for the common denominator of all formulas in our case description. *Conditions* state which values of parameters is this case optimal for. The label (A+: II) of a condition should be read as follows: The Case I is adjacent to the Case II, where the opposite condition (A-): $s_2 \leq \alpha s_1$ holds. *Jobs* give the main input sequence for lower bound. Note that the adversary may stop the sequence after any number of jobs. (In the general semi-online case the adversary may need to submit some more jobs, so that the input will be in Ψ, while maintaining $C^{*,\Psi}_{\max}$. But this is not needed in online scheduling.) The *nonbasic dual variables* are labels of inequalities, which we allow to be not tight, i.e., all other inequalities are turned into equations. The

upper bound coefficients are values of nonzero dual variables. These give the matching upper bound on the competitive ratio, which is obtained by summing up the corresponding inequalities multiplied by these coefficients. Note that all nonbasic dual variables have coefficients of zero value (and thus are not listed), because of the dual complementary slackness of linear programing.

Now we show how to check the correctness of Case I. The correctness of all other cases and all other restrictions in this paper can be checked in the same way. First we check the value of the objective function (using $s_1 = (1-\alpha)S$):

$$q_1 + q_2 + q_3 + q_4 \;=\; \alpha^2 S/D + \alpha(1-\alpha)S/D + (1-\alpha)S/D \;=\; S/D \;=\; r$$

Then we compute the values of the optima variables $O_1, \ldots, O_4$. We know that basic inequalities are satisfied by equality:

$$\begin{array}{lcrcrcrcl}
O_1 & \overset{(z_1)}{=} & q_1/S & = & \alpha^3/D & & & & \\
O_2 & \overset{(z_{2,2})}{=} & q_2/s_1 & = & \alpha^2/D & = & (q_1+q_2)/S & \overset{(z_2)}{=} & O_2 \\
O_3 & \overset{(z_{3,3})}{=} & q_3/s_1 & = & \alpha/D & = & (q_1+q_2+q_3)/S & \overset{(z_3)}{=} & O_3 \\
O_4 & \overset{(z_{4,4})}{=} & q_4/s_1 & = & 1/D & = & (q_1+q_2+q_3+q_4)/S & \overset{(z_4)}{=} & O_4\,.
\end{array}$$

The equal signs are labeled by labels of used equalities. Similarly the inequality signs are labeled by labels of sufficient conditions in following text.

We can also easily verify that the equation (z_{norm}) holds. We check the remaining (i.e., nonbasic) inequalities:

$$\begin{array}{lrcrcl}
(z_{2,3}) & q_2+q_3 & = & (\alpha+\alpha^2)s_1/D & & \\
 & & \leq^A & \alpha(s_1+s_2)/D & = & (s_1+s_2)O_3 \\
(z_{2,4}) & q_2+q_3+q_4 & = & (1+\alpha+\alpha^2)s_1/D & & \\
 & & \leq^B & (s_1+s_2+s_3)/D & = & (s_1+s_2+s_3)O_4 \\
(z_{3,4}) & q_3+q_4 & = & (1+\alpha)s_1/D & & \\
 & & \leq^A & (s_1+s_2)/D & = & (s_1+s_2)O_4\,.
\end{array}$$

We get $(z_{0,1}), (z_{0,2}), (z_{\leq,2})$ and $(z_{\leq,3})$ trivially from $\alpha \leq 1$ and $S \geq s_1$. Thus we know that our solution is feasible when A+ and B+ holds, so all algorithms are at least r-competitive for such sets of speeds. The sequence that proves this is for example: $p_1 = \cdots = p_4 = q_1/4, p_5 = q_2, p_6 = q_3, p_7 = q_4$.

Now we check the optimality of our solution. We check that all upper bound coefficients are nonnegative with the exception of z_{norm}:

$$\begin{array}{rcl}
z_2 D & = & (s_3 - s_4) + \alpha s_4 \;\geq\; 0 \\
z_3 D & = & (s_2 - s_3) + \alpha(s_3 - s_4) + \alpha^2 s_4 \;\geq\; 0 \\
z_4 D S & = & s_1 S - s_1(s_2 + \alpha s_3 + \alpha^2 s_4) \;\geq\; 0\,.
\end{array}$$

The coefficient z_{norm} is allowed to be negative, as (z_{norm}) is an equation. So we have all inequality coefficients nonnegative, thus we add up the inequalities

multiplied by their respective coefficients, and we use the resulting inequality:

$$\begin{aligned}
q_1+q_2+q_3+q_4 &= (q_1+q_2+q_3+q_4)(s_1+\alpha s_2+\alpha^2 s_3+\alpha^3 s_4)/D \\
&= q_1(z_1+z_2+z_3+z_4)+q_2(z_{2,2}+z_2+z_3+z_4)+q_3(z_{3,3}+z_3+z_4) \\
&\quad + q_4(z_{4,4}+z_4) \\
&\le O_1 S z_1 + O_2(S z_2 + s_1 z_{2,2}) + O_3(S z_3 + s_1 z_{3,3}) + O_4(S z_4 + s_1 z_{4,4}) \\
&= O_1 s_4 S/D + O_2 s_3 S/D + O_3 s_2 S/D + O_4 s_1 S/D \\
&= (O_1 s_4 + O_2 s_3 + O_3 s_2 + O_4 s_1) S/D \\
&\quad + (O_1 s_4 + O_2 s_3 + O_3 s_2 + O_4 s_1 - 1) z_{norm} \\
&= S/D\ .
\end{aligned}$$

This proves the optimality of our solution, i.e., there is no better solution and the algorithm from [2] is r-competitive.

Now we continue the list of the cases:

Case II Ratio: $r = \dfrac{S^2}{\sum_{i=1}^{4}\sum_{j=i}^{4} s_i s_j + \alpha(s_3+s_4)s_4 - s_4^2}$

Conditions:

(A-: I) $s_2 \le \alpha s_1$

(C+: III) $s_2 + s_3 \ge \alpha(s_1+s_2)$

Common denominator: $D = \sum_{i=1}^{4}\sum_{j=i}^{4} s_i s_j + \alpha(s_3+s_4)s_4 - s_4^2$

Nonbasic dual vars: $z_{0,1}, z_{0,2}, z_{\le,2}, z_{\le,3}, z_{2,2}, z_{2,4}, z_{3,3}$

Jobs:

$$\begin{aligned}
q_1 &= (s_3+s_4)(S-s_1)/D \\
q_2 &= (s_3+s_4)s_1/D \\
q_3 &= s_2 S/D \\
q_4 &= s_1 S/D
\end{aligned}$$

U.b. coefficients:

$$\begin{aligned}
z_{norm} &= -r \\
z_1 &= z_{2,3} = s_4 S/D \\
z_2 &= z_{3,4} = s_3 S/D \\
z_3 &= (s_2 S - (s_1+s_2)s_4)/D \\
z_4 &= (s_1(s_1+s_4) - s_2 s_3 \\
&\quad - (1-\alpha)(s_3+s_4)s_4)/D \\
z_{4,4} &= (s_2 S + (s_3+s_4)s_4)/D
\end{aligned}$$

Case III Ratio: $r = \dfrac{S^2}{\sum_{i=1}^{4}\sum_{j=i}^{4} s_i s_j}$

Conditions: (implicit: B-)

(A-: IV) $s_2 \le \alpha s_1$

(C-: II) $s_2 + s_3 \le \alpha(s_1+s_2)$

Common denominator: $D = \sum_{i=1}^{4}\sum_{j=i}^{4} s_i s_j$

Nonbasic dual vars: $z_{0,1}, z_{0,2}, z_{\le,2}, z_{\le,3}, z_{2,2}, z_{2,3}, z_{3,3}$

Jobs:

$$\begin{aligned}
q_1 &= s_4 S/D \\
q_2 &= s_3 S/D \\
q_3 &= s_2 S/D \\
q_4 &= s_1 S/D
\end{aligned}$$

U.b. coefficients:

$$\begin{aligned}
z_{norm} &= -r \\
z_1 &= z_{2,4} = s_4 S/D \\
z_2 &= z_{3,4} = s_3 S/D \\
z_3 &= z_{4,4} = s_2 S/D \\
z_4 &= (s_1 S - \sum_{i=2}^{4}\sum_{j=1}^{i-1} s_i s_j)/D
\end{aligned}$$

Case IV Ratio: $r = \dfrac{S}{s_1 + \alpha s_2 + \alpha^2 s_3 + s_4^2/S}$

Conditions: (implicit: C-)

(A+: III) $s_2 \geq \alpha s_1$

(B-: I) $s_2 + s_3 \leq (\alpha + \alpha^2) s_1$

Common denominator: $D = s_1 + \alpha s_2 + \alpha^2 s_3 + s_4^2/S$

Nonbasic dual vars:

$z_{0,1}, z_{0,2}, z_{\leq,2}, z_{\leq,3}, z_{2,2}, z_{2,3}, z_{3,4}$

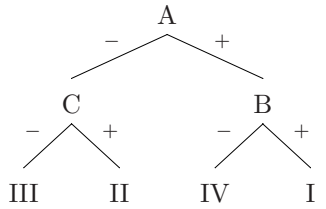
Jobs:

$$\begin{aligned} q_1 &= s_4/D \\ q_2 &= (\alpha(S - s_1) - s_4)/D \\ q_3 &= \alpha s_1/D \\ q_4 &= s_1/D \end{aligned}$$

U.b. coefficients:

$$\begin{aligned} z_{norm} &= -r \\ z_1 &= z_{2,4} = s_4/D \\ z_2 &= z_{3,3} = s_3/D \\ z_3 &= (s_2 - (1-\alpha)s_3)/D \\ z_4 &= (s_1 - s_4(S - s_4)/S \\ &\quad - (1-\alpha)(s_2 + s_3\alpha))/D \\ z_{4,4} &= (s_2 + \alpha s_3)/D \end{aligned}$$

We should also check that all listed cases cover whole space of the valid parameters (the speeds of the machines). This is easy, as condition A splits the space to the cases I+IV and the cases II+III. Then, I and IV are separated by B and fully cover the halfspace A+. Similarly II and III are separated by C and fully cover the halfspace A-. The diagram to the right of this paragraph represents the situations. It is a decision tree, where the nodes are labeled by the conditions and the leaves represent the cases.



4 Semi-online scheduling

4.1 Known sum of processing times, $\sum p_j = \bar{P}$

Here we are given a value $\bar{P}$ and Ψ contains all $\mathcal{J}$ with $P = \bar{P}$. Here we have to solve $n - 1$ linear programs for each $n < m$, and take the maximum of their solutions. The linear program for arbitrary n follows. Note that the shorthand S sums all speeds of the machines, i.e., including $s_{n+1}, \ldots, s_m$.

$$\begin{array}{rcll} \textbf{maximize} & r = q_1 + q_2 + \cdots + q_n & & \\ \textbf{subject to} & & & \\ 1 &= s_1 O_n + s_2 O_{n-1} + \cdots + s_n O_1 & (z_{norm}) & \\ q_1 + \cdots + q_n &\leq SO_k & (z_k) & 1 \leq k \leq n-1 \\ q_j + \cdots + q_k &\leq (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) & 1 \leq j \leq k \leq n \\ q_k &\leq q_{k+1} & (z_{\leq,k}) & 1 \leq k \leq n-1 \\ 0 &\leq q_1 & (z_0)\ . & \end{array} \tag{5}$$

Note that the condition $(z_{j,k})$ is also defined for $j = k$. Then it simplifies to:

$$q_j \quad \leq \quad s_1 O_j \quad (z_{j,j})$$

We omit the inequality (z_n) as it is implied by $(z_{1,n})$. (The implication follows trivially from $S = s_1 + \cdots + s_n + s_{n+1} + \cdots + s_m$.)

We now examine the special cases of $m = 2, 3$ and 4. The linear program is trivial for $n = 1$, and we conclude that for $m = 2$ the approximation ratio is equal to 1, i.e., there is an optimal algorithm.

We can see this also intuitively: The algorithm starts scheduling the incoming jobs in the interval $[0, T_1)$ where $T_1 \geq \bar{P}/S$. Consider the first time when a job is scheduled at the first real machine M_1. It is always possible to schedule this job at the empty machine M_1 so that it completes before the current optimal makespan. Furthermore, after M_1 is used the first time, the algorithm guarantees that in the interval $[0, T_1)$ there is only one real machine idle at any time. This in turn implies that the remaining jobs can be completed by time T_1, as the total size of all jobs is $\bar{P} \leq S \cdot T_1$.

Padding. In [5] we proved a theorem that shows that for any nice restriction Ψ, the restriction $\Psi, \sum p_j = \bar{P}$ has same overall competitive ratio. I.e., adding to any nice restriction the knowledge of the total size of jobs does not improve the overall approximation ratio. (It turns out, that all studied restrictions are nice.) This may sound surprising, as for two machines, knowing the sum allows to generate an optimal schedule, and also for three machines the improvement is significant.

4.1.1 $m = 3$

For $m = 3$, it remains to solve the linear program for $n = 2$. The ratio splits to two cases:

Case I Ratio: $r = \dfrac{(s_1 + s_2)S}{s_2(s_1 + s_2) + s_1 S}$

Conditions:

(A+: II) $s_1(s_1 + s_2) \quad \geq \quad s_2 S$

Common denominator: $D = s_2(s_1 + s_2) + s_1 S$

Nonbasic dual vars: $z_0, z_{\leq,1}, z_{1,1}$

Jobs:

$q_1 = s_2 S/D$

$q_2 = s_1 S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_1 = s_2(s_1 + s_2)/D$

$z_{1,2} = s_1 S/D$

Case II Ratio: $r = \dfrac{s_1(s_1+s_2)}{s_1^2+s_2^2}$

Conditions:

(A-: I) $s_1(s_1+s_2) \leq s_2 S$

Common denominator: $D = s_1^2 + s_2^2$

Nonbasic dual vars: $z_0, z_{\leq,1}, z_1$

Jobs:

$q_1 = s_1 s_2/D$

$q_2 = s_1^2/D$

U.b. coefficients:

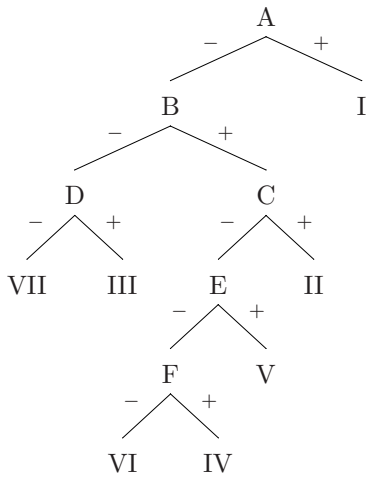
$z_{norm} = -r$

$z_{1,2} = s_1 S/D$

$z_{1,1} = z_{2,2} = s_2(s_1+s_2)/D$

4.1.2 $m = 4$

Here we solve the linear program for $n = 3$. Note, that the competitive ratio is the maximum of results of linear programs for all $n < m$. The diagram giving overall figure on how the conditions split the cases is to the right side of this paragraph.



Case I Ratio: $r = \dfrac{(s_1+s_2+s_3)S}{(s_2+s_3)(s_1+s_2+s_3)+s_1 S}$

Conditions:

(A+: II,III) $(s_1+s_2)(s_1+s_2+s_3) \geq (s_2+s_3)S$

Common denominator: $D = (s_2+s_3)(s_1+s_2+s_3)+s_1 S$

Nonbasic dual vars: $z_0, z_{\leq,2}, z_{1,1}, z_{1,2}, z_{2,2}, z_{2,3}$

Jobs:

$q_1 = q_2 = (s_2+s_3)S/(2D)$

$q_3 = s_1 S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_2 = s_2(s_1+s_2+s_3)/D$

$z_{1,3} = s_1 S/D$

$z_1 = s_3(s_1+s_2+s_3)/D$

Case II Ratio: $r = \dfrac{(s_1+s_2)(s_1+s_2+s_3)S}{s_1^2S+(s_1s_3+s_2(S+s_3))(s_1+s_2+s_3)}$

Conditions: (implicit: F+)

(A-: I) $(s_1+s_2)(s_1+s_2+s_3) \leq (s_2+s_3)S$

(B+: III) $s_1s_3 \geq s_2^2$

(C+: IV,V) $s_1(s1+s_2+s_3) \geq s_3S$

Common denominator: $D = s_1^2S+(s_1s_3+s_2(S+s_3))(s_1+s_2+s_3)$

Nonbasic dual vars: $z_0, z_{\leq,1}, z_{\leq,2}, z_2, z_{1,1}, z_{2,2}$

Jobs:

$q_1 = s_3(s_1+s_2)S/D$

$q_2 = s_2(s_1+s_2)S/D$

$q_3 = s_1(s_1+s_2)S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_1 = s_3(s_1+s_2)(s_1+s_2+s_3)/D$

$z_{1,2} = z_{3,3} = s_2(s_1+s_2+s_3)S/D$

$z_{1,3} = s_1^2S/D$

Case III Ratio: $r = \dfrac{(s_1+s_2)(s_1+s_2+s_3)S}{s_1^2S+(s_1s_3+s_2(S+s_3))(s_1+s_2+s_3)}$

Conditions:

(A-: I) $(s_1+s_2)(s_1+s_2+s_3) \leq (s_2+s_3)S$

(B-: II) $s_1s_3 \leq s_2^2$

(D+: VII) $s_1(s_1+s_2)(s1+s_2+s_3) \geq s_2(s_2+s_3)S$

Common denominator: $D = s_1^2S+(s_1s_3+s_2(S+s_3))(s_1+s_2+s_3)$

Nonbasic dual vars: $z_0, z_{\leq,1}, z_{\leq,2}, z_2, z_{1,1}, z_{2,3}$

Jobs:

$q_1 = s_2(s_2+s_3)S/D$

$q_2 = s_1(s_2+s_3)S/D$

$q_3 = s_1(s_1+s_2)S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_1 = s_3(s_1+s_2)(s_1+s_2+s_3)/D$

$z_{1,2} = z_{3,3} = s_2(s_1+s_2+s_3)S/D$

Case IV Ratio: $r = \dfrac{(s_1+s_2)^2S}{s_1^2(S-s_1)+(s_1+s_2)(s_1s_3+s_2(S+s_3))}$

Conditions: (implicit: A-,B+)

(C-: II)) $s_1(s_1+s_2+s_3) \leq s_3S$

(E-: V)) $s_1^3 \leq s_3(s_1+s_2)^2$

(F+: VI)) $s_1(s_1^2+s_1s_2+s_2^2) \geq s_2^2S$

Common denominator: $D = s_1^2(S-s_1)+(s_1+s_2)(s_1s_3+s_2(S+s_3))$

Nonbasic dual vars: $z_0, z_{\leq,1}, z_{\leq,2}, z_2, z_{1,3}, z_{2,2}$

Jobs:

$q_1 = s_1(s_1+s_2)^2/D$

$q_2 = s_2(s_1+s_2)(S-s_1)/D$

$q_3 = s_1(s_1+s_2)(S-s_1)/D$

U.b. coefficients:

$z_{norm} = -r$

$z_1 = (s_3(s_1+s_2)^2-s_1^3)/D$

$z_{1,1} = z_{2,3} = s_1^2S/D$

$z_{1,2} = z_{3,3} = s_2(s_1+s_2)S/D$

Case V Ratio: $r = \dfrac{s_1(s_1+s_2)(s_1+s_2+s_3)}{s_2s_3(s_1-s_3)+s_1(2s_1+s_2)(s_2+s_3)}$

Conditions: (implicit: A-)

(B+: VII) $s_1s_3 \geq s_2^2$

(C-: II) $s_1(s_1+s_2+s_3) \leq s_3S$

(E+: IV,VI) $s_1^3 \geq s_3(s_1+s_2)^2$

Common denominator: $D = s_2s_3(s_1-s_3)+s_1(2s_1+s_2)(s_2+s_3)$

Nonbasic dual vars: $z_0, z_{\leq,1}, z_{\leq,2}, z_1, z_2, z_{2,2}$

Jobs:

$q_1 = s_3s_1(s_1+s_2)/D$

$q_2 = s_2s_1(s_1+s_2)/D$

$q_3 = s_1^2(s_1+s_2)/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{1,1} = z_{2,3} = s_3(s_1+s_2)(s_1+s_2+s_3)/D$

$z_{1,2} = z_{3,3} = s_1s_2(s_1+s_2+s_3)/D$

$z_{1,3} = (s_1^3-(s_1+s_2)^2s_3)/D$

Case VI Ratio: $r = \dfrac{s_1(s_1^2+s_1s_2+s_2^2)}{s_1^3+s_2^2(s_1+s_3)}$

Conditions: (implicit: A-,C-)

(B+: VII) $s_1s_3 \geq s_2^2$

(E-: V) $s_1^3 \leq s_3(s_1+s_2)^2$

(F-: VI) $s_1(s_1^2+s_1s_2+s_2^2) \leq s_2^2S$

Common denominator: $D = s_1^3+s_2^2(s_1+s_3)$

Nonbasic dual vars: $z_0, z_{\leq,1}, z_{\leq,2}, z_1, z_2, z_{1,3}$

Jobs:

$q_1 = s_1s_2^2/D$

$q_2 = s_1^2s_2/D$

$q_3 = s_1^3/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{1,1} = s_3(s_1^2+s_1s_2+s_2^2)/D$

$z_{1,2} = s_1(s_1^2+s_2^2-s_3(s_1+s_2))/D$

$z_{2,2} = (s_3(s_1+s_2)^2-s_1^3)/D$

$z_{2,3} = s_1(s_1^2-s_2s_3)/D$

$z_{3,3} = s_2(s_1s_2+s_1s_3+s_2s_3)/D$

Case VII Ratio: $r = \dfrac{s_1(s_1+s_2)(s_1+s_2+s_3)}{s_1^2(s_1+s_2)+s_2(s_1+s_3)(s_2+s_3)}$

Conditions: (implicit: A-)

(B-: V,VI) $s_1s_3 \leq s_2^2$

(D-: III) $s_1(s_1+s_2)(s1+s_2+s_3) \leq s_2(s_2+s_3)S$

Common denominator: $D = s_1^2(s_1+s_2)+s_2(s_1+s_3)(s_2+s_3)$

Nonbasic dual vars: $z_0, z_{\leq,1}, z_{\leq,2}, z_1, z_2, z_{2,3}$

Jobs:

$q_1 = s_1s_2(s_2+s_3)/D$

$q_2 = s_1^2(s_2+s_3)/D$

$q_3 = s_1^2(s_1+s_2)/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{1,1} = z_{2,2} = s_3(s_1+s_2)(s_1+s_2+s_3)/D$

$z_{1,2} = s_1(s_2-s_3)(s_1+s_2+s_3)/D$

$z_{1,3} = s_1(s_1^2-s_2s_3)/D$

$z_{3,3} = s_1(s_1+s_3)(s_1+s_2+s_3)/D$

4.2 Known maximal processing time, $p_{\max} = \bar{p}$

Here we are given $\bar{p}$, the maximal size of a job. We need to compute one linear program for $n \geq m$ and $n-2$ linear programs, one for each $n \in \{2, \ldots, n-1\}$. We should take the maximum of the results and 1 (that is the result for $n = 1$) to obtain the competitive ratio for given m. The program for $n \geq m$ follows:

$$
\begin{array}{rcll}
\textbf{maximize} & r = p + q_1 + \cdots + q_m & & \\
\textbf{subject to} & & & \\
1 & = & s_1 O_m + \cdots + s_m O_1 & (z_{norm}) \\
p + q_1 + \cdots + q_k & & & \\
& \leq & S O_k & (z_k) \qquad 1 \leq k \leq m \\
p & \leq & s_1 O_k & (z_{k,k}) \qquad 1 \leq k \leq m-1 \\
p + q_{j+1} + \cdots + q_k & & & \\
& \leq & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) \qquad 1 \leq j < k \leq m \\
0 & \leq & q_1 & (z_{0,1}) \\
0 & \leq & q_2 & (z_{0,2}) \\
q_k & \leq & q_{k+1} & (z_{\leq,k}) \qquad 2 \leq k \leq m-1 \\
q_m & = & p & \text{(substitution)}
\end{array}
\tag{6}
$$

We omit $(z_{1,m})$ as it is implied by (z_m) as well as $(z_{m,m})$ implied by $(z_{m-1,m})$.

The linear program for the sequences of length $2 \leq n < m$ is similar:

$$
\begin{array}{rcll}
\textbf{maximize} & r = p + q_2 + \cdots + q_n & & \\
\textbf{subject to} & & & \\
1 & = & s_1 O_n + \cdots + s_n O_1 & (z_{norm}) \\
p & \leq & s_1 O_k & (z_{k,k}) \qquad 1 \leq k \leq n-1 \\
p + q_{j+1} + \cdots + q_k & & & \\
& \leq & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) \qquad 1 \leq j < k \leq n \\
0 & \leq & q_2 & (z_{0,2}) \\
q_k & \leq & q_{k+1} & (z_{\leq,k}) \qquad 2 \leq k \leq n-1 \\
q_n & = & p & \text{(substitution)}
\end{array}
\tag{7}
$$

We omit $(z_{n,n})$ here also.

4.2.1 $m = 2$

For two machines we need to solve only (6) for $m = 2$. The result is covered by a single case, valid for all combinations of the speeds. But this case is also valid for greater values of m, although only if the condition A+ is satisfied. (The condition A+ is trivially satisfied for $m = 2$.) We include the general description of the case here:

Case I Ratio: $r = \dfrac{S^2 + s_1 S}{S^2 + s_1^2}$

Conditions:

(A+) $s_1 s_2 \geq (S - s_1 - s_2)S$

Common denominator: $D = S^2 + s_1^2$

Nonbasic dual vars:
all but: $z_1, \ldots, z_m, z_{norm}$,
$z_{i,i}$ for $i = 1, \ldots, m-1$

Jobs:
$p = s_1^2 S/D$
$q_1 = s_1(S - s_1)S/D$
$q_2 = \cdots = q_{m-1} = 0$

U.b. coefficients:
$z_{norm} = -r$
$i = 1, \ldots, m-2$:
$z_{i,i} = s_{m-i+1} S(S + s_1)/D$
$z_{m-1,m-1} = (s_1 s_2 - (S - s_1 - s_2)S)S/D$
$z_{m-1} = s_1(S - s_1)S/D$
$z_m = s_1^2(S + s_1)/D$

4.2.2 $n = 2$

We need to solve also (7) for $n = 2$ to solve cases when $m > 2$. This becomes trivial as there are only two jobs and both have size p in the wost case sequence. Thus the competitive ratio is $\frac{2s_1(s_1+s_2)}{s_1(s_1+s_2)+s_1^2+s_2^2}$ here.

4.2.3 $m = 3$

We have already solved $n = 2$, thus it suffices to solve (6) for $m = 3$. Here we have two cases, one already shown in the solution of $m = 2$.

Case II Ratio: $r = \dfrac{S^2 + 2s_1 S}{S^2 + 2s_1^2 + s_1 s_2}$

Conditions:

(A-:I) $s_1 s_2 \leq s_3 S$

Common denominator: $D = S^2 + 2s_1^2 + s_1 s_2$

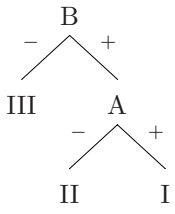
Nonbasic dual vars:
$z_{0,1}, z_{0,2}, z_{1,2}, z_{2,2}, z_{2,3}$

Jobs:
$p = q_2 = s_1 S/D$
$q_1 = (S - s_1)S/D$

U.b. coefficients:
$z_{norm} = -r$
$z_{\leq,2} = z_1 = (s_3 S - s_1 s_2)/D$
$z_2 = s_2(2s_1 + S)/D$
$z_3 = s_1(2s_1 + S)/D$
$z_{1,1} = (s_2 + 2s_3)S/D$

4.2.4 $n = 3$

Solutions of (7) for $n = 3$. These are needed to evaluate the cases of $m > 3$ machines. The overview on how the cases split is to the right again.



Case I Ratio: $r = \dfrac{2s_1(s_1+s_2)S}{s_1(s_1+s_2)(S+s_1)+s_2s_3(3s_1+s_2)}$

Conditions:

(A+:II) $2s_1s_3 \geq s_2(s_1+s_2)$

(B+:III) $s_1s_2 \geq s_3(s_1+s_2)$

Common denominator: $D = s_1(s_1+s_2)(S+s_1)+s_2s_3(3s_1+s_2)$

Nonbasic dual vars: $z_{0,2}, z_{\leq,2}, z_{2,2}$

Jobs:

$p = s_1(s_1+s_2)^2/D$

$q_2 = 2s_1s_3(s_1+s_2)/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{1,1} = 2s_3(s_1+s_2)S/D$

$z_{1,2} = 2s_1s_2(s_1+s_2)S/D$

$z_{1,3} = (2s_1^2+(s_1+s_2)s_3-s_1s_2)\cdot(s_1+s_2)/D$

$z_{2,3} = (s_1s_2-(s_1+s_2)s_3)S/D$

Case II Ratio: $r = \dfrac{(2s_1+s_2)S}{S^2+s_1(s_1-s_3)}$

Conditions:

(A-:I) $2s_1s_3 \leq s_2(s_1+s_2)$

(B+:III) $s_1s_2 \geq s_3(s_1+s_2)$

Common denominator: $D = S^2+s_1(s_1-s_3)$

Nonbasic dual vars: $z_{0,2}, z_{\leq,2}, z_{2,3}$

Jobs:

$p = s_1S/D$

$q_2 = s_2S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{1,1} = s_3(2s_1+s_2)S/(s_1D)$

$z_{1,2} = (s_2+s_3)S/D$

$z_{1,3} = s_1(2s_1+s_2)/D$

$z_{2,2} = (s_1s_2-(s_1+s_2)s_3)/(s_1D)$

Case III Ratio: $r = \dfrac{3s_1(s_1+s_2)S}{(s_1(2S+s_1-s_3)+s_2s_3)S-3s_1^2s_3}$

Conditions:

(B-:I,II) $s_1s_2 \leq s_3(s_1+s_2)$

Common denominator: $D = (s_1(2S+s_1-s_3)+s_2s_3)S-3s_1^2s_3$

Nonbasic dual vars: $z_{0,2}, z_{2,2}, z_{2,3}$

Jobs:

$p = q_2 = s_1(s_1+s_2)S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{\leq,2} = S(s_3(s_1+s_2)-s_1s_2)/D$

$z_{1,1} = 3s_3(s_1+s_2)S/D$

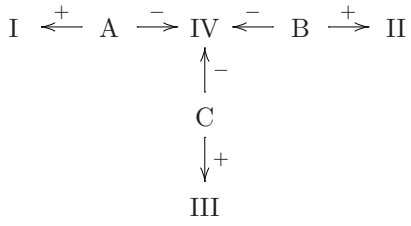
$z_{1,2} = 3s_1s_2S/D$

$z_{1,3} = 3s_1^2(s_1+s_2)/D$

4.2.5 $m = 4$

Solutions of (6) for $m = 4$, the Case I is shown in $m = 2$. Note that we need to compare this solutions to the solutions of (7) for $n = 1, 2, 3$.

The case diagram may look strange, but this is due to the fact, that the conditions A+,B+ and C+ are pairwise exclusive.



Case II Ratio: $r = \dfrac{S^2 + 3s_1S}{S^2 + s_1(3s_1 + 2s_2 + s_3)}$

Conditions:

(B+: IV) $s_4S \geq s_1(s_2 + 2s_3)$

Common denominator: $D = S^2 + s_1(3s_1 + 2s_2 + s_3)$

Nonbasic dual vars:

$z_{\leq,1}, z_{\leq,2}, z_{1,2}, z_{1,3}, z_{2,2}, z_{2,3}, z_{2,4}, z_{3,3}, z_{3,4}, z_{4,4}$

Jobs:

$q_1 = s_1(S - s_1)S/D$

$q_2 = q_3 = p = s_1^2S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{\leq,3} = s_4S - s_1(s_2 + 2s_3)/D$

$z_{\leq,4} = s_1((s_3 + 2s_4)S - s_1(2s_2 + s_3))/D$

$z_1 = s_1(s_2 + 2s_3 + 3s_4)S/D$

$z_2 = s_1s_3(S + 3s_1)/D$

$z_3 = s_1s_2(S + 3s_1)/D$

$z_4 = s_1^2(S + 3s_1)/D$

$z_{1,1} = s_1(s_2 + 2s_3 + 3s_4)S/D$

Case III Ratio: $r = \dfrac{s_1S(2S + s_1 + s_2)}{2s_1S^2 + s_1(s_1 - s_3)(s_1 + s_2) - (s_1 - s_2)s_4S}$

Conditions:

(C+: IV) $s_1(s_3 + s_4) \geq s_2S$

Common denominator: $D = 2s_1S^2 + s_1(s_1 - s_3)(s_1 + s_2) - (s_1 - s_2)s_4S$

Nonbasic dual vars:

$z_{\leq,1}, z_{\leq,2}, z_{\leq,3}, z_{1,3}, z_{2,2}, z_{2,4}, z_{3,3}, z_{3,4}, z_{4,4}; z_1$

Jobs:

$q_1 = s_1(s_1 + s_2)(S - s_1)S/D$

$q_2 = s_1(s_1(s_3 + s_4) - s_2S)S/D$

$q_3 = p = s_1^2(s_1 + s_2)S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{\leq,4} = s_1(s_4S^2 + s_3s_1(s_1 + s_2) - s_2s_1S)/D$

$z_2 = s_3s_1^2(2S + s_1 + s_2)/D$

$z_3 = s_1(s_4(s_1 + s_2)S + 2s_1(s_2S - s_3(s_1 + s_2)))/D$

$z_4 = s_1^3(2S + s_1 + s_2)/D$

$z_{1,1} = s_4s_1S(2S + s_1 + s_2)/D$

$z_{2,3} = s_1S(s_1(s_2 + 2s_3) - s_4S)/D$

Case IV Ratio: $r = \dfrac{S(S+2s_1)}{S^2+s_1(2s_1+s_2)}$

Conditions:

(A-: I) $s_1 s_2 \leq (s_3+s_4)S$

(B-: II) $s_4 S \leq s_1(s_2+2s_3)$

(C-: III) $s_1(s_3+s_4) \leq s_2 S$

Common denominator: $D = S^2+s_1(2s_1+s_2)$

Nonbasic dual vars:

$z_{\leq,1}, z_{\leq,2}, z_{\leq,3}, z_{1,3}, z_{2,3},$
$z_{2,4}, z_{3,3}, z_{3,4}, z_{4,4}; z_1$

Jobs:

$q_1 = s_1(S-s_1)S/D$

$q_2 = 0$

$q_3 = p = s_1^2 S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{\leq,4} = z_2 = s_1((s_3+s_4)S - s_1 s_2)/D$

$z_3 = s_2 s_1(S+2s_1)/D$

$z_4 = s_1^2(S+2s_1)/D$

$z_{1,1} = s_4 S(S+2s_1)/D$

$z_{2,2} = S(s_1(s_2+2s_3) - s_4 S)/D$

In the cases I and IV $q_2 = 0$, but we know, that jobs are ordered. Thus we need to consider inputs $p_1 = p$, $p_2 = \cdots = p_{n-2} = \epsilon$, $p_{n-1} = \max\{\epsilon, q_3\}$, $p_n = p$, where $n = q_1/\epsilon$. We approach the bound given by linear program with $\epsilon \to 0$.

4.3 Known max. proc. time and sum of proc. times, $\sum p_j = \bar{P}$ & $p_{\max} = \bar{p}$

We are given both the maximal processing time and the sum of the processing times. We introduce a constant $\beta = \frac{\bar{P}-\bar{p}}{\bar{p}}$ to be able to use homogeneity. We need to solve $n-1$ linear programs as for $\sum p_j = \bar{P}$. The linear program for every $n < m$ follows:

$$
\begin{array}{rcll}
\textbf{maximize} & r = p + q_2 + \cdots + q_n & & \\
\textbf{subject to} & & & \\
1 & = & s_1 O_n + \cdots + s_1 O_n \quad (z_{norm}) & \\
(1+\beta)p & \leq & S O_k \quad (z_k) & 1 \leq k \leq n \\
p & \leq & s_1 O_k \quad (z_{k,k}) & 1 \leq k \leq n \\
p + q_{j+1} + \cdots + q_k & & & \\
 & \leq & (s_1 + \cdots + s_{k-j+1}) O_k \quad (z_{j,k}) & 1 \leq j < k \leq n \\
q_2 + \cdots + q_n & \leq & \beta p \quad (z_{\leq,0}) & \\
q_n & \leq & p \quad (z_{\leq,1}) & \\
0 & \leq & q_2 \quad (z_{\leq,2}) & \\
q_{k-1} & \leq & q_k \quad (z_{\leq,k}) & 3 \leq k \leq n
\end{array}
\tag{8}
$$

If we want to solve the problem for some number of machines m, we need to solve all $n-1$ linear programs for $n < m$ and take the maximum of their solutions again. For $n = 1$ the program is trivial and the resulting competitive ratio is 1.

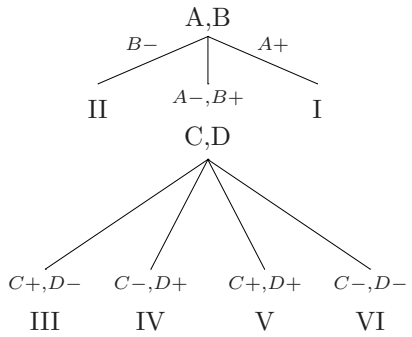
Now we list two cases, where the competitive ratio is 1, as the value of the optimal makespan can be derived from the knowledge of $\bar{p}, \bar{P}$ and the speeds of the machines. The linear programs for $n \geq 2$ have the results typically smaller than (or equal to) 1 here, thus the maximum is obtained from the program for $n = 1$.

Case I (A+) $\beta s_1 \leq s_2$: The second machine is fast enough to process all but the maximal job before the maximal job finishes on the first machine.

Case II (B-) $(1+\beta)(s_1 + \cdots + s_n) \geq nS$: Now, all optima are given by the total processing time, and the algorithm from [3] produces optimal schedule.

4.3.1 $n = 2$

We list the cases of the solution for (8). The conditions A+ and B- are mutually exclusive (e.g., $\beta \geq 1$ separates them).



Case III

Ratio: $r = \dfrac{2s_1(s_1+s_2)}{s_1(s_1+s_2)+s_1^2+s_2^2}$

Conditions: (Implicit B+, A-)

(C+:IV) $(1+\beta)s_1 \leq S$

(D-:V) $\beta \geq 1$

Common denominator: $D = s_1(s_1+s_2)+s_1^2+s_2^2$

Nonbasic dual vars: $z_{\leq,0}, z_1, z_2, z_{2,2}$

Jobs: $p = q_2 = s_1(s_1+s_2)/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{\leq,1} = s_2(s_1+s_2)/D$

$z_{1,1} = 2s_2(s_1+s_2)/D$

$z_{1,2} = 2s_1^2/D$

Case IV

Ratio: $r = \dfrac{2(s_1+s_2)S}{2s_1S+(1+\beta)s_2(s_1+s_2)}$

Conditions: (Implicit A-)

(C-:III) $(1+\beta)s_1 \geq S$

(D-:VI) $\beta \geq 1$

(B+:II) $(1+\beta)(s_1+s_2) \leq 2S$

Common denominator: $D = 2s_1S+(1+\beta)s_2(s_1+s_2)$

Nonbasic dual vars: $z_{\leq,0}, z_2, z_{1,1}, z_{2,2}$

Jobs: $p = q_2 = (s_1+s_2)S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{\leq,1} = (1+\beta)s_2(s_1+s_2)/D$

$z_1 = 2s_2(s_1+s_2)/D$

$z_{1,2} = 2s_1S/D$

Case V Ratio: $r = \dfrac{(\beta+1)s_1(s_1+s_2)}{s_1(s_1+s_2)+\beta s_1^2+s_2^2}$

Conditions: (Implicit B+)

(A-:I) $\beta s_1 \geq s_2$

(C+:VI) $(1+\beta)s_1 \leq S$

(D+:III) $\beta \leq 1$

Common denominator: $D = s_1(s_1+s_2)+\beta s_1^2+s_2^2$

Nonbasic dual vars: $z_{\leq,1}, z_1, z_2, z_{2,2}$

Jobs:

$p = s_1(s_1+s_2)/D$

$q_2 = \beta s_1(s_1+s_2)/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{\leq,2} = s_2(s_1+s_2)/D$

$z_{1,1} = (1+\beta)s_2(s_1+s_2)/D$

$z_{1,2} = (1+\beta)s_1^2/D$

Case VI Ratio: $r = \dfrac{(s_1+s_2)S}{s_1S+(s_1+s_2)s_2}$

Conditions: (Implicit A-,B+)

(C-:V) $(1+\beta)s_1 \geq S$

(D+:IV) $\beta \leq 1$

Common denominator: $D = s_1S+(s_1+s_2)s_2$

Nonbasic dual vars: $z_{\leq,1}, z_2, z_{1,1}, z_{2,2}$

Jobs:

$p = (s_1+s_2)S/((1+\beta)D)$

$q_2 = \beta(s_1+s_2)S/((1+\beta)D)$

U.b. coefficients:

$z_{norm} = -r$

$z_{\leq,2} = z_1 = s_2(s_1+s_2)/D$

$z_{1,2} = s_1S/D$

4.3.2 $n = 3$

The case list is omitted here, as there are 17 cases with 15 different formulas of value of optimal solution. Three formulas (in three cases) have value smaller or equal to 1 (these cases the Cases I and II), thus the maximal lower bound is obtained for $n = 2$ or $n = 1$ in these cases, when considering four machines. The remaining formulas have value of at least 1.

4.4 Approximately known optimum, $T \leq C^*_{\max} \leq \beta T$

We are given lower and upper bound on the optimal makespan in advance here. We give the upper bound as β times the lower bound, so we can use homogeneity. Then the ratio does not depend on actual value of the lower bound T, and it also does not occur in our linear programs. To obtain proper solution for m machines, we need to solve one linear program for the case of $n \geq m$ and $m-2$ linear programs each for one $n < m$ (the linear program for $n = 1$ is trivial).

Then we take the maximum of the solutions as usual.

$$\begin{array}{rcll}
\textbf{maximize} & r = q_1 + \cdots + q_m & & \\
\textbf{subject to} & & & \\
1 & = & s_1 O_m + s_2 O_{m-1} + \cdots + s_m O_1 & (z_{norm}) \\
q_1 + \cdots + q_k & \leq & (s_1 + \cdots + s_m) O_k & (z_k) \quad 1 \leq k \leq m \\
q_j + \cdots + q_k & \leq & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) \quad 2 \leq j \leq k \leq m \\
q_j & \leq & q_{j+1} & (z_{\leq,j}) \quad 2 \leq j \leq m-1 \\
0 & \leq & q_1 & (z_{0,1}) \\
0 & \leq & q_2 & (z_{0,2}) \\
O_k & \leq & O_{k+1} & (z_{\beta,k}) \quad 1 \leq k \leq m-1 \\
O_m & \leq & \beta O_1 & (z_\beta)
\end{array} \tag{9}$$

And for $n < m$:

$$\begin{array}{rcll}
\textbf{maximize} & r = q_1 + \cdots + q_n & & \\
\textbf{subject to} & & & \\
1 & = & s_1 O_n + s_2 O_{n-1} + \cdots + s_n O_1 & (z_{norm}) \\
q_j + \cdots + q_k & \leq & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) \quad 1 \leq j \leq k \leq n \\
q_j & \leq & q_{j+1} & (z_{\leq,j}) \quad 1 \leq j \leq n-1 \\
0 & \leq & q_1 & (z_0) \\
O_k & \leq & O_{k+1} & (z_{\beta,k}) \quad 1 \leq k \leq n-1 \\
O_n & \leq & \beta O_1 & (z_\beta)
\end{array} \tag{10}$$

The cases for this problem include also the cases of the generic online problem, with additional conditions on β. These conditions say when the β is too large to provide any useful information for the algorithm.

4.4.1 $m = 2$

For two machines, we need to solve only (9) for $m = 2$ as the case of $n = 1$ is trivial. Both cases here give ratio greater than 1, thus giving the final solution for two machines. The first case is identical to the only case of online scheduling. The condition A+ says β is large enough.

Case I Ratio: $r = \dfrac{(s_1 + s_2)^2}{s_1^2 + s_1 s_2 + s_2^2}$

Conditions:

(A+:II) $\beta s_2 \geq s_1 + s_2$

Common denominator: $D = s_1^2 + s_1 s_2 + s_2^2$

Nonbasic dual vars: $z_{0,1}, z_{0,2}, z_{\beta,1}, z_\beta$

Jobs:

$q_1 = s_1(s_1 + s_2)/D$

$q_2 = s_2(s_1 + s_2)/D$

U.b. coefficients:

$z_{norm} = -r$

$z_1 = z_{2,2} = s_2(s_1 + s_2)/D$

$z_2 = s_1^2/D$

Case II Ratio: $r = \dfrac{\beta(s_1+s_2)}{\beta s_1 + s_2}$

Conditions:

(A-:I) $\beta s_2 \leq s_1 + s_2$

Common denominator: $D = \beta s_1 + s_2$

Nonbasic dual vars: $z_{0,1}, z_{0,2}, z_{\beta,1}, z_1$

Jobs:

$q_1 = \beta s_2 / D$

$q_2 = \beta s_1 / D$

U.b. coefficients:

$z_{norm} = -r$

$z_\beta = s_2(s_1+s_2)/D$

$z_2 = \beta(s_1+s_2)/D$

4.4.2 $n = 2$

Here we solve (10) as this is needed in solutions of cases with more than two machines.

Case I Ratio: $r = \dfrac{s_1(s_1+s_2)}{s_1^2+s_2^2}$

Conditions:

(A+:II) $\beta s_2 \geq s_1$

Common denominator: $D = s_1^2 + s_2^2$

Nonbasic dual vars: $z_0, z_{\leq,1}, z_\beta, z_{\beta,1}$

Jobs:

$q_1 = s_1 s_2 / D$

$q_2 = s_1^2 / D$

U.b. coefficients:

$z_{norm} = -r$

$z_{1,1} = z_{2,2} = s_2(s_1+s_2)/D$

$z_{1,2} = s_1(s_1+s_2)/D$

Case II Ratio: $r = \dfrac{\beta(s_1+s_2)}{\beta s_1 + s_2}$

Conditions:

(A-:I) $\beta s_2 \leq s_1$

Common denominator: $D = \beta s_1 + s_2$

Nonbasic dual vars: $z_0, z_{\leq,1}, z_{\beta,1}, z_{1,1}$

Jobs:

$q_1 = s_1 / D$

$q_2 = (\beta(s_1+s_2) - s_1)/D$

U.b. coefficients:

$z_{norm} = -r$

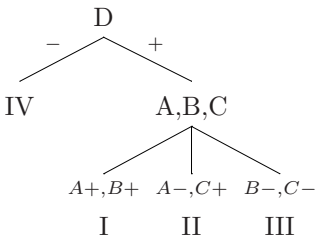
$z_\beta = s_2(s_1+s_2)/D$

$z_{1,2} = \beta(s_1+s_2)/D$

4.4.3 $m = 3$

In this case we need to take maximum of (9) for $m = 3$ and (10) for $n = 1, 2$. It turns out that the solution of linear program for $m = 3$ is always maximal of these three. First two cases are identical to the case of online scheduling. If B+ or C+ is satisfied then β is large enough and the information about the optimum is useless in general.

The conditions A-,B+ and C- can not hold simultaneously, as well as the conditions A+, B- and C+. Thus the cases split as shown on the diagram.



Case I Ratio: $r = \dfrac{S}{s_1 + \alpha s_2 + \alpha^2 s_3}$

Conditions: (Implicit D+)

(A+:II) $s_2 \geq \alpha s_1$

(B+:III) $\beta\alpha^2 \geq 1$

Common denominator: $D = s_1 + \alpha s_2 + \alpha^2 s_3$

Nonbasic dual vars: $z_0, z_{\leq,2}, z_\beta, z_{\beta,1}, z_{\beta,2}, z_{2,3}$

Jobs:

$q_1 = \alpha^2 S/D$

$q_2 = \alpha s_1/D$

$q_3 = s_1/D$

U.b. coefficients:

$z_{norm} = -r$

$z_1 = z_{2,2} = s_3/D$

$z_2 = (\alpha s_1 + (2-\alpha)s_2)/D$

$z_3 = (1-\alpha)^2(S + s_3)/D$

$z_{3,3} = (s_2 + \alpha s_3)/D$

Case II Ratio: $r = \dfrac{S^2}{s_1^2 + s_2^2 + s_3^2 + s_1 s_2 + s_1 s_3 + s_2 s_3}$

Conditions: (Implicit D+)

(A-:II) $s_2 \leq \alpha s_1$

(C+:III) $\beta s_3 \geq S$

Common denominator: $D = s_1^2 + s_2^2 + s_3^2 + s_1 s_2 + s_1 s_3 + s_2 s_3$

Nonbasic dual vars: $z_0, z_{\leq,2}, z_\beta, z_{\beta,1}, z_{\beta,2}, z_{2,2}$

Jobs:

$q_1 = s_3 S/D$

$q_2 = s_2 S/D$

$q_3 = s_1 S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_1 = z_{2,3} = s_3 S/D$

$z_2 = z_{3,3} = s_2 S/D$

$z_3 = (s_1^2 - s_2 s_3)/D$

Case III Ratio: $r = \dfrac{\beta S}{\beta s_1 + \beta\alpha s_2 + s_3}$

Conditions:

(B-:I) $\beta\alpha^2 \leq 1$

(C-:II) $\beta s_3 \leq S$

(D+:IV) $\beta\alpha \geq 1$

Common denominator: $D = \beta s_1 + \beta\alpha s_2 + s_3$

Nonbasic dual vars: $z_0, z_{\leq,2}, z_{\beta,1}, z_{\beta,2}, z_1, z_{2,3}$

Jobs:

$q_1 = S/D$

$q_2 = (\beta\alpha - 1)S/D$

$q_3 = \beta s_1/D$

U.b. coefficients:

$z_{norm} = -r$

$z_\beta = s_3 S^2/D$

$z_2 = z_{3,3} = \beta s_2 S/D$

$z_3 = (s_3 S + \beta s_1(s_1 + s_3))/D$

Case IV Ratio: $r = \dfrac{\beta S}{(\beta - 1)s_1 + S}$

Conditions: (Implicit B-,C-)

(D-:IV) $\beta\alpha \leq 1$

Common denominator: $D = (\beta - 1)s_1 + S$

Nonbasic dual vars: $z_0, z_{\leq,2}, z_{\beta,2}, z_{2,2}, z_{2,3}, z_{3,3}$

Jobs:

$q_1 = S/D$

$q_2 = 0$

$q_3 = (\beta - 1)S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_\beta = (S - s_1)S/D$

$z_{\beta,1} = \beta s_2 S/D$

$z_3 = ((\beta - 1)s_1 + S)/D$

4.5 Tightly grouped processing times, $p \leq p_j \leq \beta p$.

We are given the lower and upper bound on a processing time of a job. We have to solve $m-2$ for all values of $n = 1, \ldots, m-2$. And then we have to solve the case of $n \geq m-1$. The linear programs for $n \leq m$ look naturally:

$$\begin{array}{rcll} \textbf{maximize} & r = q_1 + q_2 + \cdots + q_n & \text{for: } n \leq m \ \ (\text{or } n \leq 2m-2) & \\ \textbf{subject to} & & & \\ 1 & = & s_1 O_n + s_2 O_{n-1} + \cdots + s_n O_1 & (z_{norm}) \\ q_j + \cdots + q_k & \leq & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) \quad 1 \leq j \leq k \leq n \\ q_k & \leq & q_{k+1} & (z_{\leq,k}) \quad 1 \leq k \leq n-1 \\ q_n & \leq & \beta q_1 & (z_\beta)\,. \end{array} \tag{11}$$

In fact this scheme can be used also for $n = m+1, \ldots, 2m-2$, but using $0 = s_{m+1} = s_{m+2} = \ldots$, we can obtain smaller linear programs. The scheme of

these programs follows, the variable q_1 stays for sum of all but last $m-1$ jobs.

$$\begin{array}{rcll}
\textbf{maximize} & r = q_1 + q_2 + \cdots + q_m & & \text{for: } m+1 \le n \le 2m-2 \\
\textbf{subject to} & & & \\
1 & = & s_1 O_m + s_2 O_{m-1} + \cdots + s_m O_1 & (z_{norm}) \\
q_1 + \cdots + q_k & \le & (s_1 + \cdots + s_{k+n-m}) O_k & (z_{1,k}) \quad 1 \le k \le m \\
q_j + \cdots + q_k & \le & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) \quad 2 \le j \le k \le m \\
q_1 & \le & (n-m+1) q_2 & (z_{\le,1}) \\
q_k & \le & q_{k+1} & (z_{\le,k}) \quad 2 \le k \le n-1 \\
(n-m+1) q_m & \le & \beta q_1 & (z_\beta)\ .
\end{array} \tag{12}$$

And at last we have to solve the case where $n \ge 2m-1$. Again we sum all but last $m-1$ jobs to q_1. But now, we know there are at least m jobs summed up in q_1. Thus we can use standard linear program for large number of jobs, but we have to add some or-conditions. The mathematical program consists of linear program (13) with additional conditions (14):

$$\begin{array}{rcll}
\textbf{maximize} & r = q_1 + q_2 + \cdots + q_m & & \text{for: } n \ge 2m+1 \\
\textbf{subject to} & & & \\
1 & = & s_1 O_m + s_2 O_{m-1} + \cdots + s_m O_1 & (z_{norm}) \\
q_1 + \cdots + q_k & \le & (s_1 + \cdots + s_m) O_k & (z_k) \quad 1 \le k \le m \\
q_j + \cdots + q_k & \le & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) \quad 2 \le j \le k \le m \\
q_j & \le & q_{j+1} & (z_{\le,j}) \quad 2 \le j \le m-1 \\
q_m & \le & \beta q_2 & (z_{\beta,2}) \\
m q_m & \le & \beta q_1 & (z_{\beta,1})
\end{array} \tag{13}$$

For every integer $N \ge m$ at least one of following conditions holds:

$$\begin{array}{rcll}
(N+1) q_m & \le & \beta q_1 & (z_{N+1,\beta}) \\
q_1 & \le & N q_2 & (z_{N,2})
\end{array} \tag{14}$$

The conditions $(z_{N,\beta})$ and $(z_{N,2})$ defines the feasibility for q_1 consisting of N jobs. The formulation of (14) shows the forbidden cases. I.e., if there is N for which $(z_{N+1,\beta})$ and $(z_{N,2})$ does not hold, then there is no N, for which both $(z_{N,\beta})$ and $(z_{N,2})$ will hold. This can be easily derived from $q_m \le \beta q_2$ and $\beta > 1$.

Solution for fixed s. This program can be solved efficiently in two steps, having fixed set of speeds $\mathbf{s}$. First we solve the (13). Let the result be $\bar{x}^*$. The $\bar{x}^*$ is either feasible to (13)+(14) and thus desired optimal solution, or there is N, for which both $(z_{N+1,\beta})$ and $(z_{N,2})$ are not satisfied by $\bar{x}^*$, moreover there is at most one such number and we will denote it $\bar{N}^*$. Then the optimal solution to (13)+(14) satisfies either $(z_{\bar{N}^*,2})$ or $(z_{\bar{N}^*+1,\beta})$ by equality. Thus we solve two more linear programs, namely (13)+$(z_{\bar{N}^*,2})$ and (13)+$(z_{\bar{N}^*+1,\beta})$. The maximum of these two solutions is the desired solution of (13)+(14).

4.5.1 $m = 2$

We will solve the case of $m = 2$ parametrically now. We use an equivalent (and more natural) reformulation of the (14):

$$\begin{array}{l}\text{There is integer } N \geq m, \text{ for which both following conditions holds:}\\ \begin{array}{rcll} Nq_m & \leq & \beta q_1 & (z_{N,\beta}) \\ q_1 & \leq & Nq_2 & (z_{N,2}) \end{array}\end{array} \tag{15}$$

We solve the (13)+(15) as a parametrical linear program with an additional parameter N. Then we examine the outgoing solutions and choose N, for which are these solutions minimal.

We obtain following solutions (Note that these hold for $n \geq 2m - 1 = 3$).

Case I

Ratio: $r = \dfrac{(\beta + N)S}{\beta s_1 + NS}$

Conditions:

(A+: II) $Ns_1 \geq \beta s_2$

Common denominator: $D = \beta s_1 + NS$

Nonbasic dual vars: $z_{2,2}, z_{\beta,1}, z_{\beta,2}, z_{<,1}$

Jobs:

$q_1 = NS/D$

$q_2 = \beta S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_1 = (\beta + N)s_2/D$

$z_2 = (\beta + N)s_1/D$

$z_{N,\beta} = s_2/D$

Case II

Ratio: $r = \dfrac{S^2}{s_1^2 + s_2^2 + s_1 s_2}$

Conditions:

(A-: I) $Ns_1 \leq \beta s_2$

Common denominator: $D = s_1^2 + s_2^2 + s_1 s_2$

Nonbasic dual vars: $z_{N,\beta}, z_{\beta,1}, z_{\beta,2}, z_{<,1}$

Jobs:

$q_1 = s_2 S/D$

$q_2 = s_1 S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_1 = s_2 S/D$

$z_2 = z_{2,2} = s_1^2/D$

It can be easily verified, that the ratio is maximized with N as low as possible, i.e., for $N = 2$, even if it means crossing the condition A.

4.5.2 $n = 2$

We also have to solve (11) for $n = 2$ (and the trivial $n = 1$) to complete the analysis of the case for two machines. The solution splits to following two cases:

Case I Ratio: $r = \dfrac{(\beta+1)s_1 S}{(\beta+1)s_1^2 + s_2 S}$

Conditions:

(A+: II) $s_1 \geq \beta s_2$

Common denominator: $D = (\beta+1)s_1^2 + s_2 S$

Nonbasic dual vars: $z_{2,2}, z_{\leq,1}$

Jobs:

$q_1 = s_1 S/D$

$q_2 = \beta s_1 S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{1,1} = (\beta+1)s_2 S/D$

$z_{1,2} = (\beta+1)s_1^2/D$

$z_\beta = s_2 S/D$

Case II Ratio: $r = \dfrac{s_1 S}{s_1^2 + s_2^2}$

Conditions:

(A-: I) $s_1 \leq \beta s_2$

Common denominator: $D = s_1^2 + s_2^2$

Nonbasic dual vars: $z_\beta, z_{\leq,1}$

Jobs:

$q_1 = s_2 s_1/D$

$q_2 = s_1^2/D$

U.b. coefficients:

$z_{norm} = -r$

$z_{1,1} = z_{2,2} = s_2 S/D$

$z_{1,2} = s_1(s_1 - s_2)/D$

4.5.3 Resulting formula for two machines

The overall ratio for two machines splits into four cases, depending on which sequence of jobs is the worst satisfying $p \leq p_j \leq \beta p$. The sequences (for $p = 1$) are: $(1, 1, 2\frac{s_1}{s_2}), (1, 1, \beta), (1, \frac{s_1}{s_2}), (1, \beta)$. Here we give the explicit formula after taking the maximum, to show how complex may be the result even if both relevant solutions of corresponding linear programs split only to two cases each.

$$r^{p\leq p_j\leq \beta p}(s_1, s_2, \beta) = \begin{cases} \dfrac{S^2}{s_1^2 + s_2^2 + s_1 s_2} & \text{for } 2s_1 \leq \beta s_2 \\ \dfrac{(\beta+2)S}{\beta s_1 + 2S} & \text{for } \begin{cases} 2s_1 \geq \beta s_2 \\ s_1 \leq \beta s_2 \\ 2(s_1 - s_2) \leq \beta s_2 \end{cases} \\ \dfrac{(\beta+2)S}{\beta s_1 + 2S} & \text{for } \begin{cases} s_1 \geq \beta s_2 \\ \beta s_1 - 2s_2 \leq \beta s_2 \end{cases} \\ \dfrac{s_1 S}{s_1^2 + s_2^2} & \text{for } \begin{cases} 2s_1 \geq \beta s_2 \\ s_1 \leq \beta s_2 \\ 2(s_1 - s_2) \geq \beta s_2 \end{cases} \\ \dfrac{(\beta+1)s_1(s_1+s_2)}{(\beta+1)s_1^2 + s_2 S} & \text{for } \begin{cases} s_1 \geq \beta s_2 \\ \beta s_1 - 2s_2 \geq \beta s_2 \end{cases} \end{cases}$$

5 Techniques for solving the Parametrized LPs

The solutions above were obtained by search through a large number of possibilities (10^4). This is impossible to do manually and thus we developed a method how to filter out most of the invalid possibilities by a computer. Remaining number of possible solutions is small enough to be solved by a man. Mathematical software Maple 9.5 was used, but any modern algebraical software should do the task. The description of the method follows.

We use the notation $\{\max \mathbf{c}^T\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$ for the primal linear program, and $\{\min \mathbf{y}^T\mathbf{b} \mid \mathbf{y}^TA = \mathbf{c}^T, \mathbf{y} \geq \mathbf{0}\}$ for the corresponding dual linear program. W.l.o.g., the number of the primal variables (the dimension of $\mathbf{x}$) is smaller than or equal to the number of the dual variables (the dimension of $\mathbf{y}$). (That is also the case of the linear programs that we solve.)

We use the duality of the linear programing [21], i.e., if there is an optimal solution to the primal program, then there is a pair of the primal and the dual basic solutions which are optimal. Then we use the dual complementary slackness: the primal inequalities not satisfied by equality imply zero values of the corresponding dual variables. We also use the fact it suffices to examine the vertices of the polytope. We know that the result is bounded, because there is universal upper bound on competitive ratio and the input sequence with at least one nonzero job gives a positive lower bound. Thus we take the set of the dual variables and we generate all subsets of cardinality equal to the dimension of the linear program (which is the number of the primal variables for all linear programs that we examine). We get all points of intersections of the conditions this way. We call them the solution candidates. Then we have to find the points that are feasible and optimal for some valid values of input parameters. From the duality slackness conditions, there is one to one mapping between the solution candidates of primal program and the solution candidates of the dual one. We let the computer automatically examine these solution candidates.

Now we stick to one arbitrary fixed subset Y of the dual variables and we describe how the computer helps us to examine the solution pair induced by this subset. Let Y be a square matrix with $y_{i,i} = 1$ if $i \in Y$ and $y_{i,j} = 0$ otherwise. Now we have the primal candidate solution satisfying system of equations $YA\mathbf{x} = Y\mathbf{b}$ and the candidate dual solution satisfying $\mathbf{y}^TA = \mathbf{c}^T$ & $\mathbf{y}^T(I-Y) = 0$. I.e., we set a primal inequality to equality if it corresponds to some selected dual variable. We set the not selected dual variables to zero and we change the dual inequalities to equations. We solve these two sets of linear equations using Maple, and then we examine this solution pair.

At first we try to filter out the infeasible solution pairs. The feasibility domain of the solution pair is intersection of feasibility domains of both solutions, because feasibility of both solutions for some fixed values of the parameters (the machine speeds) implies also their optimality. So how is our domain of feasibility defined? The primal solution is feasible when all inequalities (namely the inequalities corresponding to the not selected dual variables) are satisfied. The dual solution is feasible when all variables are nonnegative.

It may happen that either the primal or the dual candidate solution does

not exist, i.e., the system of equations has no solution. But we already know that optimal competitive ratio is bounded, which contradicts feasibility of such a solution pair, we eliminate it. The positive lower bound also contradicts feasibility of the solution pair, which has zero value of the resulting competitive ratio.

Now we examine the domain of feasibility of both primal and dual candidate solutions. We developed a heuristic that uses the inequalities between the parameters (i.e., the speeds of the machines, the inequalities are of the form $s_i \geq s_{i+1}$ and $s_i \geq 0$) and determines the validity of the given inequality. The outcome of this heuristic is one of the three cases: (i) surely always valid, (ii) surely always invalid or (iii) there may be values of parameters for which is the inequality valid and another values for which is the inequality invalid. Note that inequality that is always valid or always invalid may be so complex that our heuristic evaluates it as the third case. Our heuristic also uses the factorization of polynomials to eliminate factors that are always positive or always negative. This decreases the polynomial degree of the inequality. So our heuristic may return a simpler inequality that is equivalent to the original one in the third case.

The feasibility domain is given as a set of inequalities. We use our heuristic on them. If we find an inequality that is always invalid (i.e., for all valid values of parameters), we eliminate such a solution pair for infeasibility. If we do not eliminate the pair, we eliminate the inequalities that are surely always valid, and we replace the inequalities with the simpler versions, if our heuristic finds some. We try to further eliminate the solution pair for infeasibility, or to simplify the inequalities defining the feasible region.

Now we are done with single inequalities. So we consider pairs of inequalities. We already have the set of inequalities reduced only to inequalities that may be invalid for some values of parameters. A pair of such inequalities may be in contradiction, then the solution pair is infeasible. Or one inequality may be implied by another one, then we reduce the set of inequalities defining the feasible region. To test the contradiction or the implication, we simply try to add or subtract the conditions, one of them possibly multiplied by a factor from some small predefined set of factors. We test the result for invalidity using our heuristic again.

After all these computations are done, there remain several solution pairs that have to be eliminated by hand. There may be a condition too complex for our heuristic, or there may be three or more conditions in contradiction. Also, our set of factors for testing contradiction may not contain the factor needed to prove the contradiction of the two conditions. Number of these solution pairs vary, but in general there were fewer such solution pairs than the valid ones. The tools that we developed for the automated part are also useful here.

At last, sometimes there are more solution pairs with the same competitive ratio. Domains of feasibility of such pairs may overlap, and sometimes they do. But in all the examined cases there was one or several non overlapping solution pairs, that covered the whole domain of such a formula for the competitive ratio, while the remaining pairs were superfluous.

We finish our inspection of solutions by finding which cases are neighbor by which inequality, thus it can be easily verified (even without using the computer), that the feasibility domains cover the set of all valid values of parameters (the speeds of machines).

Conclusions. We solve the special cases of $m = 3$ and $m = 4$ for the online scheduling and for semi-online scheduling with known sum of processing times. The online scheduling on four machines was more demanding on the computer, as there was $\binom{12}{5} = 792$ basic solution candidates, all but six were found infeasible automatically by computer. Four of the remaining six give the four cases of the optimal competitive ratio. On the other hand, the semi-online scheduling with known sum of processing times for prefixes of three jobs (the hardest case when solving four machines), has only $\binom{10}{5} = 252$ basic solution candidates, all but 20 were found infeasible, and the remaining 20 cases had to be processed manually. Only seven of them are relevant for the optimal competitive ratio. Solving these cases exactly is now possible only using our method (or a similar one), because of the amount of mathematical (algebraic) operations that must be done to go through all the cases. Our method can be further improved, but this will not improve our results dramatically, because of the exponential case explosion. This work also shows that the complexity of exact formulas of competitive ratio grows dramatically with the number of machines.

Acknowledgments. This research was partially supported by Institute for Theoretical Computer Science, Prague (project 1M0545 of MŠMT ČR) and grant IAA100190902 of GA AV ČR. The author is also partially supported by Institutional Research Plan No. AV0Z10190503.

References

[1] D. Du. Optimal preemptive semi-online scheduling on two uniform processors. *Inform. Process. Lett.*, 92(5):219–223, 2004.

[2] T. Ebenlendr, W. Jawor, and J. Sgall. Preemptive online scheduling: Optimal algorithms for all speeds. *Proc. 13th European Symp. on Algorithms (ESA)*, volume 4168 of *Lecture Notes in Comput. Sci.*, pages 327–339. Springer, 2006.

[3] T. Ebenlendr and J. Sgall. Optimal and online preemptive scheduling on uniformly related machines. *Proc. 21st Symp. on Theoretical Aspects of Computer Science (STACS)*, volume 2996 of *Lecture Notes in Comput. Sci.*, pages 199–210. Springer, 2004.

[4] T. Ebenlendr and J. Sgall. Semi-Online Preemptive Scheduling: Online Algorithm for All Variants (conference version). *Proc. 26st Symp. on Theoretical Aspects of Comput. Sci. (STACS)*, volume 09001 of *Dagstuhl Seminar Proceedings*, pages 346-360, IBFI, 2009.

[5] T. Ebenlendr and J. Sgall. Semi-Online Preemptive Scheduling: Online Algorithm for All Variants (full version). To appear in *Theory of Computing Systems – Special Issue of STACS 2009* – 2010.

[6] L. Epstein and L. M. Favrholdt. Optimal preemptive semi-online scheduling to minimize makespan on two related machines. *Oper. Res. Lett.*, 30:269–275, 2002.

[7] L. Epstein and J. Sgall. A lower bound for on-line scheduling on uniformly related machines. *Oper. Res. Lett.*, 26(1):17–22, 2000.

[8] T. F. Gonzales and S. Sahni. Preemptive scheduling of uniform processor systems. *J. ACM*, 25:92–101, 1978.

[9] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical J.*, 45:1563–1581, 1966.

[10] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.*, 17:263–269, 1969.

[11] Y. He and G. Dósa. Semi-online scheduling jobs with tightly-grouped processing times on three identical machines. *Discrete Appl. Math.*, 150(1):140–159, 2005.

[12] Y. He and Y. Jiang. Optimal algorithms for semi-online preemptive scheduling problems on two uniform machines. *Acta Inform.*, 40:367–383, 2004.

[13] Y. He and Y. Jiang. Optimal algorithms for semi-online preemptive scheduling problems on two uniform machines. *J. Comput. Sci. and Technol.*, 19:733–739, 2004.

[14] Y. He and G. Zhang. Semi on-line scheduling on two identical machines. *Computing*, 62(3):179–187, 1999.

[15] E. Horwath, E. C. Lam, and R. Sethi. A level algorithm for preemptive scheduling. *J. ACM*, 24:32–43, 1977.

[16] Y. Jiang, Y. He. Optimal semi-online algorithms for preemptive scheduling problems with inexact partial information. *Theoret. Comput. Sci.*, 44(7-8):571–590, 2007.

[17] H. Kellerer, V. Kotov, M. G. Speranza, and Z. Tuza. Semi on-line algorithms for the partition problem. *Oper. Res. Lett.*, 21:235–242, 1997.

[18] S. Seiden, J. Sgall, and G. J. Woeginger. Semi-online scheduling with decreasing job sizes. *Oper. Res. Lett.*, 27:215–221, 2000.

[19] Z. Tan and Y. He. Semi-on-line problems on two identical machines with combined partial information. *Oper. Res. Lett.*, 30:408–414, 2002.

[20] Z. Tan and Y. He. Semi-online scheduling problems on two identical machines with inexact partial information. *Theoret. Comput. Sci.*, 377(1-3):110–125, 2007.

[21] Vazirani, Vijay V., Approximation algorithms, Ch. 12: LP duality. Springer, 2001.