Charles University
Faculty of Mathematics and Physics

DOCTORAL THESIS

Tomáš Ebenlendr

Combinatorial algorithms for online problems: Semi-online scheduling on related machines

Institute of Mathematics of the Academy of Sciences
of the Czech Republic
Institute for Theoretical Computer Science

Advisor: Doc. RNDr. Jiří Sgall, DrSc.
Branch: I4 - Discrete Models and Algorithms

2010

Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DISERTAČNÍ PRÁCE

Tomáš Ebenlendr

Kombinatorické algoritmy se zaměřením na online problémy: Semi-online rozvrhování na strojích s různými rychlostmi

Matematický ústav Akademie věd České Republiky
Institut teoretické informatiky

Školitel: Doc. RNDr. Jiří Sgall, DrSc.
Obor: I4 - diskrétní modely a algoritmy

2010

Acknowledgements

I am thankful to my advisor Jiří Sgall, especially for guiding my PhD. studies and my research. He also helped me to improve quality of this thesis, by advice on structure of the thesis, reading it and pointing out unclear or wrong parts.

I would like to thank Institute of Mathematics of the Academy of Sciences and Department of Applied Mathematics of Faculty of Mathematics and Physics of the Charles University. I enjoyed to do my research with people from these institutions.

This thesis was partially supported by Institute for Theoretical Computer Science, Prague (project 1M0545 of MŠMT ČR), grant IAA100190902 of GA AV ČR, grant 166610 of GA UK ČR and Institutional Research Plan No. AV0Z10190503.

I declare, that I wrote this doctoral thesis by myself and only by the use of cited sources. I agree with lending and publishing this work.

Prohašuji, že jsem svou disertační prăci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčovaním práce a jejím zveřejňováním.

V Praze dne Mgr. Tomáš Ebenlendr

Title: Combinatorial algorithms for online problems: Semi-online scheduling on related machines

Author: Mgr. Tomáš Ebenlendr

Department: Institute of Mathematics of the Academy of Sciences of the Czech Republic
Institute for Theoretical Computer Science

Advisor: Doc. RNDr. Jiří Sgall, DrSc.

Advisor's email: sgall@kam.mff.cuni.cz

Abstract

We construct a framework that gives optimal algorithms for a whole class of scheduling problems. This class covers the most studied semi-online variants of preemptive online scheduling on uniformly related machines with the objective to minimize makespan. The algorithms from our framework are deterministic, yet they are optimal even among all randomized algorithms. In addition, they are optimal for any fixed combination of speeds of the machines, and thus our results subsume all the previous work on various special cases. We provide new lower bound of 2.112 for the original online problem. The (deterministic) upper bound is $e \approx 2.718$ as there was known e-competitive randomized algorithm before.

Our framework applies to all semi-online variants which are based on some knowledge about the input sequence. I.e., they are restrictions of the set of valid inputs. We use our framework to study restrictions that were studied before, and we derive some new bounds. Namely we study known sum of processing times, known maximal processing time, sorted (decreasing) jobs, tightly grouped processing times, approximately known optimal makespan and few combinations. Based on the analysis of this algorithm, we derive some global relations between various semi-online restrictions.

The framework uses linear programs to compute the optimal competitive ratio for given input environment as our algorithms need to know this value. The environment is described by some parameters (speeds of machines), these parameters play role of constants in the linear programs. We developed a technique to symbolically solve small linear programs. We obtain formulas in the parameters for competitive ratio in the special case of up to four machines this way.

The last chapter a provides new lower bound of 2.564 on competitive ratio of deterministic online nonpreemptive scheduling.

Keywords: scheduling, online, semionline, linear programming

Název práce: Kombinatorické algoritmy se zaměřením na online problémy: Semi-online rozvrhování na strojích s různými rychlostmi

Autor: Mgr. Tomáš Ebenlendr

Katedra (ústav): Matematický ústav Akademie věd České Republiky
Institut teoretické informatiky

Školitel: Doc. RNDr. Jiří Sgall, DrSc.

E-mail školitele: sgall@kam.mff.cuni.cz

Abstrakt

Hlavním výsledkem této práce je konstrukce optimálních algoritmů pro celou třídu rozvrhovacích problémů. Tato třída zahrnuje většinu zkoumaných semi-online variant preemptivního rozvrhování na strojích s různými rychlostmi s cílem minimalizovat délku rozvrhu. Takto zkonstruované algoritmy jsou deterministické, nicméně dosahují optimální kompetitivní poměr i mezi pravděpodobnostními algoritmy. Navíc jsou optimální i pro libovolnou pevnou kombinaci rychlostí strojů, proto lze naši konstrukci uplatnit i na veškeré dříve studované speciální případy. Ukážeme nový dolní odhad 2.112 pro obecné online rozvrhování. Deterministický horní odhad $e \approx 2.718$ pak plyne z dřívější existence e-kompetitivního pravděpodobnostního algoritmu.

Zmíněnou konstrukci lze aplikovat ve všech semi-online variantách, které jsou založeny na znalosti o vstupní sekvenci. Ty lze chápat jako omezení množiny platných vstupů. Tuto konstrukci pak použijeme ke studiu dříve zkoumaných omezení, čímž získáme nové odhady kompetitivního poměru. Jmenovitě zkoumáme známou sumu velikostí úloh, známou největší úlohu, (sestupně) setřízené úlohy, úlohy přibližně stejné velikosti, přibližně známou délku rozvrhu a několik jejich kombinací. Na základě analýzy algoritmu z naší konstrukce dostaneme několik obecných vztahů mezi některými semi-online omezeními.

Naše konstrukce používá lineární programy pro výpočet optimálního kompetitivního poměru pro prostředí zadané na vstupu, protože naše algoritmy potřebují znát tuto hodnotu. Prostředí je zadáno parametry (především rychlostmi strojů), tyto parametry mají funkci konstant v řešených lineárních programech. Vyvinuli jsme techniku pro symbolické řešení malých lineárních programů a tou jsme získali vzorce dávající optimální kompetitivní poměr, do kterých stačí dosadit rychlosti pro prostředí s nejvýše čtyřmi stroji.

V poslední kapitole pak dokážeme nový dolní odhad 2.564 pro deterministické online rozvrhování bez preempcí.

Klíčová slova: rozvrhování, online, semionline, lineární programování

Published results

This doctoral thesis is a superset of following results. First result was the semi-online algorithm for known optimal makespan in the author's master thesis and it was published on the international conference STACS [ES04] and later also in Journal of Scheduling [ES09a]. This result was later generalized to an optimal online algorithm [EJS06] (conference ESA) and [EJS09] (journal Algorithmica). At last it was further generalized to the whole class of semi-online problems in [ES09b] (conference STACS)) and [ES10] (journal Theory of Computer Systems). The special cases of a small number of machines was published on the conference PPAM [Ebe10].

Contents

Chapter 1

Introduction

Scheduling is now a classical problem from the area of combinatorial optimization. Approximation and online algorithms for scheduling are extensively studied, because most scheduling problems are (NP-)hard to solve exactly and because many scenarios deal with input coming in parts, with the requirement of immediate output for each part. There is a strong connection between approximation and online algorithms because many natural approximation heuristics for scheduling are in fact (semi-)online algorithms. Our work studies online and semi-online algorithms for scheduling and their possibilities.

Our online model belongs to *one-by-one* scheduling, i.e., the algorithm sees only one job and it has to decide its schedule immediately and irrevocably. The algorithm has always possibility to start the job from the beginning of the time if there is a machine where the job can be scheduled. This particular variant was studied for a long time, the first well-known algorithm for online scheduling is the Graham's [Gra66] list scheduling algorithm, which is 2-competitive on identical machines.

We are also interested in the semi-online scheduling. The semi-online algorithms work in the same setting as the online algorithm with the exception of some advantage over the online algorithm. We study advantages that are based on some knowledge of the input sequence. One such variant is scheduling with the known sum of processing times, where the algorithm knows the total processing time in advance. We study also other knowledge-based semi-online variants.

We want to minimize makespan, that is the last time when a machine processes a job. This scenario is also called load balancing. We consider machines with different speeds (uniformly related machines) both with and without preemptions. Allowing preemptions means that the algorithm is allowed to interrupt a job at any time and continue the processing of this job on another machine or at a later time.

Our main result is a framework of optimal algorithms when preemptions are allowed. This framework applies not only to online scheduling, but also to various semi-online variants. Moreover these algorithms are deterministic, yet optimal among randomized algorithms. I.e., randomization does not help these problems. The framework constructs (deterministic) algorithms with competitive ratio given as a parameter. These algorithms do not fail (and maintain given competitive ratio) if there exist any (possibly randomized) algorithm with this ratio for the given set of machines. I.e., if we know any R-competitive (randomized) algorithm, then we can run our (deterministic) algorithm with the parameter R and it successfully maintains to be R-competitive.

We also developed linear programs that compute the optimal competitive ratio for an arbitrary fixed set of speeds for online scheduling and all semi-online variants studied in this work. So the algorithms from our framework may solve the linear programs in its initialization stage to compute the parameter R. We obtain the exact value of R for every given set of machines this way.

The structure of the worst set of machines is unknown to us, thus we don't know the exact number of overall competitive ratio. We use the above mentioned linear programs to obtain new overall lower bounds for some of the semi-online variants. This cannot be done directly, as the speeds of the machines determine the linear inequalities. We obtain a quadratic (non-convex) program by taking the speeds as variables and we try to find some local optimum numerically by a computer. Verification of these bounds is just checking of the inequalities as we provide the values of all involved variables. Moreover these variables directly correspond to the input instance (the speeds of machines and the sizes of jobs). An important result is improving a lower bound of 2 based on a geometric sequence of machines and jobs [ES00] to a bound of 2.112 based on the input sequence that was found by a computer.

We show that one of the studied semi-online restrictions does not help to the algorithm in general, i.e., we show how to modify any hard input sequence to satisfy that particular restriction.

We provide new overall upper bound for one semi-online case, where the competitive ratio can be computed directly without solving linear programs. The online case as well as all semi-online cases are upper bounded by e-competitive randomized algorithm for online variant from the author's master thesis [ES04]. Existence of that algorithm proves that algorithms from our framework are at worst e-competitive. Note that our algorithms are deterministic.

We can also ask for closed formulas for the competitive ratio for small numbers of machines. We can acquire them by solving the linear programs symbolically (i.e., we let the parameters — the speeds of machines — to be variables) and we do this for up to four machines. We know that an optimal solution is a vertex of the bounding polytope. This polytope is just the intersection of halfspaces which represent the constraints. Then the vertices are intersections of hyperplanes — boundaries of these halfspaces. We use exhaustive search over all points that are generated by all possible intersections of dimension zero of these hyperplanes. There can be a large number of them (thousands), but nowadays algebraic software can help us to filter out most of the intersections that do not correspond to an optimal solution for any valid values of parameters (machine speeds). So we are left with a few (typically ten to twenty) intersections and we can do the final search by hand. Note that the result corresponds to several intersections, the optimality of each of them depends on the actual values of parameters. Formula of the resulting competitive ratio which corresponds to one of the intersections is a ratio of two polynomials, because it is a solution of a set of linear equations. We analyzed the cases of small machines for most of the studied semi-online variants as well as for the online scheduling.

We augment the work by a new lower bound for the case where preemptions are not allowed. This bound is based on counting how many jobs in a geometric sequence fit on one machine. Then taking the sum over all machines and comparing this sum to the number of jobs in the sequence gives a bound. If we let the common ratio of the geometric sequence to limit to 1, we obtain our new bound. This bound is in the form of a constraint on the value of a definite integral. We evaluate this bound numerically and we obtain the value of 2.564. The previous lower bound of 2.438 [BCK00] was achieved by a computer verification of a large set of configurations of the output schedule.

1.1 What I have learned to

This work is about online scheduling. I learned about scheduling a lot, but I want to mention what I have learned besides the online scheduling. I will also not mention secondary skills like making presentations at conferences, improving my English, etc.

I had only a bare knowledge about linear programming before I started the research related to this work. I knew about practical usefulness of it, but I was not aware of any case where the linear program would be useful in theory. At least I was not able to imagine how finding a solution to a particular linear program would be helpful for theory. We use linear programs which prove existence of an efficient algorithm. Our linear program in fact optimizes the only lower bound with the constraint that the bound is correct. Thus the optimal solution is the best achievable quality of algorithms that solve our problem. The nice thing is that we do not need linear programming to check the feasibility of the solution, i.e., to check the correctness of our result. Also the optimality of the result can be checked if we provide the corresponding dual solution.

My research lead to linear programs already in its early stage, namely when we found the proof of the optimality of our new algorithm. I had to explore linear programming more when trying to understand the limits of our algorithm. I learned how natural principle is the complementary slackness of dual solutions. We were then able to use an algebraic software to obtain the formulas of optimal solutions of our small sized parametric linear programs. We also used a computer to find some local optima for nonlinear programs when searching for “worst” combination of the parameters. Obtained solution is verifiable lower bound (because the feasibility of the solution is naturally easily verifiable).

1.2 Brief introduction to the area

Following paragraphs give the basic description of the area for readers that are not familiar with it. The variants not containing emphasized definitions of terms are not studied in this work.

1.2.1 Scheduling

The goal of scheduling is to schedule some jobs to one or more machines. We distinguish many variants of scheduling based on the constraints of the schedule and the objective function. In the basic problem, we consider the jobs with only one property, namely their processing time. The constraint on the schedule is that one job has to be assigned to one time interval (*timeslot*) of corresponding length and one machine. The intervals of different jobs assigned to the same machine may not overlap. We call this variant *nonpreemptive* scheduling on *identical* machines. The most studied goal function is the length of the schedule, i.e., the last end of any timeslot on any machine used in the schedule. This length is called *makespan.* There are many modifications to the basic problem, we give here a brief list, starting with modifications used in this thesis.

The machines may have different speeds and a job assigned to a machine with double speed needs half sized timeslot. The speeds of the machines may be fixed, then we talk about *(uniformly) related machines.* The term unrelated machines is used for the variant where speed of machine depends on the individual job, i.e., we have a matrix of processing times telling for each job how long it will be processed when scheduled to each machine. We schedule always on related machines throughout this work.

We can also consider *preemptive* jobs. The algorithm is allowed to split such jobs to pieces and assign the pieces separately, with the restriction that two pieces of same jobs cannot run at once. I.e., the job may be interrupted and scheduled on other machine or its execution may be postponed. The preemptions are allowed in most of this work, only the last chapter gives a lower bound for nonpreemptive scheduling.

Now we continue with modifications that are not used in this work, we start with various properties of jobs.

An important modification introduces release times. (The timeslot assigned to the job may not start earlier than the release time of the job). Deadlines can be introduced as well, but then we typically have to allow jobs to be not scheduled at all, and we want to maximize the number of scheduled jobs.

Another possible modification of jobs is parallelism, the jobs may be allowed or forced to use multiple machines. Parallel jobs can be assigned one timeslot and a subset of machines. We say that jobs are malleable, if we allow the algorithm to choose the size of the set of machines assigned to a job and the processing time of the job depends on this number.

The jobs may be split into tasks that have to be processed on different machines, i.e., each machine is specialized to do one task. We call this variant shop scheduling. There are also several variants depending on the restrictions on the order of the tasks in the schedule of a single job.

Different objective functions are also studied. We can consider either the last time a job is running on some machine and take any norm of these numbers over all machines, or we can take any norm of completion times of all jobs. We can also measure another time, e.g., waiting time of a job (i.e., the difference between the release time and the start one of the job). We already mentioned that there are scenarios, where no valid schedule for all jobs may exist, and then we are typically interested in the number (or sum of weights) of jobs that can be scheduled in a valid schedule.

We have to note that the basic variant, i.e., minimizing the makespan on identical machines, is NP-complete using a trivial reduction from the problem of knapsack. Most other scheduling variants are NP-complete as well. Thus the approximation algorithms are studied extensively in this area.

1.2.2 Approximations

The approximation algorithms are typically measured by so-called approximation ratio. We fix an input and take the value of the objective function for the output of the algorithm and divide it by the objective value of the optimal solution. Better algorithms have the ratio closer to 1. For randomized algorithms we typically take the expectation of the objective value over the random bits of the algorithm. We assume without loss of generality that the approximation ratio is greater than 1. This does not hold for maximization problems, but we turn them into minimization problems simply by considering reciprocal of the objective value. We do not allow negative values of the objective function for relative error to make sense.

The worst-case approximation ratio is the most studied one. I.e., we say that the algorithm is an α-approximation if the approximation ratio at most α for every input. If we say that some problem is α-inapproximable, then we mean that there is no polynomial algorithm with the approximation ratio at most α.

1.2.3 Online algorithms

Sometimes we do not have full information about the input when we start to solve the problem and we have to adjust our solution according to the information that comes over time. Algorithms for such problems are called *online*. Formally, the algorithm has to respond with some immediate output to each part of the input. The individual problem gives restrictions on valid output of the algorithm, namely on the consistence of the output.

In the case of scheduling the algorithm may not reschedule the job, once the job was scheduled. There are two main online paradigms used in scheduling.

The online nature may be orthogonal to the time of the algorithm, the jobs may be submitted one by one from a list, and the algorithm has to create a irrevocable schedule for every job without seeing next jobs in the list. We call this *one-by-one* online scheduling. We can imagine a system where the owner pays fixed price per time unit if there is at least one machine running. The owner sells timeslots on machines to the job owners. Every job owner wants to know immediately when his job will run.

The other paradigm is that the online nature of the algorithm is bound to the time used in the schedule. Then the algorithm may not reschedule jobs that were started in the past. Moreover the algorithm cannot interrupt the jobs unless the preemption is allowed. The jobs are typically unknown to the algorithm before their release times. The jobs may also have unknown processing times (non-clairvoyant scheduling) until they are finished. This paradigm corresponds to standard scenario where jobs come over time.

We measure the *competitive ratio* of the algorithm. That is nothing but the (worst-case) approximation ratio described in the previous subsection, i.e., for every input we take the output of the algorithm and the optimal solution and take the ratio between the values of objective function applied to these two solutions. We take the expectation over random bits for randomized algorithms again. Having the ratio for every possible input, the competitive ratio of the algorithm is the worst one.

Sometimes the competitive ratio is generalized by allowing a constant to be subtracted from the objective value of the algorithm's output. This generalization gives the same values of competitive ratios in makespan scheduling, as every instance can be scaled to arbitrary large numbers (processing times).

1.3 Our model

We study online scheduling on uniformly related machines. It means that p/s of time is needed to process a job with processing time p on a machine with speed s. We are interested in the makespan (the length of the schedule). We allow preemption in Chapter 2, which means the execution of the job may be interrupted and resumed on other machine or after some delay. We study nonpreemptive scheduling in Chapter 3.

The one-by-one online scheduling is studied in this work, i.e., the algorithm sees only one job on the input and it has to fully determine the schedule of this job before it sees the next job (if there is one). That means the order of the jobs in the input sequence has major influence on the run of the algorithm. We say that the algorithm is in the *step j* if it schedules jth job. The word *time* is reserved for the time in the schedule (e.g., starting and finishing time of a job) and is totally unrelated to the steps of the algorithm.

We also study various semi-online variants of the online scheduling problem, that means we give some advantage to the online algorithm. The semi-online problems are in fact online problems, which were derived from some original online problem. We study problems derived by restricting the set of valid inputs. The semi-online algorithms are sometimes categorized as approximations, because the algorithms solving these problems are not valid in the original online problem.

1.4 Previous and our results

The first remarkable scheduling algorithm, which is described by Graham [Gra66], is already online, $(2-\frac{1}{m})$-competitive greedy algorithm for makespan scheduling on m identical machines. This problem and algorithm is also known as list scheduling. This algorithm does not use preemptions, but the competitive analysis holds also in the preemptive setting.

Offline scheduling

The offline scheduling with the makespan objective is well understood, and results for uniformly related machines were usually obtained using similar methods as for identical machines. Algorithms for exact solutions are known when preemptions are allowed for identical machines [McN59] as well as for related machines [HLS77, GS78], however the algorithm described in Section 2.2 can be also used here. It simplifies exactly to the algorithm from the master thesis of the author [ES04]. We consider this simplified algorithm much easier to understand than the previous algorithms. The problem is NP-hard when preemptions are not allowed, but polynomial approximation schemes are known for identical machines [HS87] as well as for related machines [HS88]. That means that for arbitrarily small fixed ϵ there exist a polynomial algorithm with approximation ratio $(1+\epsilon)$. Thus we can solve the problem with an arbitrary fixed precision.

Online nonpreemptive scheduling

The online makespan scheduling exposes a different situation. Let us look at the nonpreemptive version first. The greedy algorithm (list scheduling) is an optimal deterministic algorithm for two and three identical machines [FKT89] achieving competitive ratios of $\frac{3}{2}$ and $\frac{5}{3}$ respectively. No tight results are known for more machines, but there are known better algorithms than the greedy one. For $m=4$, the best known algorithm is by Chen et al. [CvVW94b] with competitive ratio of 1.733,the corresponding lower bound of $\sqrt{3}=1.732$ is by Rudin and Chandrasekaran [RC03]. The latest deterministic algorithm for general m of identical machines is by Fleischer and Wahl [FW00] achieving a competitive ratio of 1.9201. The best known lower bound is by Rudin [Rud01] and has value of 1.880.

Optimal randomized algorithm is known only for two identical machines, see Bartal et al. [BFKV95]. The lower bound of $1+((m/(m-1))^m-1)^{-1}$ for any m identical machines was obtained by Chen et al. [CvVW94a] and Sgall [Sga97] independently. This and all the previously known lower bounds bounded also preemptive algorithms. A better lower bound is known for $m=3$, with value of $27/19+\epsilon$ for some small fixed ϵ by Tichý [Tic04]. This bound uses the fact that jobs cannot be split and thus it is the first bound that does not work in the preemptive setting. Randomized algorithms better than deterministic are also known for $m=3,\ldots,7$, the result by Seiden [Sei00]. A better algorithm for general m with competitive ratio 1.916 was described by Albers [Alb02]. This algorithm uses only one random bit in its initialization, i.e., we have two deterministic algorithms with the nice property, that the arithmetic mean of their competitive ratios is good for any input sequence.

Much less is known for the case of related machines. Best known general algorithms are 5.828-competitive deterministic and 4.311-competitive randomized one by Berman et al. [BCK00]. They also give the first lower bounds that do not hold for identical machines. The deterministic one has value of 2.438 and is done by computer, it checked a large graph of the output schedules. We present a new lower bound of 2.564 in this work. Our lower bound uses the computer only to do one numerical optimization of a single-variable integral inequality.

The randomized lower bound of 2 (which holds also in the preemptive setting) was constructed by Epstein and Sgall [ES00]. Epstein et al. [ENS$^+$01] give an algorithm for two related machines, they analyze the competitive ratio (and lower bounds) parametrically based on the ratio of the speeds of the two machines. Their algorithm is optimal for equal speeds as well as for one machine twice faster than the other.

Online preemptive scheduling

Now we will describe the situation for preemptive on-line scheduling. Scheduling with preemptions in the basic model is much simpler because we can split the load between machines when scheduling every single job. As we already said the lower bound of $1 + ((m/(m-1))^m - 1)^{-1}$ for any m of identical machines [CvVW94a, Sga97] applies here. Chen et al. also constructed the matching optimal algorithm in [CvVW95].

For $m = 2$ related machines, the optimal algorithm for any set of speeds was given by Wen and Du [WD98] as well as Epstein et al. [ENS$^+$01].

The best previous lower bound for a general number of related machines was the bound of 2 by Epstein and Sgall [ES00]. The author presented in his master thesis the 4-competitive deterministic and the e-competitive randomized algorithms.

We will show a deterministic algorithm that computes the optimal competitive ratio (even for randomized algorithms) for any given set of speeds using the linear programming and then it maintains this competitive ratio for any input sequence of jobs. This means that the randomization does not give any advantage with respect to the makespan for this problem. However we are not able to obtain a new non-trivial overall bound on the outcome of the linear program, thus we know that the overall competitive ratio is somewhere between 2.112 and $e = 2.718$. The lower bound is also our new result. It is a sequence for $m = 200$. Plugging this sequence into the lemma from [ES00] gives the desired bound. We list all processing times of jobs and all speeds of machines, thus verification is simple, although it may be tedious. The speeds and processing times were found by a computer as a local extreme of a non-convex quadratic program. We also analyzed the case of four machines and we identify the formula of the competitive ratio as a function of speeds of machines.

Semi-online restrictions

We will present our algorithm in more general form as a framework to construct the optimal algorithm for many semi-online variants of preemptive scheduling on related machines. Namely these are the variants that can be expressed as restrictions of the set of allowed inputs.

We stress that our semi-online results are only on related machines with preemptions allowed, we will not repeat this in following paragraphs.

Known sum of processing times, denoted $\sum p_j = P$. The algorithm knows the total sum of processing times. That means, for example, that it also knows (for every step) if the job on input is the last (nonzero) one. This restriction, for non-preemptive version on two identical machines, was studied by Kellerer et al. in [KKST97], which is probably also the first paper which studied and compared several notions of semi-online algorithms. The non-preemptive version on identical machines is further studied under the name online partition.

We will show that the overall ratio is the same as in the general online case, while it is much lower for small number of machines. We note that there is 1-approximation possible for two machines and we analyze the case of three and four machines.

Non-increasing processing times, denoted *decr*. Here each subsequent job must be smaller or equal to the previous one. The greedy algorithm (list) on identical machines was already analyzed by Graham [Gra69]. The optimal algorithm for identical machines with preemptions was given by Seiden et al. [SSW00] and for any speed combination of two related machines both without [EF02a] and with [EF02b] preemptions by Epstein and Favrholdt.

We prove that the worst sequence is that of all jobs equal for any sequence of speeds. Moreover the knowledge that all jobs are equal will not help the algorithm thus the competitive ratios of these two cases are equal.

We provide formula giving optimal competitive ratio for any fixed set of speeds and we provide upper and lower bounds on overall competitive ratio.

Known optimal makespan, denoted $C^*_{\max} = T$. The semi-online algorithm that outputs the optimal schedule was the main result of the author's master thesis [ES04], the framework from this work is in fact a deep generalization of that algorithm.

The nonpreemptive version on identical machines is called bin stretching [Eps03].

Known maximal processing time, denoted $p_{\max} = p$. The algorithm knows the value of maximal processing time. It is easy to see that any algorithm that works in the setting where the first job is the maximal one, can be emulated also here, yielding an algorithm with the same competitive ratio. The algorithm just schedules the maximal job virtually in the step zero, i.e., it creates reservation of timeslots for this job. Then the algorithm simply uses this reservation for first job of the maximal processing time. All other jobs are scheduled normally. This restriction was introduced by He and Zhang [HZ99] for non-preemptive scheduling on two identical machines, with the proof that the greedy algorithm (list) is optimal. The complete analysis of the preemptive version on two related machines was given by He and Jiang [HJ04a]. Seiden et al. [SSW00] studied the case of the identical machines and they show that the approximation ratio is the same for known maximal processing time as for the non-increasing processing times.

We show that this is not the case for general speeds. We provide the new lower bound of 1.908, which was found by the computer as a locally worst sequence on $m = 200$ machines. We also give a complete analysis for three and four machines.

Inexact partial information. The knowledge described above may not be given exactly. Here we consider the case when an interval of admissible values is known. For example, the algorithm is given $\bar{P}$ and α, and the following bound on total processing time: $\bar{P} \leq \sum p_j \leq \alpha\bar{P}$. Such variants were studied first by Tan and He [TH07] without preemptions and for two identical machines. The complete analysis of the preemptive version with approximately known optimum and maximal processing time on two machines and approximately known optimum on identical machines was given by Jiang and He [JH07].

We give a complete analysis for an approximately known optimum for three machines. This is denoted $T \leq C^*_{\max} \leq \alpha T$.

Tightly grouped processing times, denoted $p \leq p_j \leq \alpha p$. The bounds on processing times of the jobs are given here. This restriction is introduced by He and Zhang [HZ99]. They prove that the greedy algorithm (list) is optimal for two identical machines without preemption. The complete analysis of the preemptive version on two related machines was given independently by Du [Du04] and He and Jiang [HJ04a]. The case of three identical machines has been also fully analyzed by He and Jiang [HJ04b].

We sketch a complete analysis for two machines, reproving and simplifying the results of [Du04], [HJ04a].

Combined restrictions. The algorithm may be provided with a combination of informations described above. Some combinations were studied by Tan and He [TH02] for two identical machines without preemption.

We present two variants, the knowledge of the total processing time combined either with the known maximal processing time or with non-increasing jobs. We show that overall ratio is the same as if the algorithm does not know the total processing time in advance, although it differs for any fixed set of machines. We provide a formula giving optimal competitive ratio for any sequence of speeds in the case with non-increasing processing times. We also provide full analysis of scheduling on three and four machines in the case with know maximal processing time.

Other semi-online scenarios

Our framework applies directly in the semi-online variants described above. We list other semi-online variants below. Our framework is useful in some of these variants, but it requires some additional work.

Local knowledge. Our framework is designed to work with some global knowledge and all semi-online variants above fall in this category. But there have been studied models that contain some local information also. Our framework can be used there, formally it involves adding some information to each job and reformulating the semi-online restriction as a requirement of consistency on these informations. The algorithm gets all the additional information about the job together with its processing time.

One example is the knowledge that the job is the last one. The additional information is just a single bit indicator for each job. The consistency requirement says that only the last job has this bit set. The information that the job is the last one is not useful by itself. But it can be combined with the knowledge that the last job has to be the maximal one [ZY99, EY07]. We have not performed any calculation.

Buffers. Limited reordering of jobs is allowed here. Namely, the algorithm has a buffer on limited number of jobs and it can store the job there instead of scheduling it. It is allowed to store the job to the buffer and schedule another job from the buffer in the same step. The algorithm has to schedule all jobs from the buffer when there is no further job on input. This variant was introduced for the non-preemptive version by Kellerer et al. [KKST97], later tight bounds on identical machines were given by Englert et al. [EÖW08]. The preemptive version was recently studied by Dósa and Epstein [DE09].

Our framework cannot work here as is. But we can make a semi-online knowledge from the buffering part of each algorithm and use our framework there. It turns out that our framework gives the optimal algorithm for an arbitrary buffering strategy. Then it is much simpler to find the best buffering strategy, namely it can be proved that storing largest jobs works. We will publish the proofs in an upcoming article.

Resource augmentation. The semi-online variants, where the algorithm is given some resources that are not available in the optimal schedule, are also studied. E.g., suppose that we want to know how we can perform by acquiring second set of machines. Such scenario is studied in [DH04] for two identical machines: The algorithm schedules to four machines, but its output is compared to the best schedule which uses only two machines. We give no hope in our framework to be useful here. The proof of the correctness of the algorithm from our framework is tightly connected to the knowledge of the optimal makespan under the conditions that the algorithm has. The resource-augmented algorithms clearly operate under different conditions.

Chapter 2

Preemptive scheduling on related machines

2.1 Introduction

2.1.1 The definition of the model

We study the problem of scheduling of jobs to machines with different speeds. We have a set of m machines $\mathcal{M} = \{M_1, \ldots, M_m\}$, with speeds $s_1, \ldots, s_m$. We can imagine computers with different CPUs here. We are also given a sequence of n jobs $\mathcal{J} = (J_1, \ldots, J_n)$, with processing times $p_1, \ldots, p_n$. Here we can imagine some computations with length known in advance. The jobs may be *preempted* in our model, that means the job may be interrupted and then continued on another machine or after some time. There is no additional cost for preemption in our model. The only restriction is that the job cannot be processed in parallel, i.e., at any given time it is processed by at most one machine.

We are interested in the assignment of the jobs to machines. We call this assignment a *schedule*. Note that this is a partial assignment as jobs are processed only for finite amount of time. Formally we denote this assignment by $\sigma : \mathcal{J} \times \mathbb{R}^+ \to \mathcal{M}$. That means for any positive time τ, job J_j is assigned either to some machine, or not processed at all. We define the shorthand $\sigma_j(\tau)$ for $\sigma(J_j, \tau)$. Job scheduled to machine M_i is processed at speed $s(M_i) = s_i$. Job that is not scheduled at time τ at all is processed at speed $s(\perp) = 0$. If job J_j is properly scheduled, then it is finished at the last time when it is scheduled to some machine. Formally $\int_0^\infty s(\sigma_j(\tau))d\tau = p_j$.

In fact we will study only nice schedules, that are partially constant assignments with small number of constant parts: *timeslots*. The timeslot is simply a time interval on one machine. Thus the integration is just summation over the timeslots. E.g., if job J_j will be scheduled only to machine M_i, in one timeslot, then the length of this timeslot has to be p_j/s_i for the schedule to be valid.

This model is called *preemptive* scheduling. More studied model is *nonpreemptive* scheduling. In that model only one timeslot may be assigned to each job. Thus once started at some machine a job continues running on that machine uninterrupted until it is processed.

So we described the environment, and now we have to say what we want to optimize. In our environment there always exists valid schedule of whole input sequence, thus we measure the quality of the schedule by when are the jobs finished. We denote last time when job is processed by some machine by $C_j^\sigma = \sup\{\tau \mid \sigma_j(\tau) \neq \bot\}$ and we call it completion time of the job. Our goal will be to minimize maximal completion time $C_{\max}^\sigma = \max\{C_j^\sigma \mid j = 1, \dots, n\}$. This number we call *makespan* of a schedule σ. There are also studied models with other goal functions, e.g., total sum of completion times ($\sum_{j=1}^n C_j^\sigma$).

For given set of machines and input sequence $\mathcal{J}$, we will use notation σ^* for optimal schedule and σ^{A} for schedule output of some algorithm A. We will use shorthands $C_{\max}^*[\mathcal{M}, \mathcal{J}] = C_{\max}^{\sigma^*}$ and $C_{\max}^{\mathsf{A}}[\mathcal{M}, \mathcal{J}] = C_{\max}^{\sigma^{\mathsf{A}}}$, for schedules for machines $\mathcal{M}$ on input sequence $\mathcal{J}$ created by optimal offline algorithm and by algorithm A respectively. We will omit $\mathcal{M}$ in both shorthands most time, as there will be only one (possibly general) fixed set of machines in the context.

We also use natural terms *free* or *idle* and the opposite *busy*. Namely the machine M_i is *busy* at time τ if there is job J_j that is scheduled to M_i at time τ, (i.e., $\sigma_j(\tau) = M_i$). If there is no such job, then we say that M_i is *idle* at time τ.

2.1.2 Basic facts

Now we will show how to compute $C_{\max}^*[\mathcal{J}]$. First, $C_{\max}^*[\mathcal{J}]$ can be bounded by the total work done on all machines. I.e.,

$$C_{\max}^*[\mathcal{J}] \geq \frac{\sum_{j=1}^n p_j}{\sum_{i=1}^m s_i}. \tag{2.1}$$

Second, the makespan of the optimal schedule is at least the makespan of the optimal schedule of any ℓ jobs. For $\ell < m$ this latter schedule uses only ℓ fastest machines, so the work of any ℓ jobs must fit on these machines. So,

$$C_{\max}^*[\mathcal{J}] \geq \frac{P_\ell}{\sum_{i=1}^\ell s_i} \qquad \text{for } \ell = 1, \dots, m-1, \tag{2.2}$$

where P_ℓ denotes the sum of ℓ largest processing times in $\mathcal{J}$. We also define S_i as the sum of speeds of i fastest machines. Moreover we use $S = S_m$ and $P = P_n$ for total speed of the machines and for total processing time respectively. The following is a known fact [HLS77, GS78, ES04].

Fact 2.1.1 *The value of $C_{\max}^*[\mathcal{J}]$ is the minimal value that satisfies (2.1) and (2.2). I.e.,:*

$$C_{\max}^*[\mathcal{J}] = \max\left\{\frac{P}{S}, \frac{P_1}{S_1}, \frac{P_2}{S_2}, \dots, \frac{P_{m-1}}{S_{m-1}}\right\} \tag{2.3}$$

The problem does not imply any ordering on the machines, i.e., we can permute the values s_i and we obtain the same problem. We need to compare individual speeds very often, so we order the machines by their speed decreasingly. Moreover we define additional machines of zero speed, to simplify some formulas. I.e.:

$$s_1 \geq s_2 \geq \cdots \geq s_{m-1} \geq s_m \geq 0 = s_{m+1} = s_{m+2} = \cdots \ .$$

Note that then we have $S_1 = s_1$, $S_2 = s_1 + s_2$, and so on.

2.1.3 Online algorithms

This problem as stated above, that is its offline version, is solved in several papers very well [HLS77, GS78, ES04].

The problems, where input come over time and the algorithm has to output some partial output on every input information without a chance to modify what it output in past, are called *online*. They can be further divided to *real-time* problems and *sequential* problems. Now we will describe only the sequential problems as this is also the case of our model.

The input for sequential problem is not bound to time, but it is split into parts, which are ordered in a sequence. The algorithm has to produce output for current part of input before reading the next part. The algorithm cannot change decisions made when solving previous part of inputs, i.e., it's output must be consistent with all previous parts. (The consistency is defined in the solved problem.) This is also our case. The time in the schedule is totally unrelated to the part of the input which the algorithm now reads.

The online restriction is often so strong, that optimal offline solution cannot be achieved in online problem. Then we study *competitive ratio* of the algorithm, that is worst-case ratio between the value of output of the algorithm and the value of the best offline solution.

Online one-by-one scheduling. We consider sequential online problem that arises from our model by dividing the input sequence into single jobs. That is, the algorithm sees only one job on the input. It has to decide its schedule. After the algorithm outputs the schedule for the job, it sees next job. The algorithm does not know the number of jobs and the parameters of jobs in advance. The set of machines is either given as part of the definition of the problem, or it is read by the algorithm together with first job.

The algorithm has to make decisions based on the prefixes of the input sequence, so we introduce following notation: $\mathcal{J}_{[j]}$ denotes the sequence of first j jobs.

Having an algorithm A, we will use $C^{\mathsf{A}}_{\max}$ for makespan of schedule produced by this algorithm. Then we say A is R-competitive, if $C^{\mathsf{A}}_{\max} \leq R \cdot C^{*}_{\max}$ for any input sequence.

The following lemma is due to Epstein and Sgall [ES00]. We include the proof, as it is very helpful in understanding our results.

Lemma 2.1.2 ([ES00]) *For any randomized R-competitive online algorithm A for preemptive scheduling on m machines, and for any input sequence $\mathcal{J}$ we have*

$$\sum_{j=1}^{n} p_j \quad \leq \quad R \cdot \sum_{i=1}^{m} s_i C^*_{\max}[\mathcal{J}_{[n-i+1]}].$$

For non-preemptive scheduling, the same holds if $C^*_{\max}$ refers to the non-preemptive optimal makespan.

Proof: Fix a sequence of random bits used by A. Let T_i denote the last time when at most i machines are running and set $T_{m+1} = 0$. First observe that

$$\sum_{j=1}^{n} p_j \quad \leq \quad \sum_{i=1}^{m} s_i T_i. \tag{2.4}$$

During the time interval $(T_{i+1}, T_i]$ at most i machines are busy, and their total speed is at most $s_1 + s_2 + \cdots + s_i$. Thus the maximum possible work done in this interval is $(T_i - T_{i+1})(s_1 + s_2 + \cdots + s_i)$. Summing over all i, we obtain $\sum_{i=1}^{m} s_i T_i$. In any valid schedule all the jobs are completed, so (2.4) follows.

Since the algorithm is online, the schedule for $\mathcal{J}_i$ is obtained from the schedule for $\mathcal{J}$ by removing the last $i-1$ jobs. At time T_i there are at least i jobs running, thus after removing $i-1$ jobs at least one machine is busy at T_i. So we have $T_i \leq C^{\mathsf{A}}_{\max}[\mathcal{J}_{[n-i+1]}]$ for any fixed random bits. Averaging over random bits of the algorithm and using (2.4), we have

$$\sum_{j=1}^{n} p_j \quad \leq \quad \mathbb{E}\left[\sum_{i=1}^{m} s_i C^{\mathsf{A}}_{\max}[\mathcal{J}_{[n-i+1]}]\right] \quad = \quad \sum_{i=1}^{m} s_i \mathbb{E}\left[C^{\mathsf{A}}_{\max}[\mathcal{J}_{[n-i+1]}]\right].$$

Since A is R-competitive, i.e., $\mathbb{E}[C^{\mathsf{A}}_{\max}[\mathcal{J}_i]] \leq R \cdot C^*_{\max}[\mathcal{J}_i]$, the lemma follows. $\square$

We will show that there is no other bound on competitive ratio:

Theorem 2.1.3 *There exists deterministic algorithm with competitive ratio arbitrarily close to the minimal value that satisfies Lemma 2.1.2.*

2.1.4 Semi-online algorithms

Sometimes we know some information about the input sequence in advance. In fact most semi-online problems are ordinary online problems by definition. The word "semi" means here that the new problem was derived from more general online problem by providing some information about the general input sequence in advance. This may be either a global information (e.g., total processing time of all jobs) or local information (e.g., processing time of the next job.) There are also other "semi"-online relaxations of online problems based on relaxing of the restrictions on output (e.g., buffer on one job).

We will study the semi-online variant of online scheduling defined by some global information in advance. This global information can be viewed as a restriction of set of inputs. In online algorithms we don't worry about time complexity, thus there is nearly no difference between the information provided as a parameter of the problem and the information provided in the first part of the input. (Note that we can deal also with knowledge that is not global. The idea how to convert such knowledge to a global knowledge is described in the results section of the introduction.)

We denote a general restricted set of inputs by Ψ. We call a sequence *partial input* if it is a prefix of some input sequence; the set of partial inputs is denoted by $\mathrm{pref}(\Psi)$. The partial inputs are exactly the sequences that the algorithm can see at some point. A (randomized) semi-online algorithm A with restriction Ψ is an R-approximation algorithm if $\mathbb{E}[C^{\mathsf{A}}_{\max}[\mathcal{J}']] \leq R \cdot C^*_{\max}[\mathcal{J}']$ for any $\mathcal{J}' \in \Psi$. Note that this implies that for any prefix $\mathcal{J}$ of $\mathcal{J}'$, $\mathbb{E}[C^{\mathsf{A}}_{\max}[\mathcal{J}]] \leq R \cdot C^*_{\max}[\mathcal{J}']$.

The algorithm knows current partial input $\mathcal{J}$ and does not know the whole input $\mathcal{J}'$. So it needs to satisfy $\mathbb{E}[C^{\mathsf{A}}_{\max}[\mathcal{J}]] \leq R \cdot C^*_{\max}[\mathcal{J}']$ for any $\mathcal{J}' \in \Psi$ which begins with $\mathcal{J}$. Thus we define optimal offline makespan for any $\mathcal{J} \in \mathrm{pref}(\Psi)$ to be consistent with the information provided to the algorithm:

Definition 2.1.4 *For an input restriction Ψ and an input sequence $\mathcal{J}$, we define the optimal makespan as infimum over all possible end extensions of $\mathcal{J}$ that satisfy Ψ:*

$$C^{*,\Psi}_{\max}[\mathcal{J}] = \inf\left\{C^*_{\max}[\mathcal{J}'] \mid \mathcal{J}' \in \Psi \ \&\ \mathcal{J} \in \mathrm{pref}(\{\mathcal{J}'\})\right\}$$

This definition is meaningful for any input sequence (i.e., for any prefix of some $\mathcal{J} \in \Psi$). For $\mathcal{J} \in \Psi$ the values of $C^*_{\max}[\mathcal{J}]$ and $C^{*,\Psi}_{\max}[\mathcal{J}]$ are equal.

Also Lemma 2.1.2 needs to be reformulated as it is not strong enough in semi-online setting:

Lemma 2.1.5 *Given any randomized R-approximation semi-online algorithm A for preemptive scheduling on m machines with an input restriction Ψ, then for any input sequence $\mathcal{J} \in \Psi$ and for any subsequence of jobs $1 \leq j_1 < j_2 < \cdots < j_k \leq n$ we have*

$$\sum_{i=1}^{k} p_{j_i} \quad \leq \quad R \cdot \sum_{i=1}^{k} s_{k+1-i} C^{*,\Psi}_{\max}[\mathcal{J}_{[j_i]}].$$

Having a witness sequence $\mathcal{J}$ that forbids approximation ratio R according to Lemma 2.1.5, the adversary will submit jobs from this sequence one by one until it finds that makespan of the algorithm's output is larger than $RC^{*,\Psi}_{\max}[\mathcal{J}_i]$, where $\mathcal{J}_i$ is the sequence of jobs submitted so far. Then the adversary has to find sequence $\mathcal{J}'$ which begins with $\mathcal{J}$ and leads to the same makespan: $\mathcal{J}' = \mathrm{argmin}_{\mathcal{J}' \supseteq \mathcal{J}_i \& \mathcal{J}' \in \Psi}\{C^{*,\Psi}_{\max}[\mathcal{J}']\}$. (In fact it suffices to find some good approximation as the minimum needs not to exist). Then the adversary finishes the input sequence with jobs from $\mathcal{J}' \setminus \mathcal{J}_i$. Makespan of the output of the algorithm will not decrease, and the choice of sequence $\mathcal{J}'$ sequence ensures that $C^{*,\Psi}_{\max}$ will not increase. This way adversary wins the game and proves that the algorithm is not R-competitive.

Proof: (of Lemma 2.1.5) Fix a sequence of random bits used by A. Let T_i denote the last time when at most i machines are running the jobs from the subsequence $J_{j_1}, J_{j_2}, \dots, J_{j_k}$. First observe that

$$\sum_{i=1}^{k} p_{j_i} \;\leq\; \sum_{i=1}^{k} s_i T_i. \tag{2.5}$$

During the time interval $(T_{i+1}, T_i]$ at most i machines are busy with jobs from $(J_{j_\ell})_{\ell=1}^{k}$, and their total speed is at most $s_1 + s_2 + \cdots + s_i$. Thus the maximum possible work done on $(J_{j_\ell})_{\ell=1}^{k}$ in this interval is $(T_i - T_{i+1})(s_1 + s_2 + \cdots + s_i)$. Summing over all i, we obtain $\sum_{i=1}^{k} s_i T_i$. In any valid schedule of $(J_{j_\ell})_{\ell=1}^{k}$ all the jobs are completed, so (2.5) follows.

Since the algorithm is online, the schedule for $\mathcal{J}_{[j_i]}$ is obtained from the schedule for $\mathcal{J}$ by removing the jobs J_j where $j > j_i$. At time T_i there are at least i jobs from $(J_{j_\ell})_{\ell=1}^{k}$ running, thus at least one job from $(J_{j_\ell})_{\ell=1}^{k-i+1}$ is running. So we have $T_i \leq C_{\max}^{\mathsf{A}}[\mathcal{J}_{[j_{k-i+1}]}]$ for any fixed random bits. Averaging over random bits of the algorithm and using (2.5), we have

$$\sum_{i=1}^{k} p_{j_i} \;\leq\; \mathbb{E}\left[\sum_{i=1}^{k} s_i C_{\max}^{\mathsf{A}}[\mathcal{J}_{[j_{k-i+1}]}]\right] \;=\; \sum_{i=1}^{k} s_i \mathbb{E}\left[C_{\max}^{\mathsf{A}}[\mathcal{J}_{[j_{k-i+1}]}]\right].$$

Since A is R-competitive, i.e., $\mathbb{E}[C_{\max}^{\mathsf{A}}[\mathcal{J}_{[j_{k-i+1}]}]] \leq R \cdot C_{\max}^{*,\Psi}[\mathcal{J}_{[j_{k-i+1}]}]$, . □

This lemma gives lower bound on the competitive ratio R. We will denote this bound by $r^{\Psi}(\mathcal{M})$. This bound is tight for many semi-online problems as we will see later.

Let r^{Ψ} be the largest lower bound on the approximation ratio obtained by Lemma 2.1.5:

Definition 2.1.6 *For any set of machines $\mathcal{M}$ and any partial input sequence $\mathcal{J} \in$* pref(Ψ)*,*

$$r^{\Psi}(\mathcal{M}, \mathcal{J}) = \sup_{1 \leq j_1 < j_2 < \cdots < j_k \leq n} \frac{\sum_{i=1}^{k} p_{j_i}}{\sum_{i=1}^{k} s_{k+1-i} \cdot C_{\max}^{*,\Psi}[\mathcal{M}, \mathcal{J}_{[j_i]}]}.$$

For any $\mathcal{M}$, let $\quad r^{\Psi}(\mathcal{M}) = \sup_{\mathcal{J} \in \mathrm{pref}(\Psi)} r^{\Psi}(\mathcal{M}, \mathcal{J}).$

Finally, let $\quad r^{\Psi} = \sup_{\mathcal{M}} r^{\Psi}(\mathcal{M}).$

2.2 The optimal algorithm

The author used doubling strategy to obtain 4-competitive deterministic and e-competitive randomized algorithm in his master thesis [ES04]. He and his advisor designed optimal algorithm for the whole class of preemptive online scheduling problems later. We present this algorithm here.

2.2.1 The idea

The basic idea of the algorithm is to use the slowest machine(s) for processing of a job. This saves fast machines for large jobs that may come in the future. Our algorithm first computes the competitive ratio $R(\mathcal{M}) \geq r(\mathcal{M})$, and later uses this value for taking decisions about the schedule.

In preemptive scheduling, we are allowed to use more machines for a schedule of one job (but not in parallel). Thus, we calculate the time where the job has to finish to maintain the competitive ratio. It involves computing the value of the optimal schedule. This value is multiplied by the previously computed competitive ratio R. By this way we will get time T, which is the deadline for a job to maintain the R-competitiveness of the algorithm.

Knowing the time interval $[0, T)$ that we are allowed to use, we use as slow machines as we can to manage that the job ends exactly at that time. But we have many possibilities how to do that. We want to be able to process as many large jobs as possible in the future.

Now we introduce notion of a *virtual machine* which is generalization of a virtual machine from the author's master thesis [ES04]. We index free space at each time. Let $V_k(t)$ be the k-th fastest free machine at time t. The function V_k represents k-th virtual machine. The idea is that if we have k large jobs, we cannot process them faster than scheduling them on first k virtual machines. Scheduling a job to virtual machine V_k at time t means exactly scheduling it to machine $V_k(t)$ at that time. Note that $V_k(t)$ may be undefined if there are less than k idle machines at time t. To schedule a job to V_k at time t where $V_k(t)$ is undefined means then to not schedule the job at time t to any machine.

We view the virtual machines as machines with speeds that change in time. Initially $V_i(t) = M_i$ for each t as there is no job scheduled. We also define $V_{m+1}(t)$ naturally as a function which is undefined for each t, this notation simplifies the description of the algorithm.

We want to preserve the fastest virtual machines as fast as possible (lexicographically, i.e., to optimize the speed of the fastest machine, then the second fastest, and so on), so our algorithm finds the slowest virtual machine that is able to process current job while maintaining competitive ratio. Let this machine be V_k. We will not schedule the job to any V_ℓ with $\ell < k$. If we consider any schedule of $k-1$ new jobs before scheduling the current job, then we can maintain completion times of these job even after scheduling our new job. The way we choose V_k maximizes the number $k-1$ for which this property holds.

Thus our algorithm has a pair V_k, V_{k+1} of adjacent virtual machines, such that only one of them can process current job in given time T. Now it saves a part of the faster virtual machine, by splitting the job between these two machines. Again we want to save as large part of the faster machine as possible, thus the job finishes exactly at given time being processed by either of these two virtual machines for each time $\tau < T$.

After scheduling of the job we have to update the virtual machines to maintain the definition, because knowing V_k explicitly seems to be better to implement. This is done simply by choosing the not used virtual machine from the pair $V_k(t), V_k + 1(t)$ to be new $V_k(t)$ for each $t \in [0, T)$. We have to shift slower virtual machines also: $V_\ell(t) \leftarrow V_{\ell+1}(t)$ for $\ell = k+1, \ldots, m$ and $t \in [0, T)$. ($V_{m+1}(t)$ remains undefined.)

Semi-online. Our algorithm remains nearly the same for semi-online or offline scheduling. Only two computations of parameters are replaced. First one, the value of the optimal makespan of jobs seen so far is replaced by computation (estimation) of minimal makespan that is achievable by some input sequence, that is consistent both with input seen so far and with the "semi-online" information provided to the algorithm, i.e., the number $C^{*,\Psi}_{\max}[\mathcal{J}_{[j]}]$ defined by Definition 2.1.4. Second computation is the estimation of the competitive ratio. Our algorithm achieves the best possible competitive ratio for any given set of machines provided that it is able to do these two computations correctly and precisely. We have also the option to provide imprecise bounds, as these may be simpler to compute. Namely, we can provide some upper bound R on the competitive ratio and we can provide a procedure that upper bounds optimal makespan. If this procedure returns bound that is always at most α times bigger than the actual optimal makespan, then we get the algorithm with competitive ratio of αR.

Offline. The basic "semi-online" information is knowledge of optimal makespan in advance. Then our algorithm has competitive ratio 1, thus it can be compared with offline algorithms. Its time complexity is asymptotically the same as the complexity of the best known algorithm and also it issues the smallest number of preemptions in the worst case. The algorithm for the offline problem is identical to the algorithm from the author's master thesis [ES04].

2.2.2 The algorithm

As we stated before, our algorithm first calls the procedure GetRatioValue to obtain parameter R, which is the competitive ratio maintained by the algorithm. Then upon arrival of a job J_j it calls GetOptValue to get the value of minimum makespan that optimum can achieve. This is simply the makespan of current optimal schedule for online scheduling. The value T_j is computed as R times the result of GetOptValue($\mathcal{J}_{[j]}$).

Then the algorithm finds two adjacent virtual machines V_k and V_{k+1}, and time t_j, such that if J_j is scheduled on V_{k+1} in the time interval $(0, t_j]$ and on V_k from t_j on, then J_j finishes exactly at time T_j. It is essential that each job is stretched over the whole interval $(0, T_j]$, which is the maximal time interval which it can use without violating the desired competitive ratio. Next we update the virtual machines, which means that in the interval $(0, T_j]$ we merge V_k and V_{k+1} into V_k and shift machines $V_{\ell+1}$, $\ell > k$, to V_ℓ. Then we continue with the next job. This gives a complete informal description of the algorithm sufficient for its implementation.

To prove that our algorithm works, it is sufficient to show that each job J_j

Algorithm 1 RatioStretch

```
1: procedure Initialize($\Psi; M_1, M_2, \ldots, M_m$)
2:     $R \leftarrow$ GetRatioValue($\Psi; M_1, M_2, \ldots, M_m$)
3:     for $i \in \{1, \ldots, m\}$,  $\tau \in (0, \infty)$ do $V_i(\tau) \leftarrow M_i$
4:     for $\tau \in (0, \infty)$ do $V_{m+1} \leftarrow \perp$          ▷ $V_{m+1}$ just helps the formulation.
5:     (for $i \in \{1, \ldots, m+1\}$ do $S_i \leftarrow \emptyset$          ▷ Defines $S_i$ for the proof.)
6: end procedure

7: procedure OnlineJob($J_j$)
8:     $T_j \leftarrow R \cdot$ GetOptValue($\Psi; M_1, M_2, \ldots, M_m; J_1, J_2, \ldots, J_j$)
9:     $k \leftarrow \min\{k \mid W_k(T_j) \geq p_j \geq W_{k+1}(T_j)\}$
                              ▷ If there is no $k$ then output "failed" and stop.
10:    find $t_j \in (0, T_j]$ such that $W_k(t_j) + W_k(T_j) - W_k(t_j) = p_j$

11:    for $\tau \in (0, t_j]$ do
12:        output $\sigma_j(\tau) \leftarrow V_{k+1}(\tau)$          ▷ Schedule the job to $V_{k+1}$ in $(0, t_j]$.
13:        for $i \in \{k+1, \ldots, m\}$ do $V_i(\tau) \leftarrow V_{i+1}(\tau)$
                              ▷ Update virtual machines on $(0, t_j]$.
14:    end for
15:    for $\tau \in (t_j, T_j]$ do
16:        output $\sigma_j(\tau) \leftarrow V_k(\tau)$          ▷ Schedule the job to $V_k$ in $(t_j, T_j]$.
17:        for $i \in \{k, \ldots, m\}$ do $V_i(\tau) \leftarrow V_{i+1}(\tau)$
                              ▷ Update virtual machines on $(t_j, T_j]$.
18:    end for
19:    for $\tau > T_j$ output $\sigma_j(\tau) \leftarrow \perp$
20:    ($S_k \leftarrow S_k \cup S_{k+1}$          ▷ Defines $S_i$ for the proof.)
21:    (for $i \in \{k+1, \ldots, m\}$ do $S_i \leftarrow S_{i+1}$          ▷ Defines $S_i$ for the proof.)
22: end procedure

23: Initialize($\Psi; M_1, M_2, \ldots, M_m$); $j \leftarrow 1$
24: while $j \leq n$ do OnlineJob($J_j$); $j \leftarrow j + 1$
25: end while
```

scheduled on V_1, completes by time T_j; this is equivalent to the fact that we can schedule J_j as described above. We show that this is true as long as GETRATIOVALUE returns a valid upper bound on competitive ratio.

To facilitate the proof, we maintain an assignment of scheduled jobs (and consequently busy machines at each time) to the set of virtual machines, i.e., for each virtual machine V_i we compute a set $\mathcal{S}_i$ of jobs assigned to V_i. Although the incoming job J_j is split between two different virtual machines, at the end of each iteration each scheduled job belongs to exactly one set $\mathcal{S}_i$, since right after J_j is scheduled the virtual machines executing this job are merged (during the execution of J_j). We stress that the sets $\mathcal{S}_i$ serve only as means of bookkeeping for the purpose of the proof, and their computation is not an integral part of the algorithm.

See Figure 2.1 for an example.

At each time τ, machine M_ℓ *belongs* to V_i if it is the ith fastest idle machine at time τ (i.e., $V_i(\tau) = M_\ell$), or if it is running a job $j \in \mathcal{S}_i$ at time τ. At each time τ the real machines belonging to V_i form a set of adjacent real machines, i.e., all machines $M_k, M_{k+1}, \ldots, M_\ell$ for some $k \leq \ell$. This relies on the fact that we always schedule a job on two adjacent virtual machines which are then merged into a single virtual machine during the times when the job is running, and on the fact that these time intervals $(0, T_j]$ increase with j, as adding new jobs cannot decrease the optimal makespan (i.e., $T_j = RC^{*,\Psi}_{\max}[\mathcal{J}_{[j]}] \leq RC^{*,\Psi}_{\max}[\mathcal{J}_{[j+1]}] = T_{j+1}$).

Let $v_i(t)$ denote the speed of the virtual machine V_i at time t, i.e. $v_i(t) = s(V_i(t))$. We define $v_i(t) = s(\perp) = 0$ if V_i is undefined at time t. This corresponds to the fact that a job scheduled to V_i at time t will be processed by speed $v_i(t)$.

Let $W_i(t) = \int_0^t v_i(\tau) d\tau$ be the total work which can be done on machine V_i in the time interval $(0, t]$. By definition we have $v_i(t) \geq v_{i+1}(t)$ and thus also $W_i(t) \geq W_{i+1}(t)$ for all i and t. Note also that $W_{m+1}(t) = v_{m+1}(t) = 0$ for all t.

Theorem 2.2.1 *Suppose the procedure* GETOPTVALUE *returns always a number in* $[C^{*,\Psi}_{\max}, \alpha C^{*,\Psi}_{\max}]$ *and procedure* GETRATIOVALUE *returns an upper bound on competitive ratio* $R \geq r^{\Psi}(\mathcal{M})$. *Then the algorithm* RATIOSTRETCH *never outputs "failed" and is* $\bar{R} = \alpha R$ *competitive for online preemptive scheduling on* m *uniformly related machines with speeds* $s_1 \geq s_2 \geq \cdots \geq s_m$.

We have to ensure, that outputs of GETOPTVALUE are nondecreasing, but this can be done simply by returning the maximum of results of calls to GETOPTVALUE so far. As optima are nondecreasing in time, this cannot break the condition that result of GETOPTVALUE is in $[C^{*,\Psi}_{\max}, \alpha C^{*,\Psi}_{\max}]$.

Proof: If RATIOSTRETCH schedules a job, it is always completed at time $T_j \leq \bar{R} \cdot C^{*,\Psi}_{\max}[(J_i)_{i=1}^n]$. Thus to prove the theorem, it is sufficient to guarantee that the algorithm does not fail to find the number k on line 9. This is equivalent to the statement that there is always enough space on V_1, i.e., that $p_j \leq W_1(T_j)$ in the iteration when j is to be scheduled. Since $W_{m+1} = 0$ by the definition, this is sufficient to guarantee that required k exists. Given the choice of k, it is always possible to find time t_j (on the next line of the algorithm) as the expression

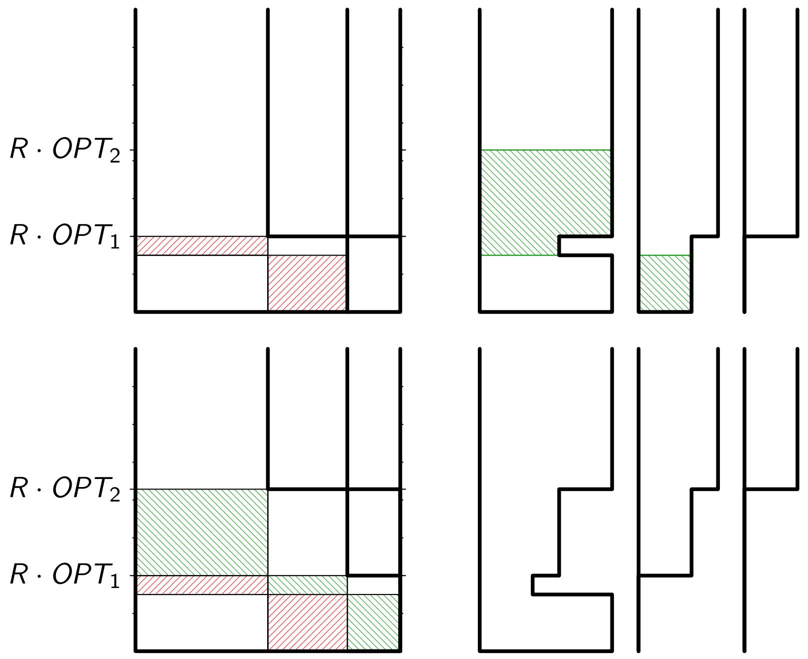


Figure 2.1: An illustration of a schedule of two jobs on three machines produced by RatioStretch. Vertical axis denotes the time, horizontal axis corresponds to the speed of the machines. The pictures on the left depict the schedule on the real machines, with bold lines separating the virtual machines. The pictures on the right show only the idle time on the virtual machines. The top pictures show the situation after the first job, with the second job being scheduled on the first two virtual machines. The bottom pictures show the situation with after the second job is scheduled and virtual machines updated.

$W_{k+1}(t_j)+W_k(T_j)-W_k(t_j)$ is continuous in t_j, for $t_j = 0$ it is equal to $W_k(T_j) \geq p_j$, and for $t_j = T_j$ it is equal to $W_{k+1}(T_j) \leq p_j$.

Consider now all the jobs that are scheduled on the first virtual machine, i.e., the set $\mathcal{S}_1$. Let $\bar{J}_1, \bar{J}_2, \ldots, \bar{J}_{n'-1}$ denote the jobs in $\mathcal{S}_1$, ordered as they appear on input. Let $\bar{J}_{n'} = J$ be the incoming job. Let T_j be the time computed on line 8 when a job $\bar{J}_j$ is processed.

Consider any $j' = 1, \ldots, n'$ and any time $\tau \in (0, T_{j'}]$. Using the fact that the times T_j are non-decreasing in j and that the algorithm stretches each job j over the whole interval $(0, T_j]$, there are at least $n' - j'$ jobs from $\mathcal{S}_1$ running at τ, namely jobs $\bar{J}_{j'}, \bar{J}_{j'+1}, \ldots, \bar{J}_{n'-1}$. Including the idle machine, there are at least $m + 1 - j'$ real machines belonging to V_1. Since V_1 is the first virtual machine and the real machines are adjacent, they must include the fastest real machines $M_1, \ldots, M_{m+1-j'}$. It follows that the total work that can be processed on the real machines belonging to V_1 during the interval $(0, T_{j_m}]$ is at least $s_1 T_{n'} + s_2 T_{n'-1} + \cdots + s_{n'} T_1$. The total processing time of jobs in $\mathcal{S}_1$ is $\bar{p}_1 + \bar{p}_2 + \cdots + \bar{p}_{n'-1}$. Thus to prove that

$\bar{J}_{n'}$ can be scheduled on V_1 we need to verify that

$$\bar{p}_{n'} \leq s_1 T_{n'} + s_2 T_{n'-1} + \cdots + s_{n'} T_1 - (\bar{p}_1 + \bar{p}_2 + \cdots + \bar{p}_{n'-1}). \tag{2.6}$$

From the assumption of the correctness of GetRatioValue, we can use Lemma 2.1.5 (the first inequality), and then (in second inequality) we use the correctness of GetOptValue:

$$(\bar{p}_1 + \bar{p}_2 + \cdots + \bar{p}_{n'}) \ \leq \ R \sum_{i=1}^{n'} s_i C^{*,\Psi}_{\max}[\bar{\mathcal{J}}[n'-i+1]] \ \leq \ \sum_{i=1}^{n'} s_i T_{n'-i+1} \ . \qquad \square$$

Corollary 2.2.2 *With procedures that return the exact values for $C^{*,\Psi}_{\max}$ and $r^{\Psi}(\mathcal{M})$ is our algorithm optimal (semi-)online algorithm with competitive ratio $r^{\Psi}(\mathcal{M})$.*

Proof: The corollary is direct consequence of Theorem 2.2.1 and Definition 2.1.6. $\square$

Note that our algorithm is deterministic. (We assume both procedures to be deterministic here.) The lower bound $r^{\Psi}(\mathcal{M})$ holds also for randomized algorithms. Thus randomization does not help here in preemptive (semi-)online scheduling.

2.2.3 A brief description of the procedures

Procedure GetOptValue This procedure can be easily implemented for the online case. The value of optimal makespan is computed by formula from Fact 2.1.1:

$$C^*_{\max}[\mathcal{J}_j] = \max\left\{ \frac{P_1}{S_1}, \frac{P_2}{S_2}, \cdots, \frac{P_{m-1}}{S_{m-1}}, \frac{P}{S_m} \right\} ,$$

where $S_i = s_1 + s_2 + \cdots + s_i$ is the sum of the speeds of i fastest machines and P_i is the sum of processing times of i largest jobs in $\mathcal{J}_j$. Thus it can be computed very fast.

There is no general formula to compute $C^{*,\Psi}_{\max}$ for any Ψ. It is also possible, that $C^{*,\Psi}_{\max}[\mathcal{J}]$ cannot be even approximated for some weird restriction Ψ and for some sequence $\mathcal{J} \in \mathrm{pref}(\Psi) \setminus \Psi$ as we can encode some hard problem into the conditions defining the restriction Ψ. The studied restrictions are much nicer. The formula is typically as simple as the formula for the online case.

Procedure GetRatioValue This procedure returns optimal competitive ratio for given speeds $s_1, \ldots, s_m$. This is typically done by solving one or several linear programs with $O(m^2)$ variables (see next section). Construction of these programs is possible because of simplicity of formulas for computing the optimal makespan. We can solve any linear program to arbitrary precision in polynomial time, thus the algorithm may be efficient as well. Alternatively we may provide some valid upper bound R on competitive ratio. Then the algorithm will be R-competitive. E.g., we know that there is e-competitive randomized algorithm for the online case [ES04], thus this procedure can simply return e here to obtain e-competitive deterministic online algorithm.

2.3 The optimal competitive ratio and the bounds

In many cases we will be able to construct the exact procedures GetOptValue and GetRatioValue, thus Corollary 2.2.2 will apply. In this section we will show the construction of GetRatioValue together with proofs of correctness (and effectiveness) for the online scheduling. Then we will continue with the list of various variants of semi-online scheduling, but we will not show the proofs there, as they will be nearly the same as in the online variant. The procedure GetOptValue is typically trivial, more precisely, it is only taking of maximum of several values, as in Fact 2.1.1.

The idea will be the same for all cases: we construct a mathematical (linear) program that finds a worst case sequence, that is sequence which is witness for competitive ratio from Definition 2.1.6. This cannot be done in general case, but it turns out that all studied semi-online cases are nice, and allow to be solved by this approach. As we already mentioned we do not show the proofs of correctness here. We only show the correspondence between the input sequence and the (feasible) solution to the respective linear program. This information suffices to repeat the proof from the online variant in listed semi-online cases.

2.3.1 The online model

For each input sequence $\mathcal{J} = (J_j)_{j=1}^n$, Lemma 2.1.2 shows that the competitive ratio is at least $\sum_{j=1}^n p_j / (\sum_{k=1}^m s_{m-k+1} C^*_{\max}[\mathcal{J}_{[n-m+k]}])$. We are searching a sequence giving the best bound. The set of input sequences can be restricted in two ways. Firstly, we may assume that the processing times of jobs are non-decreasing: Sorting the jobs can only decrease the values of $C^*_{\max}[\mathcal{J}_{[j]}]$, as in each of these partial instances some jobs are possibly replaced by smaller jobs; thus the bound on R can only increase. Secondly, if there are more than $m-1$ jobs, then we can replace first $n-m+1$ jobs with m jobs of size $\sum_{j=1}^{n-m+1} p_j/m$. All $C^*_{\max}\mathcal{J}_{[n-m+k]}$ are then either determined by (2.1) or by terms of type (2.2), where no job from the first m new jobs occurs. These bounds are same as in the original sequence, thus our new sequence also gives no worse bound than the original one.

We can also suppose that there are some jobs of zero size at the beginning of the sequence, to not consider the case of $n < m$ separately, but this will not be the case for semi-online variants. This is because prepending zero-sized jobs to any sequence does not change the value of the bound according Lemma 2.1.2. On the other hand the value of the bound from Definition 2.3.7 will typically change by prepending zero-sized jobs.

Now it is easy to give a mathematical program to compute the optimal lower bound, given the parameters $s_1 \geq \cdots \geq s_m$. The mathematical program has variables $q_1, q_2, \ldots, q_m, O_1, O_2, \ldots, O_m$. Variable q_1 corresponds to the sum of all processing times in $\mathcal{J}_{[n-m+1]}$, variables $q_2, \ldots, q_m$ to the processing times of the last $m-1$ jobs, and variables O_k correspond to $C^*_{\max}[\mathcal{J}_{[n-m+k]}]$.

$$\begin{array}{rcll}
\textbf{maximize} & \multicolumn{2}{l}{\dfrac{q_1+q_2+q_3+\cdots+q_m}{s_1O_m+s_2O_{m-1}+\cdots+s_mO_1}} & \\
\textbf{subject to} & & & \\
q_1+q_2+\cdots+q_k & \le & (s_1+s_2+\cdots+s_m)O_k & \text{for } k=1,\ldots,m \\
q_j+q_{j+1}+\cdots+q_k & \le & (s_1+s_2+\cdots+s_{k-j+1})O_k & \text{for } 2\le j\le k\le m \\
q_j & \le & q_{j+1} & \text{for } j=2,\ldots,m-1 \\
0 & \le & q_1 & \\
0 & \le & q_2 &
\end{array}$$

We can see that this program is homogeneous, i.e., multiplying all values of variables by arbitrary factor has no influence on feasibility nor optimality of the solution. This reflects the fact that the bound is invariant under scaling, so we may assume that $\sum_{i=1}^{m} s_i C^*_{\max}[\mathcal{J}_{[n-i+1]}] = 1$. Thus we add this as a constraint to make the program linear. (We call this constraint *normalization.*)

Definition 2.3.1 *Let $R(s_1,s_2,\ldots,s_m)$ denote the value of the objective function of the optimal solution of the following linear program:*

$$\begin{array}{rcll}
\textbf{maximize} & \multicolumn{2}{l}{R(s_1,s_2,\ldots,s_m)=q_1+q_2+q_3+\cdots+q_m} & \\
\textbf{subject to} & & & \\
q_1+q_2+\cdots+q_k & \le & (s_1+s_2+\cdots+s_m)O_k & \textit{for } k=1,\ldots,m \\
q_j+q_{j+1}+\cdots+q_k & \le & (s_1+s_2+\cdots+s_{k-j+1})O_k & \textit{for } 2\le j\le k\le m \\
1 & = & s_1O_m+s_2O_{m-1}+\cdots+s_mO_1 & \\
q_j & \le & q_{j+1} & \textit{for } j=2,\ldots,m-1 \\
0 & \le & q_1 & \\
0 & \le & q_2 &
\end{array} \tag{2.7}$$

The linear program has a feasible solution with the only non-zero variable $O_m = 1/s_1$. It is also easy to see that the objective function is bounded, the constraints imply that $q_1+q_2+\cdots+q_m \le (s_1+s_2+\cdots+s_m)O_m \le m\cdot s_1O_m \le m$. Thus the value $R(s_1,\ldots,s_m)$ is well-defined. Finally, note that the linear program has a quadratic number of constraints, and thus it can be solved efficiently.

Observation 2.3.2 *Any feasible solution of (2.7) induces input sequence which lower bounds $r(\mathcal{M})$ by $q_1+q_2+\cdots+q_m$.*

This induced sequence is: $p_1 = p_2 = \cdots = p_m = q_1/m$, $p_{m+1} = q_2$, $p_{m+2} = q_3$, $\ldots$, $p_{2m-1} = q_m$. We can see that $O_k \ge C^*_{\max}[\mathcal{J}_{[n-m+k]}]$ thus we have $\sum_{k=1}^{m} s_{m-k+1}C^*_{\max}[\mathcal{J}_{[n-m+k]}] \le 1$. Then the observation follows directly from Definition 2.1.6 of $r(\mathcal{M})$.

Lemma 2.3.3 *For any $R \ge R(s_1,s_2,\ldots,s_m)$ and for any input sequence $\mathcal{J}$ we have*

$$\sum_{j=1}^{n} p_j \;\le\; R\cdot\sum_{i=1}^{m} s_i C^*_{\max}[\mathcal{J}_{[n-i+1]}].$$

Proof: Assume, that we have some input sequence $\mathcal{J}'$ and R' for which the lemma does not hold. Then the input sequence also contradicts the lemma with $R = R(s_1, s_2, \ldots, s_m)$. Now consider the sequence $\mathcal{J}''$, which is $\mathcal{J}'$ sorted by increasing processing time. This sequence also contradicts the lemma as $C^*_{\max}[\mathcal{J}'_{[n-i+1]}] \leq C^*_{\max}[\mathcal{J}''_{[n-i+1]}]$ for each i. The on-line problem scales, so we modify $\mathcal{J}''$ to get the sequence $\mathcal{J}$ just by setting $p_j = p''_j / \sum_{i=1}^m s_i C^*_{\max}[\mathcal{J}''_{n-i+1}]$. The sequence $\mathcal{J}$ also contradicts the lemma as both sides of the inequality are scaled by same factor.

Now we can set $q_j = p_{m-n+j}$ for $j = 2 \ldots n$ and $q_1 = \sum_{j=1}^{m-n+1} p_j$. Also we set $O_k = C^*_{\max}[\mathcal{J}_{[m-k+1]}]$. But this is feasible solution to (2.7), we have $R \geq \sum_{j=1}^m q_j = \sum_{j=1}^n p_n$ as well as $1 = \sum_{i=1}^m s_i O_{m-1+1} = \sum_{i=1}^m s_i C^*_{\max}[\mathcal{J}_{[n-i+1]}]$, thus the lemma holds for R and $\mathcal{J}$, that is the contradiction. □

Theorem 2.3.4 $r(\mathcal{M}) = R(s_1, s_2, \ldots, s_m)$*, thus we can implement* GetRatioValue *with arbitrary precision for online scheduling.*

Proof: One inequality follows from Observation 2.3.2, the other from Lemma 2.3.3. □

By Corollary 2.2.2 we know, that our algorithm is optimal with respect to competitive ratio for this model. Moreover no randomized algorithm can achieve better competitive ratio. We combine this with the knowledge of e-competitive randomized algorithm [ES04], to obtain upper bound on competitive ratio of our algorithm (and the problem).

Numerical lower bound

We have the optimal algorithm for arbitrary speeds, but we do not know the numerical value of its competitive ratio. The competitive ratio for m machines is equal to the solution of a quadratic program obtained from (2.7) by considering s_i to be variables (in addition to q_j and O_k). However, this quadratic program is not convex and we do not know how to solve it. We have obtained some lower bounds numerically using mathematical software Maple.

A lower bound is simply a feasible solution of (2.7). Once the values of s_i are given, verification only involves solving a linear program. Once also the optimal values q^*_j are given, it is trivial to compute values O^*_k and verify the constraints of (2.7). A file with our solutions for $m = 3, \ldots, 200$ in a format suitable for computer verification is attached to the electronic version of this thesis.

We obtain lower bounds improving the bounds from [ES00] for $m \geq 8$. For $m \geq 58$ the bounds are above 2, and for $m = 200$ we obtain a speed combination with optimal competitive ratio above 2.112. The numerical values of the sequence on $m = 200$ machines can be found in Appendix A. We show a graphical representation of the set of speeds s_i and of the sequence of values q_j that witness a lower bound on competitive ratio of 2.054 on $m = 100$ machines in Figure 2.2. The structure of the solution is very similar to the solution for $m = 2.112$.

Thus we have:

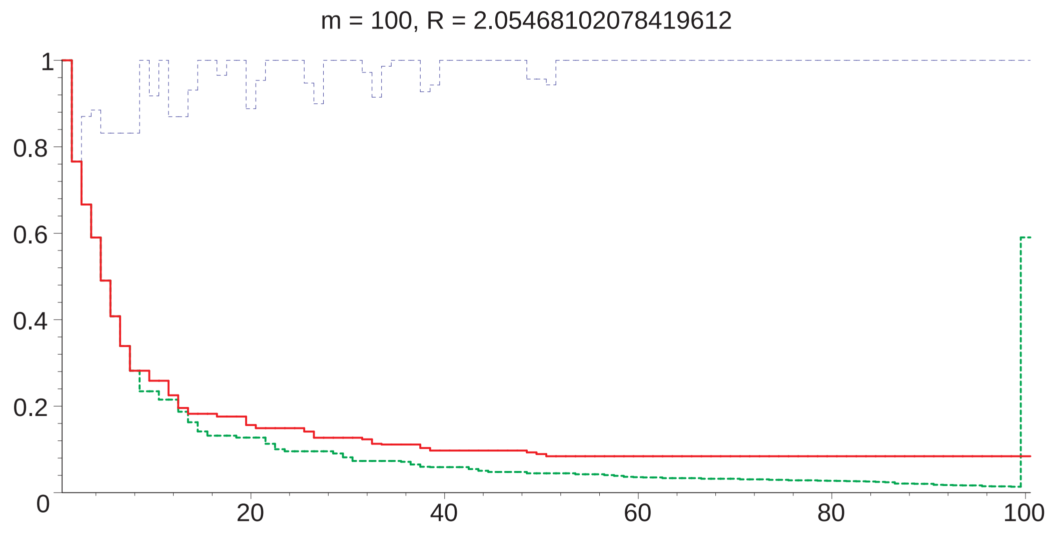


Figure 2.2: The instance for $m = 100$. The red bold curve shows speeds s_i, green dashed bold curve (bottom one) shows scaled job variables, and the thin dashed blue curve (top one) shows inverses of the speed ratios, i.e., s_{i-1}/s_i. The values of q_j are printed in reverse order, i.e., column i shows q_{m-i+1}. The values are scaled so that $q_m = 1 = s_1$. This ordering and scaling shows best the relations between job sizes and speeds.

Theorem 2.3.5 *For any online algorithm for preemptive scheduling on uniformly related machines, the competitive ratio is at least* 2.112*.*

2.3.2 The semi-online models in general

Many semi-online models can be solved similarly to the online model. We just need to construct both procedures and proof their correctness. The procedure GetOptValue will be typically maximum of few values, and the procedure GetRatioValue will typically solve some linear program.

We observed that for a general restriction it may be necessary to use an arbitrary subsequence in Definition 2.1.6. However, for many restrictions it is sufficient to use the whole sequence, similarly as for online scheduling. Usual restrictions are essentially of two kinds. The first type are the restrictions that put conditions on individual jobs or their order. These restrictions are closed under taking subsequences (not only prefixes), i.e., any subsequence of an input sequence is also in Ψ. The second type are the restrictions where some global information is given in advance, like $\sum p_j = P$ or $C^*_{\max} = T$. These are not closed under taking subsequences, but are closed under permuting the input sequence.

We define a large class of restrictions that includes both types of restrictions discussed above as well as their combinations; in particular it includes all the restrictions listed and studied here. The definition below implies that any subsequence of any input sequence is a prefix of another input. Thus, the sets of all the subsequences and all the prefixes of Ψ coincide, and Definition 2.1.6 simplifies using the monotonicity condition in the definition.

Definition 2.3.6 *An input restriction* Ψ *is* proper *if for any* $\mathcal{J} \in \Psi$ *and any subsequence* $\mathcal{I}$ *of* $\mathcal{J}$*, we have* $\mathcal{I} \in \mathrm{pref}(\Psi)$ *and furthermore* $C^{*,\Psi}_{\max}[\mathcal{I}] \leq C^{*,\Psi}_{\max}[\mathcal{J}]$*.*

Definition 2.3.7 *Let* Ψ *be a proper semi-online restriction and* $\mathcal{J} \in \mathrm{pref}(\Psi)$ *a partial input. We define*

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

From now on, we focus on proper restrictions. Consider arbitrary input sequence $\mathcal{J}$. We have $C^{*,\Psi}_{\max}[\mathcal{I}] \leq C^{*,\Psi}_{\max}[\mathcal{J}]$ for every subsequence $\mathcal{I}$ of the sequence $\mathcal{J}$. Moreover we know that $\mathcal{I} \in \Psi$. So far we proved that $r^{\Psi}(\mathcal{M}, \mathcal{J}) \geq \bar{r}^{\Psi}(\mathcal{M}, \mathcal{J})$. If we take a closer look on the definition of approximation ratio for general restriction (Definition 2.1.6), we can find the witness subsequence of jobs $\mathcal{I} = (p_{j_i})_{i=1}^{k}$ for any precision $\epsilon > 0$:

$$r^{\Psi}(\mathcal{M}, \mathcal{J}) - \epsilon \leq \frac{\sum_{i=1}^{k} p_{j_i}}{\sum_{i=1}^{k} s_{k+1-i} C^{*,\Psi}_{\max}[\mathcal{M}, \mathcal{J}_{[j_i]}]} .$$

We use $C^{*,\Psi}_{\max}[\mathcal{M}, \mathcal{I}_{[i]}] \leq C^{*,\Psi}_{\max}[\mathcal{M}, \mathcal{J}_{[j_i]}]$ from Definition 2.3.6 to obtain that $\mathcal{I}$ ensures $\bar{r}^{\Psi}(\mathcal{M}, \mathcal{I}) \geq r^{\Psi}(\mathcal{M}, \mathcal{J}) - \epsilon$.

So we can proceed to simpler formula for the optimal approximation ratio:

Observation 2.3.8 *For any proper restriction* Ψ*,*

$$r^{\Psi}(\mathcal{M}) = \sup_{\mathcal{J} \in \mathrm{pref}(\Psi)} \bar{r}^{\Psi}(\mathcal{M}, \mathcal{J}) .$$

Our strategy is to reduce the number of sequences $\mathcal{J}$ that need to be taken into account. Typically, we show that we can restrict ourselves only to sorted sequences without losing the witness of the competitive ratio. Then we know which jobs are the biggest and we can express the optimal makespans for prefixes by linear constraints in job sizes. Maximizing the expression for $\bar{r}^{\Psi}(\mathcal{M})$, which gives the desired bound, is then reduced to solving one or several linear programs with quadratic number of constraints. (The property that sorted sequences are the worst is not crucial for the algorithm. We might be able to effectively separate a constraint if there was exponential number of them.)

The following observations help us to limit the set of relevant sequences.

Observation 2.3.9 *Let* Ψ *be arbitrary proper restriction, let* $\mathcal{M}$ *be arbitrary set of machines, and let* $\mathcal{J}, \mathcal{J}' \in \mathrm{pref}(\Psi)$ *be two partial inputs with* n *jobs. Suppose that for some* $b > 0$*:*

$$\begin{aligned} \sum_{j=1}^{n} p_j &\leq b \sum_{j=1}^{n} p'_j \\ (\forall i = 1, \ldots, n) \quad C^{*,\Psi}_{\max}[\mathcal{J}_{[i]}] &\geq b\, C^{*,\Psi}_{\max}[\mathcal{J}'_{[i]}]. \end{aligned}$$

Then $\bar{r}^{\Psi}(\mathcal{M}, \mathcal{J}') \geq \bar{r}^{\Psi}(\mathcal{M}, \mathcal{J})$*.*

Proof: The observation follows immediately from the definition of $\bar{r}^{\Psi}(\mathcal{M}, \mathcal{J})$. $\square$

Observation 2.3.10 *Assume that (i) Ψ is closed under permutations of the sequence and (ii) increasing the size of the last job of a partial input cannot decrease $C^{*,\Psi}_{\max}$, i.e., for any partial inputs $\mathcal{J}, \mathcal{J}' \in \mathrm{pref}(\Psi)$ with n jobs such that $p_1 = p'_1$, $\dots$, $p_{n-1} = p'_{n-1}$, and $p_n \le p'_n$ we have $C^{*,\Psi}_{\max}[\mathcal{J}] \le C^{*,\Psi}_{\max}[\mathcal{J}']$. Then it is sufficient to consider sequences of non-decreasing jobs, i.e.,*

$$r^{\Psi}(\mathcal{M}) = \sup_{\mathcal{J} \in \mathrm{pref}(\Psi)\ :\ p_1 \le p_2 \le \dots \le p_n} \bar{r}^{\Psi}(\mathcal{M}, \mathcal{J})$$

Proof: First note that if Ψ is closed under permutations, then Ψ is a proper restriction. Whenever $\mathcal{J}$ contains two jobs with $p_k > p_{k+1}$, swapping them can only decrease $C^{*,\Psi}_{\max}[\mathcal{J}_{[k]}]$ and any other $C^{*,\Psi}_{\max}[\mathcal{J}_{[i]}]$ remains unchanged. Using Observation 2.3.9 with $b = 1$, it follows that swapping the two jobs can only increase the lower bound. Since any sequence can be sorted by swapping adjacent elements, it follows the supremum over non-decreasing inputs in Ψ is equal to the supremum over all Ψ. $\square$

2.3.3 Known sum of processing times, $\sum p_j = P$

We are given a value $\bar{P}$ and Ψ contains all $\mathcal{J}$ with $P = \bar{P}$ in this variant of semi-online scheduling. It can be easily verified that $\mathrm{pref}(\Psi)$ contains exactly all sequences $\mathcal{J}$ with $P \le \bar{P}$ and $C^{*,\Psi}_{\max}[\mathcal{J}] = \max\{C^{*}_{\max}[\mathcal{J}], \bar{P}/S\}$ for any $\mathcal{J} \in \mathrm{pref}(\Psi)$.

Since we can permute the jobs and increasing the size of the last job does not decrease $C^{*,\Psi}_{\max}$, Observation 2.3.10 implies that to compute $r^{\Psi}(\mathcal{M}) = r^{\sum p_j = P}(\mathcal{M})$, we can restrict ourselves to non-decreasing sequences $\mathcal{J}$. Furthermore, we observe that we can restrict ourselves to sequences $\mathcal{J}$ with less than m jobs. If $n \ge m$, we use the fact that $C^{*,\Psi}_{\max}[\mathcal{J}_{[i]}] \ge \bar{P}/S$ for any i and obtain

$$\bar{r}^{\Psi}(\mathcal{M}, \mathcal{J}) = \frac{P}{\sum_{i=1}^{n} s_{n+1-i} \cdot C^{*,\Psi}_{\max}[\mathcal{J}_{[i]}]} \le \frac{P}{\sum_{i=1}^{n} s_{n+1-i} \frac{\bar{P}}{S}} = \frac{P}{\bar{P}} \le 1,$$

using $n \ge m$ and $P \le \bar{P}$ in the last step. Thus sequences with $n \ge m$ cannot lead to any non-trivial bound.

Finally, we may restrict ourselves to $\mathcal{J}$ with $P = \bar{P}$: If $P < \bar{P}$, we scale up $\mathcal{J}$ to $\mathcal{J}'$ by multiplying all the processing times by $b = \bar{P}/P$. Observation 2.3.9 then applies, as each $C^{*,\Psi}_{\max}[\mathcal{J}'_{[i]}] = \max\{C^{*}_{\max}[\mathcal{J}'_{[i]}], \bar{P}/S\}$ increases by at most the scaling factor b.

Summarizing, we can assume that $\mathcal{J}$ is a non-decreasing sequence of $n < m$ jobs (w.r.t. their processing times) with $P = \bar{P}$. (Note that this does not mean that the adversary uses fewer jobs than machines, as he may need to release some small jobs at the end of the prefix sequence, to extend it to a sequence in Ψ.) To obtain $r(\mathcal{M})$, we compute $m - 1$ mathematical programs, one for each value of n, and take the maximum of the optimal values of their objective functions.

We already mentioned, that we cannot use single linear program as in the online variant. We can easily see that $C^{*,\sum p_j = P}_{\max}[\mathcal{J}^0] = \bar{P}/S > 0$, for any $\mathcal{J}^0$ containing only zero sized jobs. Thus prepending zero sized jobs will destroy the bound from definition 2.3.7. This is even amplificated here so that any sequence of at least m jobs gives no nontrivial bound.

The program for given P, $s_1, s_2, \ldots, s_m$, and n has variables q_i for job sizes and O_i for optimal makespans of the prefixes:

$$\begin{array}{rcll}
\textbf{maximize} & & R_n(s_1, s_2, \ldots, s_m) = \dfrac{\bar{P}}{s_1O_n + s_2O_{n-1} + \cdots + s_nO_1} & \\
\textbf{subject to} & & & \\
q_1 + q_2 + \cdots + q_n & = & \bar{P} & \\
\bar{P} & \leq & SO_k & \text{for } k = 1, \ldots, n \\
q_j + q_{j+1} + \cdots + q_k & \leq & (s_1 + s_2 + \cdots + s_{k-j+1})O_k & \text{for } 1 \leq j \leq k \leq n \\
q_j & \leq & q_{j+1} & \text{for } j = 1, \cdots, n-1 \\
0 & \leq & q_1 &
\end{array} \tag{2.8}$$

If we fix the input sequence, i.e., the values of q_i, then the smallest objective is achieved for O_k as small as possible which is exactly the value of the optimal makespan, by the constraints involving O_k. We can use the same technique as in the online case to prove that the mathematical program computes correctly the value $\bar{r}^{\sum p_j = P}(\mathcal{M}, \mathcal{J})$ for any $n = |\mathcal{J}| < m$. (The techinque to prove Lemma 2.3.3 and Observation 2.3.2).

We can also see that the linear program scales and the optimum does not depend on the value $\bar{P}$. Thus we can normalize using $1 = s_1O_n + s_2O_{n-1} + \cdots + s_nO_1$ and also eliminate $\bar{P}$ by combining the first two constraints. The resulting linear program is:

$$\begin{array}{rcll}
\textbf{maximize} & & R(s_1, s_2, \ldots, s_m) = q_1 + \cdots + q_n & \\
\textbf{subject to} & & & \\
1 & = & s_1O_n + s_2O_{n-1} + \cdots + s_nO_1 & \\
q_1 + q_2 + \cdots + q_n & \leq & SO_k & \text{for } k = 1, \ldots, n \\
q_j + q_{j+1} + \cdots + q_k & \leq & (s_1 + s_2 + \cdots + s_{k-j+1})O_k & \text{for } 1 \leq j \leq k \leq n \\
q_j & \leq & q_{j+1} & \text{for } j = 1, \cdots, n-1 \\
0 & \leq & q_1 &
\end{array} \tag{2.9}$$

From the analysis above it follows that for any $\mathcal{M}$, the maximum of the $m-1$ optima of the linear programs is the correct value of $r^{\sum p_j = P}(\mathcal{M})$.

Padding

We prove a theorem that shows that knowing the total size of jobs does not improve the overall approximation ratio. 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. The same result holds also in presence of an additional restriction with suitable properties. Among the restrictions that we consider, the requirements are satisfied for non-increasing jobs, known maximal job size, or the online case. By "$\Psi, \sum p_j = P$" we denote the intersection of the two restrictions, i.e., the set of all sequences $(p_j)_{j=1}^n \in \Psi$ such that $\sum_{i=1}^n p_j = \bar{P}$ for a given value of $\bar{P}$

We say that Ψ allows scaling if for any $\mathcal{J} \in \Psi$ and $b > 0$, the modified sequence $\mathcal{J}' = (bp_j)_{j=1}^n$ satisfies $\mathcal{J}' \in \Psi$. We say that Ψ allows padding if for any $\mathcal{J} \in \Psi$, and for any $\epsilon > 0$ and $P^+ > 0$ there exists a sequence $\mathcal{J}' \in \Psi$ satisfying $\sum_{J\in\mathcal{J}} p(J) + P^+ = \sum_{J\in\mathcal{J}'} p(J)$ created by extending $\mathcal{J}$ by jobs of size at most ϵ at the end (i.e., $p'_1 = p_1, \ldots, p'_n = p_n, p'_{n+1} \leq \epsilon, \ldots, p'_{n'} \leq \epsilon$). That means we can append some arbitrarily tiny jobs to $\mathcal{J}$ to increase the sum of processing times by P^+.

Theorem 2.3.11 *Suppose that Ψ is proper, allows scaling, padding, and is closed under taking prefixes. Let $\mathcal{J} \in \Psi$ and let $\mathcal{M}$ be arbitrary. Then for any $\delta > 0$ there exists $\mathcal{J}'$ and $\mathcal{M}'$ such that*

$$\bar{r}^{\Psi,\sum p_j = P}(\mathcal{M}', \mathcal{J}') \geq \bar{r}^{\Psi}(\mathcal{M}, \mathcal{J})/(1+\delta).$$

Consequently, $r^{\Psi,\sum p_j=P} = r^{\Psi}$.

Proof: We fix $\mathcal{M}$, $\mathcal{J}$, and $\bar{P}$ that lower bounds $\bar{r}^{\Psi}$, that is a bad input for algorithm operating in the restriction Ψ. We proceed towards constructing the appropriate $\mathcal{M}'$ and $\mathcal{J}'$ that proves nearly same lower bound on $\bar{r}^{\Psi,\sum p_j=P}$.

The intuitive explanation is this. First, we scale $\mathcal{J}$ to $\mathcal{J}'$ with the required total processing time. This guarantees that $\bar{r}^{\Psi,\sum p_j=P}(\mathcal{M}, \mathcal{J}')$ is defined, but possibly it is much smaller than $\bar{r}^{\Psi}(\mathcal{M}, \mathcal{J}')$. Next we modify $\mathcal{M}$ to $\mathcal{M}'$ by adding a huge number of slow machines. The condition that Ψ allows padding guarantees that the values of optima in $\bar{r}^{\Psi,\sum p_j=P}(\mathcal{M}', \mathcal{J}')$ can be witnessed by extensions of the partial inputs by arbitrarily small jobs. The capacity of the new machines is chosen to be sufficiently large, so that they can accommodate all these small jobs, thus guaranteeing that the additional restriction $\sum p_j = P$ has no significant influence on the value of the optima. At the same time, the new machines are chosen to be sufficiently slow so that the optimal bounds do not change significantly by adding the new machines.

Since Ψ allows scaling, the value $C_{\max}^{*,\Psi}[\mathcal{J}]$ is multiplied by b when $\mathcal{J}$ is scaled by b. Consequently, the value of $\bar{r}^{\Psi}(\mathcal{M}, \mathcal{J})$ does not change when $\mathcal{J}$ is scaled. Let $\mathcal{J}'$ be the sequence $\mathcal{J}$ scaled so that $\sum_{j=1}^n p'_j = \bar{P}$. Then $\bar{r}^{\Psi}(\mathcal{M}, \mathcal{J}') = \bar{r}^{\Psi}(\mathcal{M}, \mathcal{J})$. Observation 2.3.9 applies both ways here, proving the equality.

Let $O_i = C_{\max}^{*,\Psi}[\mathcal{J}'_{[i]}]$, i.e., the optimal makespan of the ith prefix of $\mathcal{J}'$. Choose a small $\sigma > 0$ so that $\sigma < \delta s_1/n$. Let $\mathcal{M}'$ be the set of machines $\mathcal{M}$ extended by $\lceil \bar{P}/(O_1\sigma) \rceil$ machines of speed σ.

We claim that, for machines $\mathcal{M}'$, we have $C_{\max}^{*,\Psi,\sum p_j=P}[\mathcal{J}'_{[i]}] \leq O_i$. We extend $\mathcal{J}'_{[i]}$ by sufficiently many jobs of size at most σO_1 so that the total processing time is $\bar{P}$; this extension is in Ψ for a sufficiently small size of jobs, since Ψ allows padding. We can schedule the jobs of $\mathcal{J}'_{[i]}$ in time O_i on the original machines. Furthermore, we can schedule the remaining jobs with the total processing time at most $\bar{P}$ on the added machines of speed σ so that they complete before time O_1; this is guaranteed by the choice of the number of new machines and the upper bound on the job size. Since $O_1 \le O_i$, the extension can be completed by time O_i and thus the extension witnesses our claim.

The claim, together with Definition 2.3.7 and $P(\mathcal{J}') = \bar{P}$ implies that

$$\bar{r}^{\Psi,\sum p_j = P}(\mathcal{M}', \mathcal{J}') \ge \frac{\bar{P}}{\sum_{i=1}^{n} s'_{n-i+1} O_i.}$$

At the same time we have:

$$\bar{r}^{\Psi}(\mathcal{M}, \mathcal{J}) = \bar{r}^{\Psi}(\mathcal{M}, \mathcal{J}') = \frac{\bar{P}}{\sum_{i=1}^{n} s_{n-i+1} O_i}.$$

To complete the proof, it now suffices to compare the denominators in the previous two formulas. Using the condition from the definition of σ, we have:

$$\begin{aligned} \sum_{i=1}^{n} s'_{n-i+1} O_i &\le \sum_{i=1}^{n} s_{n-i+1} O_i + n\sigma O_n \\ &\le \sum_{i=1}^{n} s_{n-i+1} O_i + \delta s_1 O_n \\ &\le (1+\delta) \sum_{i=1}^{n} s_{n-i+1} O_i. \end{aligned}$$

Thus:

$$\begin{aligned} \bar{r}^{\Psi,\sum p_j = P}(\mathcal{M}', \mathcal{J}') &\ge \frac{\bar{P}}{\sum_{i=1}^{n} s'_{n-i+1} O_i.} \\ &\ge \frac{\bar{P}}{(1+\delta)\sum_{i=1}^{n} s_{n-i+1} O_i} = \frac{\bar{r}^{\Psi}(\mathcal{M}, \mathcal{J})}{1+\delta}. \end{aligned}$$ □

Taking void restriction for Ψ, i.e. the original online problem, we get that overall competitive ratio for $\sum p_j = P$ is equal to overall competitive ratio for the original online problem.

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

Here we are given $\bar{p}$, the maximal size of a job. As noted in the introduction, any algorithm that works with the first job being the maximal one can be easily changed to a general algorithm for this restriction. First it virtually schedules the maximal job and then it compares the size of each job to $\bar{p}$. If it is equal for the first time, it schedules the job to the time slot(s) it reserved by virtual scheduling at the beginning. Other jobs are scheduled in the same way in both algorithms. Thus we can work with the equivalent restriction Ψ containing all the sequences where the first job is a maximal one, i.e., $p_1 \ge p_i$ for all $i \le n$. Then $\mathrm{pref}(\Psi) = \Psi$ and $C^{*,\Psi}_{\max}[\mathcal{J}] = C^{*}_{\max}[\mathcal{J}]$ for any $\mathcal{J} \in \Psi$.

Using the same argument as in the proof of Observation 2.3.10, by Observation 2.3.9, the other jobs can be reordered as in the previous case, and we can maximize only over sequences with non-decreasing job sizes from the second job on. Furthermore, we can replace the first and last jobs (which are now the two largest jobs) by two jobs with processing time $(p_1 + p_n)/2$. By Observation 2.3.9, the value $\bar{r}^{\Psi}(\mathcal{M}, \mathcal{J})$ cannot decrease.

We proceed towards the formulation of the linear programs. In this case we need to solve separately m mathematical programs, one for each $n = 1, \ldots, m-1$ and one for $n \geq m$, and take the maximum value. (The program for $n = 1$ is trivial, sequence of one job bounds competitive ratio trivially by 1.) In all the programs we can use scaling and normalization, to obtain a linear program, using a variable p for p_1 instead of a constant $\bar{p}$. Again, we cannot prepend zero sized jobs here to get one linear program.

The resulting linear programs are the following. For each $n < m$, we have a linear program

$$\begin{array}{rcll} \textbf{maximize} & & R_n(s_1, s_2, \ldots, s_m) = p + q_2 + \cdots + q_n & \\ \textbf{subject to} & & & \\ 1 & = & s_1 O_n + \cdots + s_n O_1 & \\ p & \leq & s_1 O_k & 1 \leq k \leq n \\ p + q_{j+1} + \cdots + q_k & \leq & (s_1 + \cdots + s_{k-j+1}) O_k & 1 \leq j < k \leq n \\ 0 & \leq & q_2 & \\ q_k & \leq & q_{k+1} & 1 \leq k \leq n-1 \\ q_n & = & p & \text{(substitution)} \end{array} \tag{2.10}$$

For $n \geq m$, we have a single linear program. Here the variables are p for the processing time of the first job, q_1 for the total processing time of the small jobs, i.e., $q_1 = p_2 + \cdots + p_{n-m+1}$, and $q_i = p_{n-m+i}$, for $i = 2, \ldots, m$.

$$\begin{array}{rcll} \textbf{maximize} & & R(s_1, s_2, \ldots, s_m) = p + q_1 + \cdots + q_m & \\ \textbf{subject to} & & & \\ 1 & = & s_1 O_m + \cdots + s_m O_1 & \\ p + q_1 + \cdots + q_k & \leq & S O_k & 1 \leq k \leq m \\ p & \leq & s_1 O_k & 1 \leq k \leq m \\ p + q_{j+1} + \cdots + q_k & \leq & (s_1 + \cdots + s_{k-j+1}) O_k & 1 \leq j < k \leq m \\ 0 & \leq & q_1 & \\ 0 & \leq & q_2 & \\ q_k & \leq & q_{k+1} & 1 \leq k \leq m-1 \\ q_m & = & p & \text{(substitution)} \end{array} \tag{2.11}$$

To verify that the maximum of the m linear programs is $r^{p_{\max}=p}(\mathcal{M})$, we observe that from each input $\mathcal{J}$ we can construct a feasible solution of one of the linear programs with objective at least $\bar{r}^{p_{\max}=p}(\mathcal{M}, \mathcal{J})$ and vice versa. The first direction, from the instance to a feasible solution, is described during the construction of the linear programs.

Given a feasible solution of a linear program for $n < m$, the corresponding input is simply the sequence of n jobs with $p_1 = p$ and $p_i = q_i$ for $i = 2, \cdots, n$.

Given a feasible solution of a linear program for $n \geq m$, we need to replace q_1 by a prefix of jobs with total processing time q_1. We choose $k = \max\{m, \lceil q_1/p \rceil\}$. The input instance has $n = m + k$ jobs and the processing times are $p_1 = p$, $p_2 = \cdots = p_{k+1} = q_1/k$, and $p_{k+i} = q_i$ for $i = 2, \ldots, m$. The choice of k guarantees that p is the maximal processing time. The fact that we replace q_1 by at least m jobs of equal size guarantees that the maximum in (2.3) is given either by P/S or by P_i/S_i for some i such that P_i does not contain any of these jobs replacing q_1. Thus $C^{*,\Psi}_{\max}[\mathcal{J}_{[n-i+1]}] \leq O_i$ for all $i = 1, \ldots, m$, and the lower bound given by $\mathcal{J}$ is at least the objective of the linear program.

We do not know any better upper bound than the e-competitive algorithm for the general online problem. We used same technique (i.e. numerical search for local optimum of quadratic program) as in the online variant to obtain lower bound of 1.908 by a sequence for $m = 200$ machines. We attach a file with lower bounds for $m = 2, \ldots, 200$ suitable for computer verification in the electronic version of this thesis.

2.3.5 Approximately known optimum, $T \leq C^*_{\max} \leq \alpha T$

If some value from previous restrictions is given not exactly but it is only known to belong to some interval, typically it means that the linear program is weakened by relaxing some equation to a pair of inequalities, or by relaxing some inequality, then the optimal ratio is again computed using a linear program.

In this particular variant, we have two parameters, $\bar{T}$ and $\alpha > 1$. The semi-online restriction contains all job sequences with the optimal makespan between $\bar{T}$ and $\alpha\bar{T}$. We proceed towards the formulation of the linear programs. Here we need to solve separately m mathematical programs, one for each $n = 1, \ldots, m-1$ and one for $n \geq m$, and take the maximum value.

We now describe the usual simplifications; they apply to all the m mathematical programs. Since we can permute the jobs and increasing the size of the last job does not decrease $C^{*,\Psi}_{\max}$, Observation 2.3.10 implies that we can restrict ourselves to non-decreasing sequences $\mathcal{J}$.

To express the constraint bounding the optimum, one can simply assert that $\bar{T} \leq O_k \leq \alpha\bar{T}$. We can see that the mathematical program scales and its value does not depend on the value of $\bar{T}$. Thus we can treat $\bar{T}$ as a variable and normalize the denominator of the objective function to be 1 as usual. To further simplify the linear program, we note that in the optimal solution we have $O_1 \leq O_2 \leq \cdots$, i.e., the values of the variables for the optima are ordered. Then the constraint for bounding the optima in terms of $\bar{T}$ simplifies to $O_n \leq \alpha O_1$ or $O_m \leq \alpha O_1$ and the variable $\bar{T}$ no longer appears in the program.

The resulting linear programs are the following. For each $n < m$, we have the linear program

$$\begin{array}{rcll}
\textbf{maximize} & & R_n(s_1, s_2, \ldots, s_m) = q_1 + q_2 + \cdots q_n & \\
\textbf{subject to} & & & \\
1 & = & s_1 O_n + s_2 O_{n-1} + \cdots + s_n O_1 & \\
q_j + q_{j+1} + \cdots + q_k & \leq & (s_1 + s_2 + \cdots + s_{k-j+1}) O_k & \text{for } 1 \leq j \leq k \leq n \\
q_j & \leq & q_{j+1} & \text{for } j = 1, \ldots, n-1 \\
0 & \leq & q_1 & \\
O_k & \leq & O_{k+1} & \text{for } k = 1, \ldots, n-1 \\
O_n & \leq & \alpha O_1 &
\end{array} \tag{2.12}$$

For $n \geq m$ we have a single linear program, where the variable q_1 denotes the sum of the first processing times, i.e., $q_1 = p_1 + p_2 + \cdots + p_{n-m+1}$.

$$\begin{array}{rcll}
\textbf{maximize} & & R(s_1, s_2, \ldots, s_m) = q_1 + q_2 + \cdots q_m & \\
\textbf{subject to} & & & \\
1 & = & s_1 O_m + s_2 O_{m-1} + \cdots + s_m O_1 & \\
q_1 + q_2 + \cdots + q_k & \leq & S O_k & \text{for } 1 \leq k \leq m \\
q_j + q_{j+1} + \cdots + q_k & \leq & (s_1 + s_2 + \cdots + s_{k-j+1}) O_k & \text{for } 2 \leq j \leq k \leq m \\
q_j & \leq & q_{j+1} & \text{for } j = 2, \ldots, m-1 \\
0 & \leq & q_1 & \\
0 & \leq & q_2 & \\
O_k & \leq & O_{k+1} & \text{for } k = 1, \ldots, m-1 \\
O_m & \leq & \alpha O_1 &
\end{array} \tag{2.13}$$

To verify that the maximum of the m linear programs is $r^{\Psi}(\mathcal{M})$, we observe that from each input $\mathcal{J}$ we can construct a feasible solution of one of the linear programs with objective at least $\bar{r}^{\Psi}(\mathcal{M}, \mathcal{J})$ and vice versa. The first direction, from the instance to a feasible solution, is again described during the construction of the linear programs.

Given a feasible solution of a linear program for $n < m$, the corresponding input is simply the sequence of n jobs with $p_i = q_i$. Given a feasible solution of a linear program for $n \geq m$, the corresponding input has $n = 2m - 1$ jobs and the processing times are $p_1 = \cdots = p_m = q_1/m$ and $p_{m+i} = q_{i+1}$ for $i = 1, \ldots, m-1$. In both cases we scale the jobs to get $\bar{T} = O_1$. Again, as for the previous restriction, the optima O_i are computed correctly and we obtain a desired lower bound on the approximation ratio.

Note that the bound for $m = 200$ machines for the online scheduling holds also here if $\alpha \geq 34.89$. The opposite extreme ($\alpha = 1$) is rather trivial as we know that known optimal value suffices for the algorithm to be 1-competitive.

2.3.6 Tightly grouped processing times, $p \leq p_j \leq \alpha p$

In this case we have two parameters $\bar{p}$ and α, and the restriction Ψ contains all sequences $\mathcal{J}$ with $\bar{p} \leq p_j \leq \alpha\bar{p}$, for all $j = 1, \ldots, n$. Once again, Ψ is closed under taking subsequences and, in particular, $\mathrm{pref}(\Psi) = \Psi$ and $C^{*,\Psi}_{\max}[\mathcal{J}] = C^{*}_{\max}[\mathcal{J}]$ for all $\mathcal{J} \in \Psi$.

The case of tightly grouped jobs is interesting, because it is not clear how to set up a single linear program for a sufficiently large n. The problem is that the possible total processing times of small jobs do not form a single interval, and thus the domain of the mathematical program is not convex. However, we observe that this can be done by solving one linear program to obtain another parameter. Then we use this parameter in another two linear programs and take the maximum of their results.

To form the linear programs for $n < 2m-1$ is routine, even though in this case we possibly need also values $n > m$ (in which case we use the notation $s_i = 0$ for $i > m$). Since Ψ is closed under permutations, we may restrict ourselves to sequences with non-decreasing processing times. The restriction on the processing times is now handled similarly as in the previous case the restriction of approximately known optimum. Also, we can normalize as usual. The resulting linear program for a fixed $n < 2m-1$ is:

$$\begin{array}{rcll} \textbf{maximize} & & R_n(s_1, s_2, \ldots, s_m) = q_1 + q_2 + \cdots q_n & \\ \textbf{subject to} & & & \\ 1 & = & s_1 O_n + s_2 O_{n-1} + \cdots + s_n O_1 & \\ q_j + q_{j+1} + \cdots + q_k & \leq & (s_1 + s_2 + \cdots + s_{k-j+1}) O_k & \text{for } 1 \leq j \leq k \leq n \\ q_j & \leq & q_{j+1} & \text{for } j = 1, \cdots, n-1 \\ q_n & \leq & \alpha q_1 & \end{array} \tag{2.14}$$

To solve the case when $n \geq 2m-1$ we use the standard trick with summing first $n-m+1$ jobs to variable q_1. We have to solve a mathematical program consisting of following linear program:

$$\begin{array}{rcll} \textbf{maximize} & & R'(s_1, s_2, \ldots, s_m) = q_1 + q_2 + \cdots q_n & \\ \textbf{subject to} & & & \\ 1 & = & s_1 O_m + s_2 O_{m-1} + \cdots + s_m O_1 & \\ q_1 + q_2 + \cdots + q_k & \leq & S O_k & \text{for } 1 \leq k \leq m \\ q_j + q_{j+1} + \cdots + q_k & \leq & (s_1 + s_2 + \cdots + s_{k-j+1}) O_k & \text{for } 1 \leq j \leq k \leq m \\ q_j & \leq & q_{j+1} & \text{for } j = 2, \cdots, m-1 \\ q_m & \leq & \alpha q_2 & \\ m q_m & \leq & \alpha q_1 & \end{array} \tag{2.15}$$

with additional constraint: For every integer $N \geq m$ at least one of following condition holds:

$$\begin{array}{rcl} (N+1) q_m & \leq & \alpha q_1 \\ q_1 & \leq & N q_2 \,. \end{array} \tag{2.16}$$

The additional constraint just says that either there are at most N jobs in q_1 and thus $q_1 \leq N q_2$ as all jobs that are forming q_1 must have smaller processing time than q_2. The other case is that there are at least $N+1$ jobs forming q_1, then we know that $\alpha q_1 \geq (N+1) q_m$ because of the ratio between smallest and largest job.

Having an optimal solution $\mathbf{x}^*$ to (2.15) two cases may occur. First, it may be also feasible to (2.16) for all $N \geq m$. Then we are done as $R(s_1, s_2, \dots, s_m) = sup_{J:|\mathcal{J}|>2m-1}\bar{r}^{\Psi}(\mathcal{M}, \mathcal{J})$. If this is not the case, then there is N for which $\mathbf{x}^*$ does not satisfy (2.16). Let this N be N^*. (In fact there can be only one such N, but we will not need that.) We use the separation principle of linear programming, to obtain that there is an optimal solution for (2.15 & 2.16) which satisfy one of the two conditions in (2.16) for N^* by equality. This is because (2.15) & $(N^*+1)q_m \leq \alpha q_1$ is linear program as well as (2.15) & $q_1 \leq N^* q_2$. Feasible solution to (2.15 & 2.16) is feasible to at least one of these two programs. Moreover $\mathbf{x}^*$ is not feasible to any of these programs, thus optimal solutions of these programs satisfies the additional inequality by equality. Then we can easily see that the optimal solutions of these two programs must be feasible in (2.15) because of the nature of conditions (2.16). Thus we simply solve these two additional linear programs. We take the maximum of the results of these two programs to obtain $sup_{J:|\mathcal{J}|>2m-1}\bar{r}^{\Psi}(\mathcal{M}, \mathcal{J})$.

We can see that any solution corresponding to sorted input sequence will satisfy (2.16). Now we show the other direction: from feasible solution to the input sequence. There will be N^* such that the first condition from (2.16) holds for $N < N^*$ and the second condition holds for $N \geq N^*$. This gives that $q_m/\alpha \leq q_1/N^* \leq q_2$ and we can split q_1 into N^* jobs of size q_1/N^*. Then we continue with jobs of size $q_2, q_3, \dots, q_m$. This gives a valid (sorted) input sequence, thus the linear programs compute $\bar{r}^{p \leq p_j \leq \alpha p}(\mathcal{M})$ correctly.

The bound for $m = 200$ machines for the online scheduling holds also here if $\alpha \geq 342.5$. The opposite extreme is when $\alpha = 1$, then we get a sequence of unit jobs, which is the worst case for the nondecreasing processing times as we will see in the next subsection.

2.3.7 Non-increasing processing times, *decr*

Among other restrictions, we are also interested in sequences of non-increasing jobs, as this is one of the most studied restrictions. Now the restriction Ψ contains sequences which have $p_j \geq p_{j+1}$ for all j. Note that Ψ is closed under taking subsequences and, in particular, pref$(\Psi) = \Psi$ and $C^{*,\Psi}_{\max}[\mathcal{J}] = C^*_{\max}[\mathcal{J}]$ for all $\mathcal{J} \in \Psi$.

We cannot swap jobs, however, we can replace all the jobs by jobs with all processing times equal to the arithmetic mean of the original processing times. Since the jobs are non-increasing, this cannot increase the sum of processing times in any initial segment. Consequently, as the k largest jobs of any prefix are a (possibly shorter) prefix of $\mathcal{J}$, this transformation cannot increase any $C^{*,\Psi}_{\max}[\mathcal{J}_{[i]}]$. Thus, by Observation 2.3.9, to compute $r^{\Psi}[\mathcal{J}]$, we may restrict ourselves to instances $\mathcal{J}$ with equal processing times, i.e., $p_1 = p_2 = \cdots = p_n$. By scaling, the actual size of jobs does not matter, we only need to determine the length of the sequence which gives the highest ratio.

Let us denote $\hat{r}_n(\mathcal{M}) = \bar{r}^{decr}(\mathcal{M}, \mathcal{J})$ for a sequence $\mathcal{J}$ with n jobs with $p_j = 1$. For this sequence, $C^{*,\Psi}_{\max}[\mathcal{J}] = C^*_{\max}[\mathcal{J}] = n/S_n$. (Recall that $s_i = 0$ for $i > m$ and

$S_k = \sum_{i=1}^{k} s_i$.) Using this for the prefixes, we obtain

$$\hat{r}_n(\mathcal{M}) = n \cdot \left(\sum_{k=1}^{n} \frac{k s_{n-k+1}}{S_k} \right)^{-1} . \tag{2.17}$$

Using Observation 2.3.8 and the previous transformations, we obtain that $r^{\Psi}(\mathcal{M}) = \sup_n \hat{r}_n(\mathcal{M})$. It can be seen that for any speed vector, the sequence $\hat{r}_n(\mathcal{M})$ decreases with n for $n \geq 2m$. Thus computing the approximation ratio for any given speeds reduces to finding a maximum of $2m$ closed formulas, and this is efficient.

The overall ratio

We know exact formula for any set of speeds in this variant of semi-online scheduling. Thus we will try to examine the overall competitive ratio more deeply. A natural approach to estimate the overall ratio is to find for each n the worst speed vector and the corresponding ratio $\hat{r}_n = \sup_{\mathcal{M}} \hat{r}_n(\mathcal{M})$. Based on numerical experiments, we conjecture that for each n, $\hat{r}_n$ is attained for some $\mathcal{M}$ with speeds $s_1 = s_2 = \cdots = s_{m-1}$. I.e., almost all the speeds are equal. This conjecture would imply that with non-increasing jobs, the optimal overall approximation ratio is the same for the uniformly related machines and for the identical machines, and this is equal to $(1+\sqrt{3})/2 \approx 1.366$ by [SSW00]. A subtle detail is that our conjecture allows one machine with a smaller speed (and from the calculations we know that this is necessary). However, with increasing number of machines, the difference from the value of the formula (2.17) with the last machine removed decreases to 0. Since the competitive ratio for identical machines increases with the number of machines, as shown in [SSW00], the limits would be equal.

This is related to an intriguing geometric question. Suppose we have numbers x_i, y_i, $i = 1, \ldots, n$ such that $x_i y_i = i$ for all i and both sequences $(x_i)_{i=1}^n$ and $(y_i)_{i=1}^n$ are non-decreasing. Consider the union of rectangles $[0, x_i] \times [0, y_{n+1-i}]$ over all i; this is a staircase-like part of the positive quadrant of the plane. What is the smallest possible area of this union of rectangles? Any speed vector can be turned into an instance of the geometric sequence with $x_i = S_i$ and $y_i = i/S_i$. Then, setting $x_0 = 0$, the area is equal to

$$\sum_{i=1}^{n} (x_i - x_{i-1}) y_{n-i+1} = \sum_{k=1}^{n} \frac{k s_{n-k+1}}{S_k},$$

which is inversely proportional to $\hat{r}_n(\mathcal{M})$. We conjecture that the minimum of the geometric problem is attained for an instance with $y_1 = y_2 = \ldots = y_k$ and $x_{k+1} = x_{k+2} = \ldots = x_n$ for some k. This would imply the previous conjecture.

We are not able to determine exactly the values of $\hat{r}_n$, but we can prove certain relations between these values. In particular, for any integers a, n, and n', $\hat{r}_{an} \geq \hat{r}_n$ and $\hat{r}_{n'} \leq \frac{n+1}{n} \hat{r}_n$. For the first proof, we replace a sequence of speeds from the bound for $\hat{r}_n$ by a sequence where each speed is repeated a times, and the bound follows by manipulating the formula for $\hat{r}_n$. The second inequality is shown by replacing the speeds for $\hat{r}_{n'}$ by a shorter sequence where each new speed is a sum of a segment of a speeds in the original sequence, for a suitable a. These relations show that whenever we are able to evaluate some $\hat{r}_n$ for a fixed n, the optimal overall ratio is at most $\frac{n+1}{n}\hat{r}_n$.

Lemma 2.3.12 *For any positive integers n and a and any set of machines $\mathcal{M}$ there exists a set of machines $\mathcal{M}'$ such that $\hat{r}_n(\mathcal{M}) \le \hat{r}_{an}(\mathcal{M}')$. Consequently $\hat{r}_n \le \hat{r}_{an}$.*

Proof: We choose $\mathcal{M}'$ so that it has a machines with each speed s_i. Formally, we set $s'_{ua-v} = s_u$ for any positive integer u and $v = 0, \ldots, a-1$. Let $S'_k = \sum_{i=1}^{k} s'_k$, and recall that $S_u = \sum_{i=1}^{u} s_u$.

Let $k = ua - v$ for some positive integer u and $v \in \{0, \ldots, a-1\}$. We claim that $S'_k/k \ge S_u/u$: We are comparing two averages of some sets of speeds. In S'_k we sum a copies of each speed in the sum S_u, except that v copies of the smallest speed are omitted; thus the average can only increase. Observe also that $s'_{an-k+1} = s_{n-u+1}$. Thus

$$\frac{ks'_{an-k+1}}{S'_k} \le \frac{us_{n-u+1}}{S_u}. \tag{2.18}$$

In the middle step of the following derivation, we use (2.18) for each term in the sum (obtaining a equal terms for each u). The first and the last steps follow from (2.17).

$$\begin{aligned}(\hat{r}_{an}(\mathcal{M}'))^{-1} &= \frac{1}{an}\sum_{k=1}^{an}\frac{ks'_{an-k+1}}{S'_k}\\ &\le \frac{1}{an}\sum_{u=1}^{n} a\frac{us_{n-u+1}}{S_u} = \frac{1}{n}\sum_{u=1}^{n}\frac{us_{n-u+1}}{S_u} = (\hat{r}_n(\mathcal{M}))^{-1}.\end{aligned}$$

□

Lemma 2.3.13 *For any positive integers n and n', $\hat{r}_{n'} \le \frac{n+1}{n} \cdot \hat{r}_n$.*

Proof: Let $N \ge n$ and let $a = \left\lfloor \frac{N+1}{n+1} \right\rfloor$. We prove that $\hat{r}_N \le \frac{N}{an} \cdot \hat{r}_n$.

First we show that this implies the lemma. For $N \to \infty$, $\frac{N}{an}$ converges to $\frac{n+1}{n}$. Thus $\limsup_{N\to\infty} \hat{r}_N \le \frac{n+1}{n} \cdot \hat{r}_n$. If N is a multiple of n', Lemma 2.3.12 implies that $\hat{r}_{n'} \le \hat{r}_N$. Since the multiples can be taken arbitrarily large, together with the limit property above this implies $\hat{r}_{n'} \le \limsup_{N\to\infty} \hat{r}_N \le \frac{n+1}{n} \cdot \hat{r}_n$.

Now consider an arbitrary set of machines $\mathcal{M}'$. We construct a set of machines $\mathcal{M}$ such that $\hat{r}_N(\mathcal{M}') \le \frac{N}{an}\hat{r}_n(\mathcal{M})$. This implies that $\hat{r}_N \le \frac{N}{an} \cdot \hat{r}_n$.

Denote again $S'_k = \sum_{i=1}^{k} s'_k$. We choose the speeds $\mathcal{M}$ so that we divide $\mathcal{M}'$ into groups of a speeds, and s_u is the sum of the speeds in the uth group. Formally, $s_u = \sum_{v=0}^{a-1} s'_{ua-v}$.

By (2.17) we have

$$(\hat{r}_N(\mathcal{M}'))^{-1} = \frac{1}{N}\sum_{k=1}^{an}\frac{ks'_{an-k+1}}{S'_k} \geq \frac{1}{N}\sum_{u=1}^{n}\sum_{v=0}^{a-1}\frac{(N-na+ua-v)s'_{na-ua+v+1}}{S'_{N-na+ua-v}}\,,$$

where the inequality follows by dropping the first $N-an$ terms in the sum and grouping and reindexing the remaining ones, using the substitution $k = N-na+ua-v$ for some $u \geq 1$ and $v \in \{0,\dots,a-1\}$.

Now we proceed towards bounding the inner sums. We can see $S'_{N-na+ua-v}/(N-na+ua-v) \leq S'_{ua}/(ua)$, as by the choice of a we have $N \geq an+a-1 \geq na+v$ and thus on the left-hand side we take the average of the same speeds as on the right-hand side, plus possibly some smaller ones. Thus we have

$$\sum_{v=0}^{a-1}\frac{(N-na+ua-v)s'_{na-ua+v+1}}{S'_{N-na+ua-v}} \geq \frac{ua}{S'_{ua}}\cdot\sum_{v=0}^{a-1}s'_{na-ua+v+1} = \frac{ua}{S_u}\cdot s_{n-u+1}\,.$$

Using this bound for the inner sums we have

$$(\hat{r}_N(\mathcal{M}'))^{-1} \geq \frac{1}{N}\sum_{u=1}^{n}\left(\frac{ua}{S_u}\cdot s_{n-u+1}\right) = \frac{na}{N}\cdot\frac{1}{n}\sum_{u=1}^{n}\frac{us_{n-u+1}}{S_u} = \frac{na}{N}(\hat{r}_n(\mathcal{M}))^{-1}\,.$$

□

Now we consider small values of n. For $n=3$ the formula is

$$\hat{r}_3(s_1,s_2,s_3) = \left(\frac{s_3}{3s_1}+\frac{2s_2}{3(s_1+s_2)}+\frac{s_1}{s_1+s_2+s_3}\right)^{-1}$$

Maximizing the function $\hat{r}_3(\mathcal{M})$ can be done by hand and the maximum is $\hat{r}_3 = 1.2$ for $s_1 = s_2 = 1$, $s_3 = 0$. This yields an overall upper bound of $\hat{r}_n \leq \frac{4}{3}\cdot\frac{6}{5} = 1.6$. By a computer-assisted proof we have shown that $\hat{r}_4 = (\sqrt{7}+1)/3 \approx 1.215$ and $\hat{r}_5 = \frac{5}{4} \approx 1.25$, yielding an overall upper bound of $\hat{r}_n \leq \frac{6}{5}\hat{r}_5 = \frac{3}{2} = 1.5$.

2.3.8 Combined restrictions

If Ψ is given as an intersection of two standard restrictions, the same methods for reducing the number of candidates for the worst case instances apply. Sometimes we need to be somehow careful as some simplifications connected to the two restrictions are not compatible. We give two examples, in each we know the total processing time and in the first one also the maximal processing time, while in the second the jobs are non-increasing. In both cases Theorem 2.3.11 allows padding and thus the overall approximation ratio is the same as without knowledge of the total processing time.

Known total and maximal processing times, $\sum p_j = P, p_{\max} = p$

In this section we construct the linear program for the first combined semi-online restriction. We are given two parameters, $\bar{P}$ and $\bar{p}$. The restriction Ψ contains all inputs $\mathcal{J}$ with $P = \bar{P}$, $p_1 = \bar{p}$, and $p_j \le \bar{p}$ for all $j \le n$. The interesting feature of this variant is that not all the simplifications used before are possible, as they could change the ratio of P and the maximal processing time. The overall approximation ratio is the same as for the restriction $p_{\max} = p$ as we can use padding, i.e., Theorem 2.3.11.

Similarly to the case $p_{\max} = p$, the restriction of knowing the maximal processing time is equivalent to the restriction to the set of inputs where the first job is the maximal one. Furthermore, we may restrict ourselves to partial inputs with the remaining jobs with non-decreasing processing times. Similarly to the case of $\sum p_j = P$, we may restrict ourselves to partial inputs with $n < m$.

On the other hand, unlike in the previous cases, we cannot scale so that $P = \bar{P}$ and we cannot assume that the last job has the maximal processing time. These arguments break down, as the appropriate transformation changes the ratio of P and $p_1 = p_{\max}$.

Let $\beta = (\bar{P} - \bar{p})/\bar{p}$. The mathematical program for a fixed $n < m$ follows:

$$\begin{array}{rcll}
\textbf{maximize} & & R_n(\mathcal{M}) = p + q_2 + \cdots + q_n & \\
\textbf{subject to} & & & \\
1 & = & s_1 O_n + \cdots + s_1 O_n & \\
(1+\beta)p & \le & SO_k & \text{for } 1 \le k \le n \\
p & \le & s_1 O_k & \text{for } 1 \le k \le n \\
p + q_{j+1} + \cdots + q_k & \le & (s_1 + \cdots + s_{k-j+1}) O_k & \text{for } 1 \le j < k \le n \\
q_2 + \cdots + q_n & \le & \beta p & \\
q_n & \le & p & \\
0 & \le & q_2 & \\
q_{k-1} & \le & q_k & \text{for } 3 \le k \le n\,.
\end{array} \tag{2.19}$$

Note that we convert here sequences from pref(Ψ), thus $\sum_{J \in \mathcal{J}} p(J)$ is at most $\bar{P}$, but it needs not to be equal. This corresponds to the fact that the fourth inequality in (2.19) is not equality. The correspondence between input sequences and feasible solutions works the same way as in previous cases.

We can also see that Theorem 2.3.11 applies here, thus the overall ratio is the same as for the variant of knowing only the maximal processing time.

Known total processing times and non-increasing jobs, $\sum p_j = P, decr$

We conclude by a restriction Ψ containing all sequences with total processing time equal to a parameter $\bar{P}$ and non-increasing processing times. Similarly as for non-increasing processing times, we do not need to solve linear programs.

Again, similarly to the non-increasing jobs, we can assume that all the jobs are equal; we normalize their size to be 1. Once all jobs are equal we have

$$C^{*,\Psi}_{\max}[\mathcal{J}] = \max\left\{\frac{n}{S_n}, \frac{\bar{P}}{S}\right\} \tag{2.20}$$

It must hold that $\bar{P} \geq n$, as otherwise the sequence cannot be completed to satisfy $\sum p = \bar{P}$. If $n \geq m$, then all optima are equal, yielding approximation ratio of 1. I.e., if the algorithm gets first job smaller or equal $\bar{P}/m$, it can always schedule all jobs optimally. So we can restrict ourselves only to cases with $n < m$.

Also, we can scale whole sequence by $b^{-1} = \bar{P}/n$, i.e., the scaled sequence has $p_j = \bar{P}/n$. From Observation 2.3.9 we get that the scaled sequence yields larger lower bound. Thus we can restrict ourselves to instances with $\bar{P} = n$.

Let us denote $\hat{r}_n(\mathcal{M}) = \bar{r}^{\sum p_j = P, decr}(\mathcal{M}, \mathcal{J})$ for a sequence $\mathcal{J}$ with n jobs with $p_j = 1$ and for $\bar{P} = n$. Using Observation 2.3.8 and the previous transformations, we obtain that $r^{\sum p_j = P, decr}(\mathcal{M}) = \max_{n=1}^{m-1} \hat{r}_n(\mathcal{M})$.

Using (2.20) we obtain

$$(\hat{r}_n(\mathcal{M}))^{-1} = \sum_{k=1}^{n} s_{n-k+1} \cdot \max\left\{\frac{k}{nS_k}, \frac{1}{S_n}\right\}. \tag{2.21}$$

Note that because k/S^k is nondecreasing with k, the maximum for the first few values of k is $1/S_n$ and for the remaining ones it is $k/(nS^k)$.

Theorem 2.3.11 applies here also, giving that the overall ratio is the same as for the variant of knowing only that the processing times are non increasing.

2.4 Small number of machines

Here we give closed form formulas for competitive ratio for various semi-online scheduling problems on two, three and sometimes four machines. Optimizing makespan on one machine is trivial in our problem, thus we omit this case. Note that this means that competitive ratio of scheduling on one machine is 1.

We have already showed the linear programs for respective restrictions. In this section we solve them parametrically, for a small number of machines. We use the duality of linear programming, 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.

We will need also new shorthand for a speed ratio which occurs often in resulting formulas:

$$\rho = \frac{S - s_1}{S} = \frac{s_2 + \cdots + s_m}{s_1 + s_2 + \cdots + s_m} \tag{2.22}$$

2.4.1 Online scheduling

We repeat the linear program here, because we need to label every inequality with corresponding dual variable. This allows us to give clear proof of feasibility and optimality of listed solutions. The linear program has variables $q_1, \dots, q_m$ and $O_1, \dots, O_m$ as in previous section. We label each inequality with corresponding dual variable $(z_?)$. The value of the optimal solution is the competitive ratio of the problem for m machines.

$$\begin{array}{rcllll}
\textbf{maximize} & & R = q_1 + \cdots + q_m & & & \\
\textbf{subject to} & & & & & \\
1 & = & s_1 O_m + s_2 O_{m-1} + \cdots + s_m O_1 & (z_{norm}) & & \\
q_1 + \cdots + q_k & \le & (s_1 + \cdots + s_m) O_k & (z_k) & 1 \le k \le m & \\
q_j + \cdots + q_k & \le & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) & 2 \le j \le k \le m & (2.23)\\
q_j & \le & q_{j+1} & (z_{\le,j}) & 2 \le j \le m-1 & \\
0 & \le & q_1 & (z_{0,1}) & & \\
0 & \le & q_2 & (z_{0,2}) & &
\end{array}$$

So we have the linear program and we solve it for all speed combinations of two, three and four machines ($m = 2, 3, 4$).

Beyond nondecreasing speed ratios

Theorem 2.4.1 *Let* $R = \left(\sum_{i=1}^{m} \frac{s_i}{S} \rho^{i-1} \right)^{-1}$.
Then $r(\mathcal{M}) \le R$ for any speeds $s_1 \ge \cdots \ge s_m$ and thus RatioStretch *is R-competitive. Furthermore $r^{online}(\mathcal{M}) = R$ whenever*

$$\left(1 + \rho + \cdots + \rho^{k-1}\right) s_1 \quad \le \quad s_1 + \cdots + s_k \qquad \textit{for all } k = 2, \dots, m-1.$$

We show here a solution of (2.23) which is optimal under certain conditions. This special case was solved already in [Eps01], although it was presented as the more strict case of nondecreasing speed ratios, i.e., the case where $s_{i-1}/s_i \le s_i/s_{i+1}$.

Note that the conditions are trivially satisfied for $m = 2$ and thus this also gives full solution to the case of two machines.

We list the case here in format that we use for all cases of all semi-online restrictions. 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 according Definition 2.3.7. Note that the algorithm RatioStretch matches exactly this lower bound, i.e., if it fails to achieve some competitive ratio, then it is due to the bound in Definition 2.3.7.

(Recall the definition $\rho = \frac{S - s_1}{S}$.)

Case I Ratio: $r = \dfrac{S}{\sum\limits_{i=1}^{m} \rho^{i-1} s_i}$

Conditions:

$$(A_k^+) \quad \sum_{i=1}^{k} s_k \;\geq\; \sum_{i=1}^{k} \rho^{k-1} s_1 \quad \text{for } k = 2, \ldots, m-1$$

Common denominator: $D = \sum\limits_{i=1}^{m} \rho^{i-1} s_i$

Nonbasic dual vars:	U.b. coefficients:
all but: $z_1, z_2, \ldots, z_m,$ $z_{2,2}, z_{3,3}, \ldots, z_{m,m}$	$z_{norm} = -r$
Jobs:	$z_{k,k} = \sum\limits_{i=2}^{k} s_{m-k+i}\, \rho^{i-2}/D$
$q_1 = \rho^{m-1} S/D$	for $k = 2, \ldots, m$
$q_k = s_1 \rho^{m-k}/D$	$z_k = s_{m-k+1}/D - \frac{s_1}{S} z_{k,k}$
for $k = 2, \ldots, m$	for $k = 1, \ldots, m$

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 (A_k^+) is a label of k-th general condition of type A. We will also use (A_k^-) to denote the opposite condition in adjacent cases. *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 $C_{\max}^{*,\Psi}$. 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 the 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 will show how to check the correctness of Case I. The correctness of all other cases and all other restrictions in this section can be checked in the same way. First we check the value of the objective function:

$$r \;=\; \sum_{j=1}^{m} q_j \;=\; \left(\rho^{m-1} + (1-\rho) \sum_{j=2}^{m} \rho^{m-j} \right) \frac{S}{D} \;=\; \frac{S}{D}$$

Then we will check the conditions. We can see that $(z_{0,1})$, $(z_{0,2})$ and all $(z_{\leq,j})$ hold trivially. In order to check remaining conditions, we have to calculate the values of $O_1, \ldots, O_m$. We know which dual variables are basic and we use corresponding inequalities, because they are satisfied by equality.

$$O_k \overset{(z_{k,k})}{=} q_k/s_1 \qquad = \tfrac{\rho^{m-k}}{D} \quad \text{for } k = 2, \ldots, m\,,$$

$$\begin{aligned} O_k &\overset{(z_k)}{=} \tfrac{1}{S} \textstyle\sum_{j=1}^{k} q_j \\ &= \tfrac{1}{D} \left(\rho^{m-1} + (1-\rho) \textstyle\sum_{j=2}^{k} \rho^{m-j} \right) = \tfrac{\rho^{m-k}}{D} \quad \text{for } k = 1, \ldots, m\,. \end{aligned}$$

We see that computed values are consistent.

Now every condition $(z_{j,k})$ can be easily verified:

$$\sum_{i=j}^{k} q_i \;=\; \frac{\rho^{m-k-1}}{D}\sum_{i=1}^{k-j+1} \rho^i s_1 \leq^{A^+_{k-j+1}} \frac{\rho^{m-k-1}}{D}\sum_{i=1}^{k-j+1} s_i \;=\; O_k \sum_{i=1}^{k-j+1} s_i \; .$$

At last we have to check that (z_{norm}) holds:

$$\sum_{i=1}^{m} s_i O_{m-i+1} \;=\; \sum_{i=1}^{m} s_i \frac{\rho^{i-1}}{D} \;=\; 1 \; .$$

We proved the lower bound so far. Now we prove the upper bound. We sum up the inequalities multiplied by their respective upper bound coefficients. In order to do that, we have to check that all coefficients but z_{norm} are nonnegative. This is trivial for all $z_{k,k}$ (note that $z_{1,1} = 0$). For z_k we have:

$$\begin{aligned} z_k + z_{k,k} &= \frac{s_{m-k+1}}{D} + \left(1 - \frac{s_1}{S}\right)\sum_{i=2}^{k} \frac{s_{m-k+i}}{D}\rho^{i-2} \\ &= \sum_{i=1}^{k} \frac{s_{m-k+i}}{D}\rho^{i-1} \;=\; \sum_{i=2}^{k+1} \frac{s_{m-k+i-1}}{D}\rho^{i-2} \qquad (2.24) \\ \text{and thus} \quad z_k &= \sum_{i=2}^{k+1} \frac{s_{m-k+i-1}}{D}\rho^{i-2} - z_{k,k} \\ &= \sum_{i=2}^{k} \frac{s_{m-k+i-1} - s_{m-k+i}}{D}\rho^{i-2} + \frac{s_m}{D}\rho^{k-1} \;\geq\; 0 \end{aligned}$$

Moreover (2.24) implies that $z_{k,k} + z_k = z_{k+1,k+1}$ for $k < m$, and $z_{m,m} + z_m = \sum_{i=1}^{m} s_i \rho^{i-1}/D = 1$ Thus the left-hand side of the bounding inequality is equal to:

$$\sum_{j=1}^{m} q_j \left(z_{j,j} + \sum_{k=j}^{m} z_k \right) = \sum_{j=1}^{m} q_j \; .$$

The right-hand side of the inequality is:

$$\begin{aligned} &S z_1 O_1 + \sum_{k=2}^{m} (s_1 z_{k,k} + S z_k) O_k + \left(\sum_{k=1}^{m} s_{m-k+1} O_k - 1 \right) z_{norm} \\ &\quad = \sum_{k=1}^{m} \frac{S s_{m-k+1}}{D} O_k - r \sum_{k=1}^{m} s_{m-k+1} O_k + r \;=\; r \; . \end{aligned}$$

Thus any primal feasible solution has value of the objective function of at most r. So our solution is optimal. Note that the proof of the optimality is in fact showing that described dual solution is feasible and that it indeed gives desired bound.

$m = 3$

The case of three machines splits along the condition A_2. The Case I is already described above. Now we describe the other case:

Case II Ratio: $r = \dfrac{S^2}{\sum\limits_{i=1}^{m}\sum\limits_{j=i}^{m} s_i s_j}$

Conditions:

$(A_2^-\text{:I}) \quad s_2 \leq \rho s_1$

Common denominator: $D = \sum\limits_{i=1}^{m}\sum\limits_{j=i}^{m} s_i s_j$

Nonbasic dual vars:
all but: $z_1, z_2, z_3, z_{2,3}, z_{3,3}$

Jobs:

$q_1 = s_3 S/D$

$q_2 = s_2 S/D$

$q_3 = s_1 S/D$

U.b. coefficients:

$z_{norm} = -r$

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

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

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

The label $(A_2^-\text{:I})$ of the condition says that the condition A_2 separates Case II from Case I and these cases are incident. The cases together cover all valid speed combinations, thus we have:

$$\bar{r}^{online}(s_1, s_2, s_3) = \begin{cases} \frac{S}{s_1+\rho s_2+\rho^2 s_3} & \text{for } s_2 \geq \rho s_1 \\ \frac{S^2}{s_1^2+s_2^2+s_3^2+s_1s_2+s_1s_3+s_2s_3} & \text{for } s_2 \leq \rho s_1 \end{cases}$$

$m = 4$

Here we list the cases for four machines. We omit the Case I again as it was listed before.

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

Conditions:

$(A_2^-\text{: I}) \quad s_2 \leq \rho s_1$

$(B^+\text{: III}) \quad s_2 + s_3 \geq \rho(s_1 + s_2)$

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

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

Jobs:

$q_1 = (s_3+s_4)(S-s_1)/D$

$q_2 = (s_3+s_4)s_1/D$

$q_3 = s_2 S/D$

$q_4 = s_1 S/D$

U.b. coefficients:

$z_{norm} = -r$

$z_1 = z_{2,3} = s_4 S/D$

$z_2 = z_{3,4} = s_3 S/D$

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

$z_4 = (s_1(s_1+s_4) - s_2 s_3 - (1-\rho)(s_3+s_4)s_4)/D$

$z_{4,4} = (s_2 S + (s_3+s_4)s_4)/D$

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

Conditions: (implicit: A_3^-)

$(A_2^-\text{: IV})$ $\quad s_2 \leq \rho s_1$

$(B^-\text{: II})$ $\quad s_2 + s_3 \leq \rho(s_1 + s_2)$

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

Nonbasic dual vars:

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

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:

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

This case is similar to the Case II for three machines. But we failed to extract a nice set of conditions when trying to generalize this case for any number of machines.

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

Conditions: (implicit: B^-)

$(A_2^+\text{: III})$ $\quad s_2 \geq \rho s_1$

$(A_3^-\text{: I})$ $\quad s_2 + s_3 \leq (\rho + \rho^2) s_1$

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

Nonbasic dual vars:

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

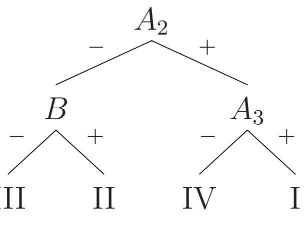
Jobs:

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

U.b. coefficients:

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

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_2 splits the space to the cases I+IV and the cases II+III. Then, I and IV are separated by A_3 and fully cover the halfspace A_2^+. Similarly II and III are separated by B and fully cover the halfspace A_2^-. 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.



2.4.2 Semi-online scheduling

2.4.3 Known sum of processing times, $\sum p_j = P$

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 $s_{n+1}, \ldots, s_m$.

$$\begin{array}{rcllll}
\textbf{maximize} & & R = q_1 + q_2 + \cdots + q_n & & & \\
\textbf{subject to} & & & & & \\
1 & = & s_1 O_n + s_2 O_{n-1} + \cdots + s_n O_1 & (z_{norm}) & & \\
q_1 + \cdots + q_n & \leq & S O_k & (z_k) & 1 \leq k \leq n-1 & \quad (2.25) \\
q_j + \cdots + q_k & \leq & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) & 1 \leq j \leq k \leq n & \\
q_k & \leq & q_{k+1} & (z_{\leq,k}) & 1 \leq k \leq n-1 & \\
0 & \leq & q_1 & (z_0)\ . & &
\end{array}$$

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 $\bar{P} \leq S \cdot T_1$.

$m = 3$

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:

$(A^+\colon \text{II})$ $s_1(s_1 + s_2) \geq s_2 S$

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

Nonbasic dual vars:	U.b. coefficients:
$z_0, z_{\leq,1}, z_{1,1}$	$z_{norm} = -r$
Jobs:	$z_1 = s_2(s_1 + s_2)/D$
$q_1 = s_2 S/D$	$z_{1,2} = s_1 S/D$
$q_2 = s_1 S/D$	

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

Conditions:

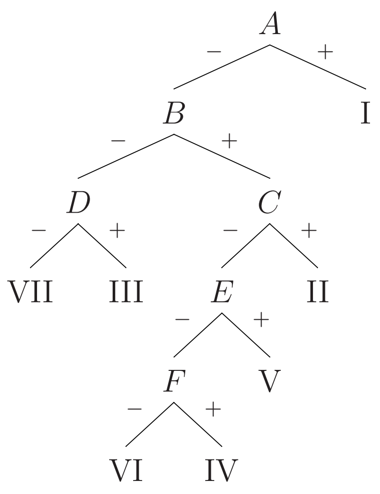
$(A^-\colon \text{I})$ $s_1(s_1 + s_2) \leq s_2 S$

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

Nonbasic dual vars:	U.b. coefficients:
$z_0, z_{\leq,1}, z_1$	$z_{norm} = -r$
Jobs:	$z_{1,2} = s_1 S/D$
$q_1 = s_1 s_2/D$	$z_{1,1} = z_{2,2} = s_2(s_1 + s_2)/D$
$q_2 = s_1^2/D$	

$m = 4$

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: $r = \dfrac{(s_1+s_2+s_3)S}{(s_2+s_3)(s_1+s_2+s_3)+s_1S}$

Conditions:

$(A^+\text{: II,III})\quad (s_1+s_2)(s_1+s_2+s_3) \geq (s_2+s_3)S$

Common denominator: $D = (s_2+s_3)(s_1+s_2+s_3)+s_1S$

Nonbasic dual vars:	U.b. coefficients:
$z_0, z_{\leq,2}, z_{1,1}, z_{1,2}, z_{2,2}, z_{2,3}$	$z_{norm} = -r$
Jobs:	$z_2 = s_2(s_1+s_2+s_3)/D$
$q_1 = q_2 = (s_2+s_3)S/(2D)$	$z_{1,3} = s_1S/D$
$q_3 = s_1S/D$	$z_1 = s_3(s_1+s_2+s_3)/D$

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^+)

$(A^-\text{: I})\quad (s_1+s_2)(s_1+s_2+s_3) \leq (s_2+s_3)S$

$(B^+\text{: III})\quad s_1s_3 \geq s_2^2$

$(C^+\text{: IV,V})\quad s_1(s1+s_2+s_3) \geq s_3S$

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

Nonbasic dual vars:	U.b. coefficients:
$z_0, z_{\leq,1}, z_{\leq,2}, z_2, z_{1,1}, z_{2,2}$	$z_{norm} = -r$
Jobs:	$z_1 = s_3(s_1+s_2)(s_1+s_2+s_3)/D$
$q_1 = s_3(s_1+s_2)S/D$	$z_{1,2} = z_{3,3} = s_2(s_1+s_2+s_3)S/D$
$q_2 = s_2(s_1+s_2)S/D$	$z_{1,3} = s_1^2S/D$
$q_3 = s_1(s_1+s_2)S/D$	

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:

$(A^-\text{: I})\quad (s_1+s_2)(s_1+s_2+s_3) \leq (s_2+s_3)S$

$(B^-\text{: II})\quad s_1s_3 \leq s_2^2$

$(D^+\text{: VII})\quad s_1(s_1+s_2)(s1+s_2+s_3) \geq s_2(s_2+s_3)S$

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

Nonbasic dual vars:	U.b. coefficients:
$z_0, z_{\leq,1}, z_{\leq,2}, z_2, z_{1,1}, z_{2,3}$	$z_{norm} = -r$
Jobs:	$z_1 = s_3(s_1+s_2)(s_1+s_2+s_3)/D$
$q_1 = s_2(s_2+s_3)S/D$	$z_{1,2} = z_{3,3} = s_2(s_1+s_2+s_3)S/D$
$q_2 = s_1(s_2+s_3)S/D$	
$q_3 = s_1(s_1+s_2)S/D$	

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

$$\begin{array}{rrcl}
\text{Conditions:} & \multicolumn{3}{l}{(\text{implicit: } A^-, B^+)} \\
(C^-\text{: II})) & s_1(s_1+s_2+s_3) & \le & s_3 S \\
(E^-\text{: V})) & s_1^3 & \le & s_3(s_1+s_2)^2 \\
(F^+\text{: VI})) & s_1(s_1^2+s_1s_2+s_2^2) & \ge & s_2^2 S
\end{array}$$

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

Nonbasic dual vars:

$z_0, z_{\le,1}, z_{\le,2}, z_2, z_{1,3}, z_{2,2}$

Jobs:

$$\begin{aligned}
q_1 &= s_1(s_1+s_2)^2/D \\
q_2 &= s_2(s_1+s_2)(S-s_1)/D \\
q_3 &= s_1(s_1+s_2)(S-s_1)/D
\end{aligned}$$

U.b. coefficients:

$$\begin{aligned}
z_{norm} &= -r \\
z_1 &= (s_3(s_1+s_2)^2 - s_1^3)/D \\
z_{1,1} = z_{2,3} &= s_1^2 S/D \\
z_{1,2} = z_{3,3} &= s_2(s_1+s_2)S/D
\end{aligned}$$

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

$$\begin{array}{rrcl}
\text{Conditions:} & \multicolumn{3}{l}{(\text{implicit: } A^-)} \\
(B^+\text{: VII}) & s_1s_3 & \ge & s_2^2 \\
(C^-\text{: II}) & s_1(s_1+s_2+s_3) & \le & s_3 S \\
(E^+\text{: IV,VI}) & s_1^3 & \ge & s_3(s_1+s_2)^2
\end{array}$$

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

Nonbasic dual vars:

$z_0, z_{\le,1}, z_{\le,2}, z_1, z_2, z_{2,2}$

Jobs:

$$\begin{aligned}
q_1 &= s_3s_1(s_1+s_2)/D \\
q_2 &= s_2s_1(s_1+s_2)/D \\
q_3 &= s_1^2(s_1+s_2)/D
\end{aligned}$$

U.b. coefficients:

$$\begin{aligned}
z_{norm} &= -r \\
z_{1,1} = z_{2,3} &= s_3(s_1+s_2)(s_1+s_2+s_3)/D \\
z_{1,2} = z_{3,3} &= s_1s_2(s_1+s_2+s_3)/D \\
z_{1,3} &= (s_1^3 - (s_1+s_2)^2 s_3)/D
\end{aligned}$$

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

$$\begin{array}{rrcl}
\text{Conditions:} & \multicolumn{3}{l}{(\text{implicit: } A^-, C^-)} \\
(B^+\text{: VII}) & s_1s_3 & \ge & s_2^2 \\
(E^-\text{: V}) & s_1^3 & \le & s_3(s_1+s_2)^2 \\
(F^-\text{: VI}) & s_1(s_1^2+s_1s_2+s_2^2) & \le & s_2^2 S
\end{array}$$

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

Nonbasic dual vars:

$z_0, z_{\le,1}, z_{\le,2}, z_1, z_2, z_{1,3}$

Jobs:

$$\begin{aligned}
q_1 &= s_1s_2^2/D \\
q_2 &= s_1^2s_2/D \\
q_3 &= s_1^3/D
\end{aligned}$$

U.b. coefficients:

$$\begin{aligned}
z_{norm} &= -r \\
z_{1,1} &= s_3(s_1^2+s_1s_2+s_2^2)/D \\
z_{1,2} &= s_1(s_1^2+s_2^2-s_3(s_1+s_2))/D \\
z_{2,2} &= (s_3(s_1+s_2)^2 - s_1^3)/D \\
z_{2,3} &= s_1(s_1^2 - s_2s_3)/D \\
z_{3,3} &= s_2(s_1s_2+s_1s_3+s_2s_3)/D
\end{aligned}$$

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

$$\begin{array}{rrcl}
\text{Conditions:} & \text{(implicit: } A^-) & & \\
(B^-\text{: V,VI}) & s_1s_3 & \leq & s_2^2 \\
(D^-\text{: III}) & s_1(s_1+s_2)(s1+s_2+s_3) & \leq & s_2(s_2+s_3)S
\end{array}$$

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:

$$\begin{aligned}
q_1 &= s_1s_2(s_2+s_3)/D \\
q_2 &= s_1^2(s_2+s_3)/D \\
q_3 &= s_1^2(s_1+s_2)/D
\end{aligned}$$

U.b. coefficients:

$$\begin{aligned}
z_{norm} &= -r \\
z_{1,1} = z_{2,2} &= s_3(s_1+s_2)(s_1+s_2+s_3)/D \\
z_{1,2} &= s_1(s_2-s_3)(s_1+s_2+s_3)/D \\
z_{1,3} &= s_1(s_1^2-s_2s_3)/D \\
z_{3,3} &= s_1(s_1+s_3)(s_1+s_2+s_3)/D
\end{aligned}$$

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

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 $n \in \{2, \dots, n-1\}$. 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:

$$\begin{array}{rcll}
\textbf{maximize} & & R = p + q_1 + \cdots + q_m & \\
\textbf{subject to} & & & \\
1 & = & s_1O_m + \cdots + s_mO_1 & (z_{norm}) \\
p + q_1 + \cdots + q_k & & & \\
 & \leq & SO_k & (z_k) \quad 1 \leq k \leq m \\
p & \leq & s_1O_k & (z_{k,k}) \quad 1 \leq k \leq m-1 \\
p + q_{j+1} + \cdots + q_k & & & \\
 & \leq & (s_1 + \cdots + s_{k-j+1})O_k & (z_{j,k}) \quad 1 \leq j < k \leq m \\
0 & \leq & q_1 & (z_{0,1}) \\
0 & \leq & q_2 & (z_{0,2}) \\
q_k & \leq & q_{k+1} & (z_{\leq,k}) \quad 2 \leq k \leq m-1 \\
q_m & = & p & \text{(substitution)}
\end{array} \tag{2.26}$$

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:

$$\begin{array}{rcll}
\textbf{maximize} & & R = p + q_2 + \cdots + q_n & \\
\textbf{subject to} & & & \\
1 & = & s_1O_n + \cdots + s_nO_1 & (z_{norm}) \\
p & \leq & s_1O_k & (z_{k,k}) \quad 1 \leq k \leq n-1 \\
p + q_{j+1} + \cdots + q_k & & & \\
 & \leq & (s_1 + \cdots + s_{k-j+1})O_k & (z_{j,k}) \quad 1 \leq j < k \leq n \\
0 & \leq & q_2 & (z_{0,2}) \\
q_k & \leq & q_{k+1} & (z_{\leq,k}) \quad 2 \leq k \leq n-1 \\
q_n & = & p & \text{(substitution)}
\end{array} \tag{2.27}$$

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

$m = 2$

For two machines we need to solve only (2.26) 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:	U.b. coefficients:
all but: $z_1, \dots, z_m, z_{norm}$,	$z_{norm} = -r$
$z_{i,i}$ for $i = 1, \dots, m-1$	$i = 1, \dots, m-2$:
Jobs:	$z_{i,i} = s_{m-i+1} S(S + s_1)/D$
$p = s_1^2 S/D$	$z_{m-1,m-1} = (s_1 s_2 - (S - s_1 - s_2)S)S/D$
$q_1 = s_1(S - s_1)S/D$	$z_{m-1} = s_1(S - s_1)S/D$
$q_2 = \cdots = q_{m-1} = 0$	$z_m = s_1^2(S + s_1)/D$

$n = 2$

We need to solve also (2.27) 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.

$m = 3$

We have already solved $n = 2$, thus it suffices to solve (2.26) 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:

$(A^-\text{:I})$ $s_1 s_2 \leq s_3 S$

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

Nonbasic dual vars:	U.b. coefficients:
$z_{0,1}, z_{0,2}, z_{1,2}, z_{2,2}, z_{2,3}$	$z_{norm} = -r$
Jobs:	$z_{\leq,2} = z_1 = (s_3 S - s_1 s_2)/D$
$p = q_2 = s_1 S/D$	$z_2 = s_2(2s_1 + S)/D$
$q_1 = (S - s_1)S/D$	$z_3 = s_1(2s_1 + S)/D$
	$z_{1,1} = (s_2 + 2s_3)S/D$

$n = 3$

Solutions of (2.27) 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.

$$\begin{tikzcd}
 & B \arrow[dl, dash, "-"'] \arrow[dr, dash, "+"] & & \\
\mathrm{III} & & A \arrow[dl, dash, "-"'] \arrow[dr, dash, "+"] & \\
 & \mathrm{II} & & \mathrm{I}
\end{tikzcd}$$

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:

$$\begin{array}{rrcl}
(A^+\text{:II}) & 2s_1s_3 & \geq & s_2(s_1+s_2) \\
(B^+\text{:III}) & s_1s_2 & \geq & s_3(s_1+s_2)
\end{array}$$

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:

$$\begin{aligned}
p &= s_1(s_1+s_2)^2/D \\
q_2 &= 2s_1s_3(s_1+s_2)/D
\end{aligned}$$

U.b. coefficients:

$$\begin{aligned}
z_{norm} &= -r \\
z_{1,1} &= 2s_3(s_1+s_2)S/D \\
z_{1,2} &= 2s_1s_2(s_1+s_2)S/D \\
z_{1,3} &= (2s_1^2+(s_1+s_2)s_3-s_1s_2) \\
&\quad \cdot (s_1+s_2)/D \\
z_{2,3} &= (s_1s_2-(s_1+s_2)s_3)S/D
\end{aligned}$$

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

Conditions:

$$\begin{array}{rrcl}
(A^-\text{:I}) & 2s_1s_3 & \leq & s_2(s_1+s_2) \\
(B^+\text{:III}) & s_1s_2 & \geq & s_3(s_1+s_2)
\end{array}$$

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

Nonbasic dual vars:

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

Jobs:

$$\begin{aligned}
p &= s_1S/D \\
q_2 &= s_2S/D
\end{aligned}$$

U.b. coefficients:

$$\begin{aligned}
z_{norm} &= -r \\
z_{1,1} &= s_3(2s_1+s_2)S/(s_1D) \\
z_{1,2} &= (s_2+s_3)S/D \\
z_{1,3} &= s_1(2s_1+s_2)/D \\
z_{2,2} &= (s_1s_2-(s_1+s_2)s_3)/(s_1D)
\end{aligned}$$

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:

$(B^-\text{:I,II})$ $s_1s_2 \le s_3(s_1+s_2)$

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

Nonbasic dual vars:	U.b. coefficients:
$z_{0,2}, z_{2,2}, z_{2,3}$	$z_{norm} = -r$
Jobs:	$z_{\le,2} = S(s_3(s_1+s_2)-s_1s_2)/D$
$p = q_2 = s_1(s_1+s_2)S/D$	$z_{1,1} = 3s_3(s_1+s_2)S/D$
	$z_{1,2} = 3s_1s_2S/D$
	$z_{1,3} = 3s_1^2(s_1+s_2)/D$

$m = 4$

Solutions of (2.26) for $m=4$, the Case I is shown in $m=2$. Note that we need to compare this solutions to the solutions of (2.27) 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.

$$\begin{tikzcd}
\text{I} & A \arrow[l, "+"'] \arrow[r, "-"] & \text{IV} & B \arrow[l, "-"'] \arrow[r, "+"] & \text{II} \\
& & C \arrow[u, "-"'] \arrow[d, "+"] & & \\
& & \text{III} & &
\end{tikzcd}$$

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

Conditions:

$(B^+\text{: IV})$ $s_4S \ge s_1(s_2+2s_3)$

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

Nonbasic dual vars:	U.b. coefficients:
$z_{\le,1}, z_{\le,2}, z_{1,2}, z_{1,3}, z_{2,2},$	$z_{norm} = -r$
$z_{2,3}, z_{2,4}, z_{3,3}, z_{3,4}, z_{4,4}$	$z_{\le,3} = s_4S-s_1(s_2+2s_3)/D$
Jobs:	$z_{\le,4} = s_1((s_3+2s_4)S-s_1(2s_2+s_3))/D$
$q_1 = s_1(S-s_1)S/D$	$z_1 = s_1(s_2+2s_3+3s_4)S/D$
$q_2 = q_3 = p = s_1^2S/D$	$z_2 = s_1s_3(S+3s_1)/D$
	$z_3 = s_1s_2(S+3s_1)/D$
	$z_4 = s_1^2(S+3s_1)/D$
	$z_{1,1} = s_1(s_2+2s_3+3s_4)S/D$

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

Conditions:

$$(C^+\text{: IV}) \quad s_1(s_3+s_4) \geq s_2S$$

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:

$$\begin{aligned}
q_1 &= s_1(s_1+s_2)(S-s_1)S/D\\
q_2 &= s_1(s_1(s_3+s_4)-s_2S)S/D\\
q_3 &= p = s_1^2(s_1+s_2)S/D
\end{aligned}$$

U.b. coefficients:

$$\begin{aligned}
z_{norm} &= -r\\
z_{\leq,4} &= s_1(s_4S^2+s_3s_1(s_1+s_2)\\
&\quad - s_2s_1S)/D\\
z_2 &= s_3s_1^2(2S+s_1+s_2)/D\\
z_3 &= s_1(s_4(s_1+s_2)S\\
&\quad + 2s_1(s_2S-s_3(s_1+s_2)))/D\\
z_4 &= s_1^3(2S+s_1+s_2)/D\\
z_{1,1} &= s_4s_1S(2S+s_1+s_2)/D\\
z_{2,3} &= s_1S(s_1(s_2+2s_3)-s_4S)/D
\end{aligned}$$

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

Conditions:

$$\begin{aligned}
(A^-\text{: I}) \quad s_1s_2 &\leq (s_3+s_4)S\\
(B^-\text{: II}) \quad s_4S &\leq s_1(s_2+2s_3)\\
(C^-\text{: III}) \quad s_1(s_3+s_4) &\leq s_2S
\end{aligned}$$

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:

$$\begin{aligned}
q_1 &= s_1(S-s_1)S/D\\
q_2 &= 0\\
q_3 = p &= s_1^2S/D
\end{aligned}$$

U.b. coefficients:

$$\begin{aligned}
z_{norm} &= -r\\
z_{\leq,4} = z_2 &= s_1((s_3+s_4)S-s_1s_2)/D\\
z_3 &= s_2s_1(S+2s_1)/D\\
z_4 &= s_1^2(S+2s_1)/D\\
z_{1,1} &= s_4S(S+2s_1)/D\\
z_{2,2} &= S(s_1(s_2+2s_3)-s_4S)/D
\end{aligned}$$

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 $\epsilon \to 0$.

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

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 $\sum p_j = P$. 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. The linear program for every $n < m$ follows.

$$\begin{array}{rcllr}
\textbf{maximize} & & R = p + q_2 + \cdots + q_n & & \\
\textbf{subject to} & & & & \\
1 & = & s_1 O_n + \cdots + s_1 O_n & (z_{norm}) & \\
(1+\beta)p & \leq & SO_k & (z_k) & 1 \leq k \leq n \\
p & \leq & s_1 O_k & (z_{k,k}) & 1 \leq k \leq n \\
p + q_{j+1} + \cdots + q_k & & & & \\
& \leq & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) & 1 \leq j < k \leq n \\
q_2 + \cdots + q_n & \leq & \beta p & (z_{\leq,0}) & \\
q_n & \leq & p & (z_{\leq,1}) & \\
0 & \leq & q_2 & (z_{\leq,2}) & \\
q_{k-1} & \leq & q_k & (z_{\leq,k}) & 3 \leq k \leq n
\end{array} \tag{2.28}$$

Now we list two cases, where the bound on the competitive ratio is ≤ 1, thus they yield no nontrivial bound.

Case I (A^-) $s_2 + \cdots + s_n \geq \beta s_1$: The bound from Definition 2.3.7 is void:

$$\frac{p + q_2 + \cdots + q_n}{s_1 O_n + \cdots + s_n O_1} \leq \frac{(p + q_2 + \cdots + q_n) s_1}{(s_1 + \cdots + s_n) p} \leq \frac{(1+\beta) p s_1}{(1+\beta) s_1 p} = 1$$

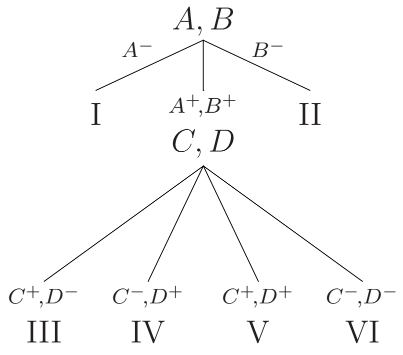
Case II (B^-) $(1+\beta)(s_1 + \cdots + s_n) \geq nS$: We have:

$$O_k = \frac{(1+\beta)p}{S} \geq \frac{pn}{s_1 + \cdots + s_n} \geq \frac{p(k-j+1)}{s_1 + \cdots + s_{k-j+1}} \geq \frac{p + q_{j+1} + \cdots + q_k}{s_1 + \cdots + s_{k-j+1}} .$$

Thus all optima are given by the sum of the processing times and RatioStretch produces the optimal schedule.

$n = 2$

We list the cases of the solution for (2.28). The conditions A^- and B^- are mutually exclusive (e.g., $\beta \geq 1$ separates them). The decision tree containing the cases for all values of parameters is to the right of this text.



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:

$$\begin{array}{rcl}
z_{norm} & = & -r \\
z_{\leq,1} & = & s_2(s_1+s_2)/D \\
z_{1,1} & = & 2s_2(s_1+s_2)/D \\
z_{1,2} & = & 2s_1^2/D
\end{array}$$

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

$$\begin{array}{rrcl}
\text{Conditions:} & \multicolumn{3}{l}{(\text{Implicit } A^+)} \\
(C^-\text{:III}) & (1+\beta)s_1 & \geq & S \\
(D^-\text{:VI}) & \beta & \geq & 1 \\
(B^+\text{:II}) & (1+\beta)(s_1+s_2) & \leq & 2S
\end{array}$$

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

Nonbasic dual vars:	U.b. coefficients:
$z_{\leq,0}, z_2, z_{1,1}, z_{2,2}$	$z_{norm} = -r$
Jobs:	$z_{\leq,1} = (1+\beta)s_2(s_1+s_2)/D$
$p = q_2 = (s_1+s_2)S/D$	$z_1 = 2s_2(s_1+s_2)/D$
	$z_{1,2} = 2s_1S/D$

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

$$\begin{array}{rrcl}
\text{Conditions:} & \multicolumn{3}{l}{(\text{Implicit } B^+)} \\
(A^+\text{:I}) & s_2 & \leq & \beta s_1 \\
(C^+\text{:VI}) & (1+\beta)s_1 & \leq & S \\
(D^+\text{:III}) & \beta & \leq & 1
\end{array}$$

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

Nonbasic dual vars:	U.b. coefficients:
$z_{\leq,1}, z_1, z_2, z_{2,2}$	$z_{norm} = -r$
Jobs:	$z_{\leq,2} = s_2(s_1+s_2)/D$
$p = s_1(s_1+s_2)/D$	$z_{1,1} = (1+\beta)s_2(s_1+s_2)/D$
$q_2 = \beta s_1(s_1+s_2)/D$	$z_{1,2} = (1+\beta)s_1^2/D$

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

$$\begin{array}{rrcl}
\text{Conditions:} & \multicolumn{3}{l}{(\text{Implicit } A^+, B^+)} \\
(C^-\text{:V}) & (1+\beta)s_1 & \geq & S \\
(D^+\text{:IV}) & \beta & \leq & 1
\end{array}$$

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

Nonbasic dual vars:	U.b. coefficients:
$z_{\leq,1}, z_2, z_{1,1}, z_{2,2}$	$z_{norm} = -r$
Jobs:	$z_{\leq,2} = z_1 = s_2(s_1+s_2)/D$
$p = (s_1+s_2)S/((1+\beta)D)$	$z_{1,2} = s_1S/D$
$q_2 = \beta(s_1+s_2)S/((1+\beta)D)$	

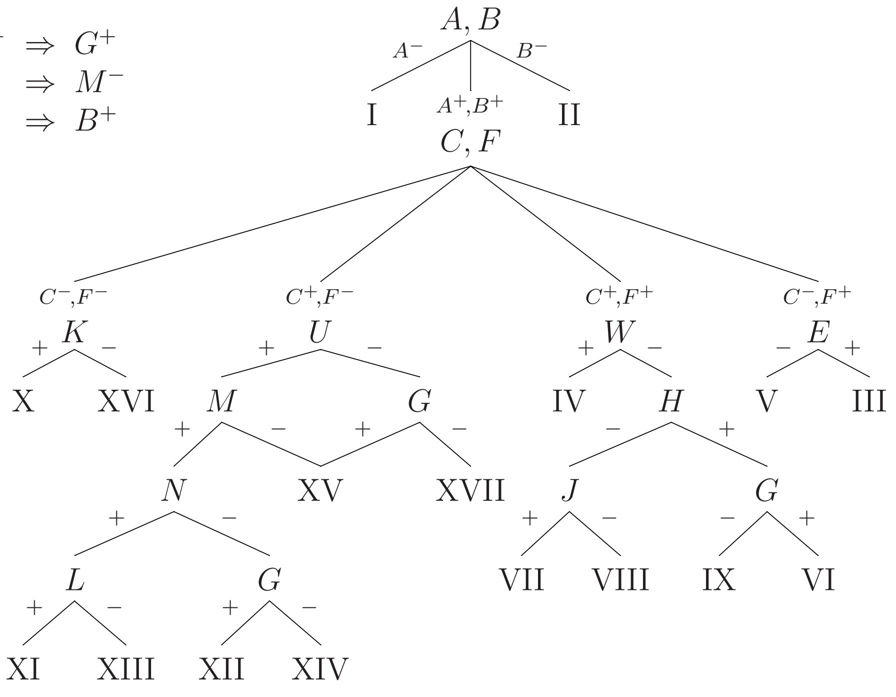
$n = 3$

The case list is rather huge, it contains 17 cases including the two cases that we already seen. We define shortcuts for the conditions first.

$$\begin{aligned}
A &= \beta s_1 - (s_2 + s_3)\\
B &= 3\,S - (1+\beta)\,(s_1 + s_2 + s_3)\\
C &= (\beta - 1)\,(s_2 + s_1) - 2\,s_3\\
E &= {s_2}^2 + s_3 s_2 + s_1 s_2 - (1+\beta)\,s_3 s_1\\
F &= S - (1+\beta)\,s_1\\
G &= 2 - \beta\\
H &= s_3\,(s_2 + s_1) - s_1 s_2\\
J &= 2\,s_1 s_3 - s_2\,(s_2 + s_1)\\
K &= (1+\beta)\,s_3/\,(s_1 + s_2 + s_3) + 1 - (1+\beta)\,(s_2 + s_1)\,/S\\
L &= (1+\beta)\,(s_2 + s_1)^2 - S\,(s_1 + s_2 + 2\,s_3)\\
M &= \beta S - (1+\beta)\,(s_2 + s_1)\\
N &= s_2 S - (1+\beta)\,s_3\,(s_2 + s_1)\\
U &= 2\,S - (1+\beta)\,(s_2 + s_1)\\
W &= s_2 - (\beta - 1)\,s_1
\end{aligned}$$

The condition A^+ denotes $A \geq 0$ and A^- denotes $A \leq 0$. We use similar notation for remaining conditions. We use these letters also in few formulas, to make them shorter. We proceed with the decision tree, together with the list of the valid implications of conditions. This list is needed to verify that the tree is correct. The list of cases comes after the tree.

$$\begin{aligned}
U^+ \;\&\; M^- &\Rightarrow G^+\\
U^- \;\&\; G^+ &\Rightarrow M^-\\
A^- &\Rightarrow B^+
\end{aligned}$$



Case III Ratio: $r = \dfrac{s_1(1+\beta)(s_1+s_2+s_3)}{D}$

$D = (s_2+s_3)(s_1+s_2+s_3) + {s_1}^2(1+\beta)$

Conditions: E^+, A^+, C^-, F^+, (implicit: W^+, G^+, B^+)

Nonbasic dual vars: $z_{\leq,3}, z_{\leq,1}, z_1, z_2, z_3, z_{1,2}, z_{3,3}$

Jobs:
$$\begin{aligned} p &= s_1(s_1+s_2+s_3)/D \\ q_2 &= (1+\beta)s_1s_3/D \\ q_3 &= s_1((1+\beta)(s_1+s_2) - (s_1+s_2+s_3))/D \end{aligned}$$

U.b. coefficients:
$$\begin{aligned} z_{\leq,0} &= (s_2+s_3)(s_1+s_2+s_3)/D \\ z_{1,1} &= (1+\beta)s_3(s_1+s_2+s_3)/D \\ z_{1,3} &= (1+\beta){s_1}^2/D \\ z_{2,2} &= (1+\beta)s_2(s_1+s_2+s_3)/D \end{aligned}$$

Case IV Ratio: $r = \dfrac{s_1(1+\beta)(s_1+s_2+s_3)}{D}$

$D = (s_2+s_3)(s_1+s_2+s_3) + {s_1}^2(1+\beta)$

Conditions: C^+, W^+, F^+, (implicit: A^+, E^+, G^+, B^+)

Nonbasic dual vars: $z_{\leq,3}, z_1, z_2, z_3, z_{1,2}, z_{2,3}, z_{3,3}$

Jobs:
$$\begin{aligned} p = q_3 &= s_1(s_1+s_2+s_3)/D \\ q_2 &= s_1(s_1+s_2+s_3)(\beta-1)/D \end{aligned}$$

U.b. coefficients:
$$\begin{aligned} z_{1,1} &= s_3(1+\beta)(s_1+s_2+s_3)/D \\ z_{2,2} &= s_2(1+\beta)(s_1+s_2+s_3)/D \\ z_{\leq,0} &= (s_2+s_3)(s_1+s_2+s_3)/D \\ z_{1,3} &= {s_1}^2(1+\beta)/D \end{aligned}$$

Case V Ratio: $r = \dfrac{(s_1+s_2)(1+\beta)s_1(s_1+s_2+s_3)}{D}$

$D = (s_1+s_2+s_3)(s_1s_2+s_3s_2+s_1s_3) + s_1(s_1s_2+s_3s_2+{s_1}^2)(1+\beta)$

Conditions: C^-, E^-, F^+, (implicit: A^+, K^+, G^+, B^+)

Nonbasic dual vars: $z_{\leq,3}, z_{\leq,1}, z_1, z_2, z_3, z_{2,2}, z_{3,3}$

Jobs:
$$\begin{aligned} q_3 &= s_1(s_1+s_2)((1+\beta)(s_1+s_2) - (s_1+s_2+s_3))/D \\ p &= s_1(s_1+s_2)(s_1+s_2+s_3)/D \\ q_2 &= s_1s_3(s_1+s_2)(1+\beta)/D \end{aligned}$$

U.b. coefficients:
$$\begin{aligned} z_{1,2} = z_{2,3} &= (1+\beta)s_1s_2(s_1+s_2+s_3)/D \\ z_{\leq,0} &= (s_1+s_2+s_3)(s_1s_2+s_3s_2+s_1s_3)/D \\ z_{1,3} &= s_1(s_1-s_2)(s_1+s_2)(1+\beta)/D \\ z_{1,1} &= s_3(s_1+s_2)(1+\beta)(s_1+s_2+s_3)/D \end{aligned}$$

Case VI Ratio: $r = \dfrac{(s_1+s_2)(1+\beta)\,s_1(s_1+s_2+s_3)}{D}$

$D = (s_1+s_2+s_3)(s_1s_3+s_3s_2+\beta\,s_1s_2)+{s_1}^2(s_1+s_2)(1+\beta)$

Conditions: W^-, H^+, C^+, G^+, F^+, (implicit: A^+, B^+)

Nonbasic dual vars: $z_{\le,3}, z_1, z_2, z_3, z_{2,2}, z_{2,3}, z_{3,3}$

Jobs:
$$\begin{aligned} p = q_3 &= s_1(s_1+s_2)(s_1+s_2+s_3)/D \\ q_2 &= s_1(s_1+s_2)(s_1+s_2+s_3)(\beta-1)/D \end{aligned}$$

U.b. coefficients:
$$\begin{aligned} z_{1,2} = z_{\le,1} &= (1+\beta)\,s_1s_2(s_1+s_2+s_3)/D \\ z_{1,1} &= s_3(s_1+s_2)(1+\beta)(s_1+s_2+s_3)/D \\ z_{1,3} &= {s_1}^2(s_1+s_2)(1+\beta)/D \\ z_{\le,0} &= (s_1+s_2+s_3)\,H/D \end{aligned}$$

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

$D = {s_1}^2(s_1+s_2)+(s_1+s_2)^2(s_1+s_3)+2\,s_1s_3s_2$

Conditions: H^-, J^+, C^+, F^+, (implicit: W^-, A^+, B^+)

Nonbasic dual vars: $z_{\le,3}, z_{\le,0}, z_1, z_2, z_3, z_{2,2}, z_{3,3}$

Jobs:
$$\begin{aligned} q_2 &= 2\,s_1(s_1+s_2)\,s_3/D \\ p = q_3 &= (s_1+s_2)^2\,s_1/D \end{aligned}$$

U.b. coefficients:
$$\begin{aligned} z_{1,1} &= 2\,s_3(s_1+s_2)(s_1+s_2+s_3)/D \\ z_{2,3} &= (s_1+s_2+s_3)(-H)/D \\ z_{1,2} &= 2\,s_1s_2(s_1+s_2+s_3)/D \\ z_{\le,1} &= (s_1+s_2+s_3)(s_1s_2+s_3s_2+s_1s_3)/D \\ z_{1,3} &= (s_1+s_2)(s_3s_2-s_1s_2+2\,{s_1}^2+s_1s_3)/D \end{aligned}$$

Case VIII Ratio: $r = \dfrac{(2\,s_1+s_2)(s_1+s_2+s_3)}{D}$

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

Conditions: H^-, W^-, J^-, F^+, (implicit: C^+, A^+, B^+)

Nonbasic dual vars: $z_{\le,3}, z_{\le,0}, z_1, z_2, z_3, z_{2,3}, z_{3,3}$

Jobs:
$$\begin{aligned} p = q_3 &= s_1(s_1+s_2+s_3)/D \\ q_2 &= s_2(s_1+s_2+s_3)/D \end{aligned}$$

U.b. coefficients:
$$\begin{aligned} z_{\le,1} = z_{1,2} &= (s_2+s_3)(s_1+s_2+s_3)/D \\ z_{2,2} &= (s_1+s_2+s_3)(-H)/(s_1D) \\ z_{1,3} &= s_1(2\,s_1+s_2)/D \\ z_{1,1} &= s_3(2\,s_1+s_2)(s_1+s_2+s_3)/(s_1D) \end{aligned}$$

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

$D = 2s_1s_3s_2 + (s_1+s_2)^2(2s_1+s_3) + (s_1{}^2+s_3{}^2)(s_1+s_2)$

Conditions: G^-, H^+, F^+, (implicit: C^+, W^-, A^+, B^+)

Nonbasic dual vars: $z_{\le,0}, z_1, z_2, z_3, z_{2,2}, z_{2,3}, z_{3,3}$

Jobs: $p = q_2 = q_3 = s_1(s_1+s_2)(s_1+s_2+s_3)/D$

U.b. coefficients:

$$\begin{aligned} z_{\le,3} &= (s_1+s_2+s_3)H/D \\ z_{\le,1} &= (s_1+s_2+s_3)(2s_1s_3+2s_3s_2+s_1s_2)/D \\ z_{1,3} &= 3s_1{}^2(s_1+s_2)/D \\ z_{1,2} &= 3s_1s_2(s_1+s_2+s_3)/D \\ z_{1,1} &= 3s_3(s_1+s_2)(s_1+s_2+s_3)/D \end{aligned}$$

Case X Ratio: $r = \dfrac{(s_1+s_2)(s_1+s_2+s_3)S(1+\beta)}{D}$

$$\begin{aligned} D = {} & s_3(s_1+s_2)(1+\beta)(s_1+s_2+s_3) + \\ & (\beta+2)(s_1+s_2+s_3)s_2S + (1+\beta)(s_1{}^2-s_2{}^2)S \end{aligned}$$

Conditions: F^-, C^-, K^+, (implicit: A^+, E^-, G^+, U^+, B^+)

Nonbasic dual vars: $z_{\le,3}, z_{\le,1}, z_2, z_3, z_{1,1}, z_{2,2}, z_{3,3}$

Jobs:

$$\begin{aligned} p &= (s_1+s_2)(s_1+s_2+s_3)S/D \\ q_2 &= s_3(s_1+s_2)S(1+\beta)/D \\ q_3 &= (s_1+s_2)S(s_1\beta+\beta s_2-s_3)/D \end{aligned}$$

U.b. coefficients:

$$\begin{aligned} z_1 &= s_3(s_1+s_2)(1+\beta)(s_1+s_2+s_3)/D \\ z_{1,3} &= (s_1-s_2)(s_1+s_2)S(1+\beta)/D \\ z_{\le,0} &= (s_1+s_2+s_3)((1+\beta)s_3(s_1+s_2)+s_2S)/D \\ z_{1,2} = z_{2,3} &= S(1+\beta)s_2(s_1+s_2+s_3)/D \end{aligned}$$

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

$D = s_1S + (1+\beta)\left((s_1+s_2+s_3)^2 - s_1s_3\right)$

Conditions: L^+, N^+, U^+, F^-, M^+, (implicit: C^+, A^+, B^+)

Nonbasic dual vars: $z_{\le,3}, z_{\le,0}, z_3, z_{1,1}, z_{2,2}, z_{2,3}, z_{3,3}$

Jobs:

$$\begin{aligned} p = q_3 &= (s_1+s_2+s_3)S/D \\ q_2 &= (s_1+s_2+s_3)((1+\beta)(s_1+s_2)-S)/D \end{aligned}$$

U.b. coefficients:

$$\begin{aligned} z_2 &= (s_1+s_2+s_3)N/(SD) \\ z_1 &= ((1+\beta)(s_1+s_2)+S)s_3(s_1+s_2+s_3)/(SD) \\ z_{1,3} &= s_1((1+\beta)(s_1+s_2)+S)/D \\ z_{\le,1} = z_{1,2} &= (s_2+s_3)(s_1+s_2+s_3)(1+\beta)/D \end{aligned}$$

Case XII Ratio: $r = \dfrac{(s_1+s_2)(s_1+s_2+s_3)\,S\,(1+\beta)}{D}$

$D = \beta\, s_2 (s_1+s_2+s_3)\, S + (1+\beta)(s_1+s_2)(s_1 S + s_3 (s_1+s_2+s_3))$

Conditions: N^-, F^-, M^+, C^+, G^+, (implicit: A^+, U^+, B^+)

Nonbasic dual vars: $z_{\le,3}, z_2, z_3, z_{1,1}, z_{2,2}, z_{2,3}, z_{3,3}$

Jobs:
$$\begin{aligned} p = q_3 &= (s_1+s_2)(s_1+s_2+s_3)\,S/D \\ q_2 &= (s_1+s_2)(s_1+s_2+s_3)\,S\,(\beta-1)/D \end{aligned}$$

U.b. coefficients:
$$\begin{aligned} z_{1,2} &= S(1+\beta)\,s_2 (s_1+s_2+s_3)/D \\ z_1 &= s_3 (s_1+s_2)(1+\beta)(s_1+s_2+s_3)/D \\ z_{1,3} &= s_1 (s_1+s_2)\,S\,(1+\beta)/D \\ z_{\le,1} &= S(1+\beta)\,s_2 (s_1+s_2+s_3)/D \\ z_{\le,0} &= (s_1+s_2+s_3)(-N)/D \end{aligned}$$

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

$D = s_3 (s_1+s_2)^2 (1+\beta) + S\,(2\, s_3 s_2 + (s_1+s_2)(2\, s_1 + s_2))$

Conditions: N^+, F^-, L^-, C^+, (implicit: A^+, K^+, M^+, U^+, B^+)

Nonbasic dual vars: $z_{\le,3}, z_{\le,0}, z_2, z_3, z_{1,1}, z_{2,2}, z_{3,3}$

Jobs:
$$\begin{aligned} q_2 &= 2\, s_3 (s_1+s_2)\,S/D \\ p = q_3 &= (s_1+s_2)^2\,S/D \end{aligned}$$

U.b. coefficients:
$$\begin{aligned} z_{2,3} &= (s_1+s_2+s_3)\,N/D \\ z_{\le,1} &= (s_1+s_2+s_3)((1+\beta)\,s_3 (s_1+s_2) + s_2 S)/D \\ z_{1,3} &= (s_1+s_2)((1+\beta)\,s_3 (s_1+s_2) + (2\, s_1 - s_2)\,S)/D \\ z_{1,2} &= 2\, s_2 (s_1+s_2+s_3)\,S/D \\ z_1 &= 2\, s_3 (s_1+s_2)(s_1+s_2+s_3)/D \end{aligned}$$

Case XIV Ratio: $r = \dfrac{3\,(s_1+s_2)(s_1+s_2+s_3)\,S}{D}$

$D = s_3 (s_1+s_2)(1+\beta)(s_1+s_2+s_3) + 3\, S s_1 (s_1+s_2) + 2\, s_2 (s_1+s_2+s_3)\, S$

Conditions: N^-, U^+, F^-, G^-, (implicit: C^+, A^+, M^+, B^+)

Nonbasic dual vars: $z_{\le,0}, z_2, z_3, z_{1,1}, z_{2,2}, z_{2,3}, z_{3,3}$

Jobs: $p = q_2 = q_3 = (s_1+s_2)(s_1+s_2+s_3)\,S/D$

U.b. coefficients:
$$\begin{aligned} z_1 &= 3\, s_3 (s_1+s_2)(s_1+s_2+s_3)/D \\ z_{\le,3} &= (s_1+s_2+s_3)(-N)/D \\ z_{1,3} &= 3\, S s_1 (s_1+s_2)/D \\ z_{\le,1} &= (s_1+s_2+s_3)((2+2\beta) s_3 (s_1+s_2) + s_2 S)/D \\ z_{1,2} &= 3\, s_2 (s_1+s_2+s_3)\,S/D \end{aligned}$$

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

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

Conditions: M^-, F^-, C^+, G^+, (implicit: A^+, K^-, B^+)

Nonbasic dual vars: $z_{\le,3}, z_3, z_{1,1}, z_{1,2}, z_{2,2}, z_{2,3}, z_{3,3}$

Jobs:
$$\begin{aligned} p = q_3 &= (s_1+s_2+s_3)\,S/\left((1+\beta)\,D\right) \\ q_2 &= (s_1+s_2+s_3)\,S\,(\beta-1)/\left((1+\beta)\,D\right) \\ q_3 &= (s_1+s_2+s_3)\,S/\left((1+\beta)\,D\right) \end{aligned}$$

U.b. coefficients:
$$\begin{aligned} z_1 &= s_3\,(s_1+s_2+s_3)/D \\ z_{1,3} &= s_1 S/D \\ z_{\le,0} &= (s_2+s_3)(s_1+s_2+s_3)/D \\ z_2 &= s_2\,(s_1+s_2+s_3)/D \end{aligned}$$

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

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

Conditions: F^-, C^-, K^-, (implicit: A^+, G^+, M^-, B^+, U^+)

Nonbasic dual vars: $z_{\le,3}, z_{\le,1}, z_3, z_{1,1}, z_{1,2}, z_{2,2}, z_{3,3}$

Jobs:
$$\begin{aligned} q_2 &= s_3 S/D \\ p &= (s_1+s_2+s_3)\,S/\left((1+\beta)\,D\right) \\ q_3 &= S\,(\beta\,(s_1+s_2)-s_3)/\left((1+\beta)\,D\right) \end{aligned}$$

U.b. coefficients:
$$\begin{aligned} z_1 &= s_3\,(s_1+s_2+s_3)/D \\ z_{\le,0} &= (s_2+s_3)(s_1+s_2+s_3)/D \\ z_2 &= s_2\,(s_1+s_2+s_3)/D \\ z_{1,3} &= s_1 S/D \end{aligned}$$

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

$D = (s_2+s_3)(s_1+s_2+s_3)(1+\beta) + 3\,s_1 S$

Conditions: B^+, G^-, U^-, (implicit: C^+, A^+, F^-, L^+)

Nonbasic dual vars: $z_{\le,0}, z_3, z_{1,1}, z_{1,2}, z_{2,2}, z_{2,3}, z_{3,3}$

Jobs: $p = q_2 = q_3 = (s_1+s_2+s_3)\,S/D$

U.b. coefficients:
$$\begin{aligned} 2\,z_{\le,3} = z_{\le,1} &= 2\,(s_2+s_3)(s_1+s_2+s_3)(1+\beta)/D \\ z_{1,3} &= 3\,s_1 S/D \\ z_1 = z_2 &= 3\,s_2\,(s_1+s_2+s_3)/D \end{aligned}$$

2.4.6 Approximately known optimum, $T \leq C^*_{\max} \leq \alpha T$

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.

$$\begin{array}{rcllr}
\textbf{maximize} & & R = q_1 + \cdots + q_m & & \\
\textbf{subject to} & & & & \\
1 & = & s_1O_m + s_2O_{m-1} + \cdots + s_mO_1 & (z_{norm}) & \\
q_1 + \cdots + q_k & \leq & (s_1 + \cdots + s_m)O_k & (z_k) \quad 1 \leq k \leq m & \\
q_j + \cdots + q_k & \leq & (s_1 + \cdots + s_{k-j+1})O_k & (z_{j,k}) \quad 2 \leq j \leq k \leq m & \\
q_j & \leq & q_{j+1} & (z_{\leq,j}) \quad 2 \leq j \leq m-1 & (2.29)\\
0 & \leq & q_1 & (z_{0,1}) & \\
0 & \leq & q_2 & (z_{0,2}) & \\
O_k & \leq & O_{k+1} & (z_{\beta,k}) \quad 1 \leq k \leq m-1 & \\
O_m & \leq & \beta O_1 & (z_\beta) &
\end{array}$$

And for $n < m$:

$$\begin{array}{rcllr}
\textbf{maximize} & & R = q_1 + \cdots + q_n & & \\
\textbf{subject to} & & & & \\
1 & = & s_1O_n + s_2O_{n-1} + \cdots + s_nO_1 & (z_{norm}) & \\
q_j + \cdots + q_k & \leq & (s_1 + \cdots + s_{k-j+1})O_k & (z_{j,k}) \quad 1 \leq j \leq k \leq n & \\
q_j & \leq & q_{j+1} & (z_{\leq,j}) \quad 1 \leq j \leq n-1 & (2.30)\\
0 & \leq & q_1 & (z_0) & \\
O_k & \leq & O_{k+1} & (z_{\beta,k}) \quad 1 \leq k \leq n-1 & \\
O_n & \leq & \beta O_1 & (z_\beta) &
\end{array}$$

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.

$m = 2$

For two machines, we need to solve only (2.29) 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: $r \;=\; \dfrac{(s_1+s_2)^2}{s_1^2+s_1s_2+s_2^2}$

Conditions:

$(A^+\text{:II})\quad \beta s_2 \;\geq\; s_1+s_2$

Common denominator: $D \;=\; s_1^2+s_1s_2+s_2^2$

Nonbasic dual vars:	U.b. coefficients:
$z_{0,1}, z_{0,2}, z_{\beta,1}, z_\beta$	$z_{norm} \;=\; -r$
Jobs:	$z_1 \;=\; z_{2,2} \;=\; s_2(s_1+s_2)/D$
$q_1 \;=\; s_1(s_1+s_2)/D$	$z_2 \;=\; s_1^2/D$
$q_2 \;=\; s_2(s_1+s_2)/D$	

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

Conditions:

$(A^-\text{:I})\quad \beta s_2 \;\leq\; s_1+s_2$

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

Nonbasic dual vars:	U.b. coefficients:
$z_{0,1}, z_{0,2}, z_{\beta,1}, z_1$	$z_{norm} \;=\; -r$
Jobs:	$z_\beta \;=\; s_2(s_1+s_2)/D$
$q_1 \;=\; \beta s_2/D$	$z_2 \;=\; \beta(s_1+s_2)/D$
$q_2 \;=\; \beta s_1/D$	

$n = 2$

Here we solve (2.30) 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:

$(A^+\text{:II})\quad \beta s_2 \;\geq\; s_1$

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

Nonbasic dual vars:	U.b. coefficients:
$z_0, z_{\leq,1}, z_\beta, z_{\beta,1}$	$z_{norm} \;=\; -r$
Jobs:	$z_{1,1} \;=\; z_{2,2} \;=\; s_2(s_1+s_2)/D$
$q_1 \;=\; s_1s_2/D$	$z_{1,2} \;=\; s_1(s_1+s_2)/D$
$q_2 \;=\; s_1^2/D$	

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

Conditions:

$(A^-\text{:I})$ $\beta s_2 \le s_1$

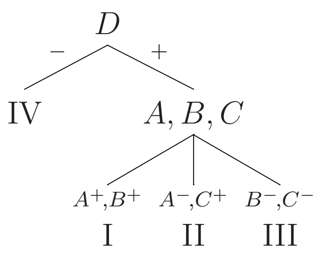
Common denominator: $D = \beta s_1 + s_2$

Nonbasic dual vars:	U.b. coefficients:
$z_0, z_{\le,1}, z_{\beta,1}, z_{1,1}$	$z_{norm} = -r$
Jobs:	$z_\beta = s_2(s_1+s_2)/D$
$q_1 = s_1/D$	$z_{1,2} = \beta(s_1+s_2)/D$
$q_2 = (\beta(s_1+s_2)-s_1)/D$	

$m = 3$

In this case we need to take maximum of (2.29) for $m = 3$ and (2.30) 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: $r = \dfrac{S}{s_1 + \rho s_2 + \rho^2 s_3}$

Conditions: (Implicit D^+)

$(A^+\text{:II})$ $s_2 \ge \rho s_1$

$(B^+\text{:III})$ $\beta\rho^2 \ge 1$

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

Nonbasic dual vars:	U.b. coefficients:
$z_0, z_{\le,2}, z_\beta, z_{\beta,1}, z_{\beta,2}, z_{2,3}$	$z_{norm} = -r$
Jobs:	$z_1 = z_{2,2} = s_3/D$
$q_1 = \rho^2 S/D$	$z_2 = (\rho s_1 + (2-\rho)s_2)/D$
$q_2 = \rho s_1/D$	$z_3 = (1-\rho)^2(S+s_3)/D$
$q_3 = s_1/D$	$z_{3,3} = (s_2 + \rho s_3)/D$

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) $s_2 \le \rho s_1$

(C^+:III) $\beta s_3 \ge S$

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:

$z_0, z_{\le,2}, z_\beta, z_{\beta,1}, z_{\beta,2}, z_{2,2}$

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: $r = \dfrac{\beta S}{\beta s_1 + \beta\rho s_2 + s_3}$

Conditions:

(B^-:I) $\beta\rho^2 \le 1$

(C^-:II) $\beta s_3 \le S$

(D^+:IV) $\beta\rho \ge 1$

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

Nonbasic dual vars:

$z_0, z_{\le,2}, z_{\beta,1}, z_{\beta,2}, z_1, z_{2,3}$

Jobs:

$q_1 = S/D$

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

$q_3 = \beta s_1/D$

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) $\beta\rho \le 1$

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

Nonbasic dual vars:

$z_0, z_{\le,2}, z_{\beta,2}, z_{2,2}, z_{2,3}, z_{3,3}$

Jobs:

$q_1 = S/D$

$q_2 = 0$

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

U.b. coefficients:

$z_{norm} = -r$

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

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

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

2.4.7 Tightly grouped processing times, $p \le p_j \le \alpha p$

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:

$$\begin{array}{rcll} \textbf{maximize} & R = q_1 + q_2 + \cdots + q_n & \text{for: } n \leq m \ \ (\text{or } n \leq 2m-2) & \\ \textbf{subject to} & & & \\ 1 & = & s_1 O_n + s_2 O_{n-1} + \cdots + s_n O_1 & (z_{norm}) \\ q_j + \cdots + q_k & \leq & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) \quad 1 \leq j \leq k \leq n \\ q_k & \leq & q_{k+1} & (z_{\leq,k}) \quad 1 \leq k \leq n-1 \\ q_n & \leq & \beta q_1 & (z_\beta)\ . \end{array} \qquad (2.31)$$

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.

$$\begin{array}{rcll} \textbf{maximize} & R = q_1 + q_2 + \cdots + q_m & \text{for: } m+1 \leq n \leq 2m-2 & \\ \textbf{subject to} & & & \\ 1 & = & s_1 O_m + s_2 O_{m-1} + \cdots + s_m O_1 & (z_{norm}) \\ q_1 + \cdots + q_k & \leq & (s_1 + \cdots + s_{k+n-m}) O_k & (z_{1,k}) \quad 1 \leq k \leq m \\ q_j + \cdots + q_k & \leq & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) \quad 2 \leq j \leq k \leq m \\ q_1 & \leq & (n-m+1) q_2 & (z_{\leq,1}) \\ q_k & \leq & q_{k+1} & (z_{\leq,k}) \quad 2 \leq k \leq n-1 \\ (n-m+1) q_m & \leq & \beta q_1 & (z_\beta)\ . \end{array} \qquad (2.32)$$

And at last we have to solve the case where $n \geq 2m-1$. 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 (2.33) with additional conditions (2.34):

$$\begin{array}{rcll} \textbf{maximize} & R = q_1 + q_2 + \cdots + q_m & \text{for: } n \geq 2m+1 & \\ \textbf{subject to} & & & \\ 1 & = & s_1 O_m + s_2 O_{m-1} + \cdots + s_m O_1 & (z_{norm}) \\ q_1 + \cdots + q_k & \leq & (s_1 + \cdots + s_m) O_k & (z_k) \quad 1 \leq k \leq m \\ q_j + \cdots + q_k & \leq & (s_1 + \cdots + s_{k-j+1}) O_k & (z_{j,k}) \quad 2 \leq j \leq k \leq m \\ q_j & \leq & q_{j+1} & (z_{\leq,j}) \quad 2 \leq j \leq m-1 \\ q_m & \leq & \beta q_2 & (z_{\beta,2}) \\ m q_m & \leq & \beta q_1 & (z_{\beta,1}) \end{array} \qquad (2.33)$$

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

$$\begin{array}{rcll} (N+1) q_m & \leq & \beta q_1 & (z_{N+1,\beta}) \\ q_1 & \leq & N q_2 & (z_{N,2}) \end{array} \qquad (2.34)$$

The conditions $(z_{N,\beta})$ and $(z_{N,2})$ defines the feasibility for q_1 consisting of N jobs. The formulation of (2.34) 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 $q_m \leq \beta q_2$ and $\beta > 1$.

Solution for fixed $s_1, s_2, \ldots, s_m$. This program can be solved efficiently in two steps, having fixed set of speeds $s_1, s_2, \ldots, s_m$. First we solve the (2.33). Let the result be $\bar{x}^*$. The $\bar{x}^*$ is either feasible to (2.33)+(2.34) 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 (2.33)+(2.34) satisfies either $(z_{\bar{N}^*,2})$ or $(z_{\bar{N}^*+1,\beta})$ by equality. Thus we solve two more linear programs, namely (2.33)+$(z_{\bar{N}^*,2})$ and (2.33)+$(z_{\bar{N}^*+1,\beta})$. The maximum of these two solutions is the desired solution of (2.33)+(2.34).

$m = 2$

We will solve the case of $m = 2$ parametrically now. We use an equivalent (and more natural) reformulation of the (2.34):

There is integer $N \geq m$, for which both following conditions holds:

$$\begin{array}{rcll} Nq_m & \leq & \beta q_1 & (z_{N,\beta}) \\ q_1 & \leq & Nq_2 & (z_{N,2}) \end{array} \qquad (2.35)$$

We solve the (2.33)+(2.35) 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:	U.b. coefficients:
$z_{2,2}, z_{\beta,1}, z_{\beta,2}, z_{<,1}$	$z_{norm} = -r$
Jobs:	$z_1 = (\beta + N)s_2/D$
$q_1 = NS/D$	$z_2 = (\beta + N)s_1/D$
$q_2 = \beta S/D$	$z_{N,\beta} = s_2/D$

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:	U.b. coefficients:
$z_{N,\beta}, z_{\beta,1}, z_{\beta,2}, z_{<,1}$	$z_{norm} = -r$
Jobs:	$z_1 = s_2 S/D$
$q_1 = s_2 S/D$	$z_2 = z_{2,2} = s_1^2/D$
$q_2 = s_1 S/D$	

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.

$n = 2$

We also have to solve (2.31) 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:	U.b. coefficients:
$z_{2,2}, z_{\leq,1}$	$z_{norm} = -r$
Jobs:	$z_{1,1} = (\beta+1)s_2 S/D$
$q_1 = s_1 S/D$	$z_{1,2} = (\beta+1)s_1^2/D$
$q_2 = \beta s_1 S/D$	$z_\beta = s_2 S/D$

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:	U.b. coefficients:
$z_\beta, z_{\leq,1}$	$z_{norm} = -r$
Jobs:	$z_{1,1} = z_{2,2} = s_2 S/D$
$q_1 = s_2 s_1/D$	$z_{1,2} = s_1(s_1 - s_2)/D$
$q_2 = s_1^2/D$	

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 $p \leq p_j \leq \alpha p$. 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.

$$R^{p\le p_j\le \alpha p}(s_1,s_2,\beta)=\begin{cases}\dfrac{S^2}{s_1^2+s_2^2+s_1s_2} & \text{for } 2s_1\le \beta s_2\\[2ex] \dfrac{(\beta+2)S}{\beta s_1+2S} & \text{for } \left\{\begin{aligned}2s_1&\ge\beta s_2\\ s_1&\le\beta s_2\\ 2(s_1-s_2)&\le\beta s_2\end{aligned}\right.\\[2ex] \dfrac{(\beta+2)S}{\beta s_1+2S} & \text{for } \left\{\begin{aligned}s_1&\ge\beta s_2\\ \beta s_1-2s_2&\le\beta s_2\end{aligned}\right.\\[2ex] \dfrac{s_1S}{s_1^2+s_2^2} & \text{for } \left\{\begin{aligned}2s_1&\ge\beta s_2\\ s_1&\le\beta s_2\\ 2(s_1-s_2)&\ge\beta s_2\end{aligned}\right.\\[2ex] \dfrac{(\beta+1)s_1(s_1+s_2)}{(\beta+1)s_1^2+s_2S} & \text{for } \left\{\begin{aligned}s_1&\ge\beta s_2\\ \beta s_1-2s_2&\ge\beta s_2\end{aligned}\right.\end{cases}$$

2.4.8 Techniques for solving the Parametrized LPs

The solutions above were obtained by searching 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 hand. 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 $\{\min \mathbf{y}^T\mathbf{b} \mid \mathbf{y}^T A = \mathbf{c}^T, \mathbf{y} \geq \mathbf{0}\}$ 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 [Vaz01], 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 $\mathbf{y}^T A = \mathbf{c}^T$ & $\mathbf{y}^T(I - Y) = 0$. 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.

At this point we did our best considering 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 neighbors 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).

Chapter 3

Lower bound on deterministic algorithms for related machines without preemption.

In this section we provide a new lower bound on deterministic nonpreemptive online scheduling on related machines. This lower bound is inspired by [BCK97], although we use an analytical bound on maximal frequency of scheduling jobs instead of the combinatorial bound obtained by computer based search through the graph of possible states of an algorithm.

We will prove that $1 \leq \int_0^1 \frac{\ln(R)}{-\ln(1-R^{-x})} dx$. After solving the inequality numerically, we obtain a lower bound on competitive ratio of $R > 2.564$. The previous bound was 2.438 [BCK97].

Now we are considering the model where preemtions are not allowed, thus every job J_j has to be scheduled in one timeslot of length p_j/s_i, where s_i is the speed of used machine.

We use a geometric sequence of jobs and a geometric sequence of machines, both sequences having the same ratio. First, we consider how the algorithm behaves on one of the machines and we upper bound the amortized frequency of scheduling a job on this machine. This amortized frequency is a function of the competitive ratio, the common ratio of the geometric sequence and the speed of the machine. Then we take the sum of these frequencies over all machines. Any real algorithm has to schedule one job in one step, thus this sum has to be at least 1. This gives our bound on competitive ratio, as the common ratio of the geometric sequence approaches to 1.

3.1 Used model

We start the numbering of the machines as well as of the jobs by 0, to get rid of unnecessary -1 in the definition of our input sequence. Thus we have machines $M_0, M_1, \ldots, M_{m-1}$ and jobs $J_0, J_1, \ldots, J_{n-1}$.

There are no preemptions allowed, every single job can be scheduled to a single machine. The job J_j takes p_j/s_i time if scheduled on machine M_i. We consider one-by-one online scheduling again. As we measure the makespan, we need not to be aware of start times of jobs on a machine in this model, as any schedule can be trivially converted to the schedule without idle time (gaps), while not increasing the makespan. Let $\mathcal{J}_i = \{J \in \mathcal{J} \mid \sigma^{\mathsf{A}}(J) = M_i\}$ be the jobs scheduled to machine M_i. The completion time of the machine is then simply the sum of processing times of the jobs scheduled to the machine divided by its speed: $C_i^{\mathsf{A}} = \frac{1}{s_i}\sum_{j:J_j\in\mathcal{J}_i} p_j$.

3.2 Lower bound

Our lower bound is proved by the geometric sequence of jobs: $p_i = \alpha^i$. They are scheduled to the geometric sequence of machines: $s_i = \alpha^{-i}$. Both sequences have the same length, i.e., $n = m$. Thus the optimal schedule after step t is to schedule the jobs to the machines in reverse order, i.e., the job J_j to the machine M_{t-j}. The value of the optimal makespan is then $C^*_{\max}(\mathcal{J}[t]) = p_t = \alpha^t$.

We define t_i to be the minimum amortized period of scheduling a job to the machine M_i of speed α^{-i} with respect to the claimed competitive ratio R. We declare the value of t_i and prove it in the next theorem:

Lemma 3.2.1 *Suppose that algorithm* A *has competitive ratio* R*. Let* $\mathcal{J}$ *be the geometric sequence of* n *jobs with the ratio* α*, i.e.,* $p_j = \alpha^j$*. Let* $\mathcal{M}$ *be the geometric sequence of* n *machines with the ratio* α^{-1}*, i.e.,* $s_i = \alpha^{-i}$*. Let*

$$t_i \;=\; \log_\alpha \frac{R}{R-\alpha^i} \qquad \textit{for } s_i = \alpha^{-i} > R^{-1} \tag{3.1}$$

Then for any fixed α *and any fixed* n *the algorithm* A *cannot schedule more than* $\frac{n}{t_i} + Rs_i + 1$ *jobs from the input sequence on any machine* M_i *with speed* $s_i = \alpha^{-i} > R^{-1}$*. Moreover the algorithm can schedule at most one job on the machine with speed* R^{-1} *and no job on slower machines.*

Proof: If $s_i < R^{-1}$, then no job can be scheduled the machine M_i. This is because our input sequence is defined so that the optimal makespan is equal to the size of the job on input, i.e., $C^*_{\max}[\mathcal{J}[j]] = p_j$. Moreover if $s_i = R^{-1}$ then there can be scheduled only one job. The same reason as before now requires that no job is scheduled on M_i before scheduling any job. We assume $s_i > R^{-1}$ from now on.

Let $\mathcal{J}_i = (J'_1, J'_2, \ldots, J'_{n_i})$ be the jobs scheduled on the machine M_i. We can bound the size of p'_j by $\sum_{k=1}^{j} p'_j \le Rs_ip'_j$ because the algorithm is R-competitive and the optimal makespan is p'_j after scheduling the job J'_j. This yields:

$$p'_j \;\ge\; \max\left\{\frac{\sum_{k=1}^{j-1} p'_k}{Rs_i - 1}, 1\right\} \qquad \text{for } j = 1, 2, \ldots, n_i\,.$$

We define a sequence $(q_j)_{j=1}^{n_i}$ of lower bounds on the processing times from $\mathcal{J}_i$ recursively:

$$q_j \;=\; \max\left\{\frac{\sum_{k=1}^{j-1} q_k}{Rs_i - 1}, 1\right\} \qquad \text{for } j = 1, 2, \ldots, n_i\,.$$

We can see that $q_j \le p'_j$, as $q_1 = 1 \le p'_1$ and $\sum_{k=1}^{j-1} q_j \le \sum_{k=1}^{j-1} p'_j$ using the induction over j.

The following fact holds trivially by the definition of q_j:

$$q_j > 1 \quad \text{implies} \quad \log_\alpha \frac{q_{j+1}}{q_j} \;=\; \log_\alpha \frac{Rs_i}{Rs_i - 1} \;=\; t_i\,. \tag{3.2}$$

Now we bound n_i. We know that $q_{n_i} \le p'_{n_i} \le p_n = \alpha^n$. Using (3.2) we have at most $n/t_i + 1$ numbers of size $q_j > 1$ in $q_1, \ldots, q_{n_i}$. We also have that $\sum_{j=1}^{\lfloor Rs_i \rfloor} q_j \ge \lfloor Rs_i \rfloor$, thus $q_j > 1$ for any $j > Rs_i$. This gives that there are no more than $n/t_i + 1 + Rs_i$ jobs scheduled to the machine M_i by an R-competitive algorithm. □

Theorem 3.2.2 *If there is an R-competitive algorithm for deterministic nonpreemptive scheduling on related machines, then following inequality holds:*

$$1 \le \int_0^1 \frac{\ln(R)}{-\ln(1 - R^{-x})} dx\,. \tag{3.3}$$

This gives $R > 2.564$.

Proof: The algorithm has to schedule all jobs, thus from Lemma 3.2.1 follows:

$$n \;=\; \sum_{i=0}^{n-1} n_i \;=\; \sum_{i=0}^{\lfloor \log_\alpha R \rfloor} n_i \;\le\; (R+1)\lfloor \log_\alpha R \rfloor + \sum_{i=0}^{\lfloor \log_\alpha R \rfloor} \frac{n}{t_i}\;.$$

As n may be as large as the adversary want, we can omit the term $(R+1)\lfloor \log_\alpha R \rfloor$ and we get:

$$\begin{aligned}
1 \;&\le\; \sum_{i=0}^{\lfloor \log_\alpha R \rfloor} \frac{1}{t_i} \;=\; \sum_{i=0}^{\left\lfloor \frac{\ln R}{\ln \alpha} \right\rfloor} \frac{\ln \alpha}{-\ln(1 - \alpha^i R^{-1})} \\
&\le\; \int_{-1}^{\frac{\ln R}{\ln \alpha}} \frac{\ln \alpha}{-\ln(1 - \alpha^i R^{-1})} di \\
&=\; \int_{-\frac{\ln \alpha}{\ln R}}^{1} \frac{\ln R}{-\ln(1 - R^{y-1})} dy && (3.4) \\
&\xrightarrow{\alpha \to 1}\; \int_0^1 \frac{\ln R}{-\ln(1 - R^{y-1})} dy\;. && (3.5)
\end{aligned}$$

In the second inequality we simply bound the sum by the appropriate integral. We use the fact that $\frac{1}{t_i}$ can be viewed as a continuous function in i, moreover this function is decreasing. We substitute $i = y\frac{\ln R}{\ln \alpha}$ to get (3.4). After substituting $x = 1 - y$ in (3.5) we get exactly (3.3).

The inner function of the integral is a bounded monotone function. So we can solve the integration numerically and we get the threshold of $R \approx 2.5649877$. □

We were also curious about how large sequence we would need to prove the bound of $R' = R - \epsilon$. We denote the expression (3.4) by $F_\alpha(R)$:

$$F_\alpha(R) \; = \; \int_{-\frac{\ln \alpha}{\ln R}}^{1} \frac{\ln R}{-\ln(1 - R^{y-1})} dy$$

Let R_α be the bound that is obtained from the inequality $1 \leq F_\alpha(R_\alpha)$, that is $R_\alpha = \inf \{R \mid F_\alpha(R) \geq 1\}$. Then for any $R' < R$ we have $\epsilon' > 0$, such that $R' + \epsilon' < R$, then we have $\alpha > 1$, such that $R' + \epsilon' \leq R_\alpha$, and at last we have n, such that $F_\alpha(R_\alpha - \epsilon') \leq 1 - (R' + 1)\lfloor \log_\alpha R' \rfloor / n$. This suffices to obtain that no algorithm can be R'-competitive on the geometric input sequence of the jobs $p_0 = 1, p_1 = \alpha, \ldots, p_n = \alpha^n$ and the machines $s_0 = 1, s_1 = \alpha^{-1}, \ldots, s_n = \alpha^{-n}$. We used a computer to do a numerical check that $R_\alpha > R - 3(\alpha - 1)$ and $F_\alpha(R_\alpha - \epsilon') < 1 - \frac{\epsilon'}{2}$. The first inequality says that $\alpha = 1 + \epsilon/4$ and $\epsilon' = \epsilon/4$ suffices to provide $R' + \epsilon' \leq R_\alpha$. The second inequality hints the choice of n, we want to obtain $(R+1)\lfloor \log_\alpha R \rfloor \leq n\frac{\epsilon'}{2}$. We can see that $n = \Omega\left(\frac{1}{\epsilon' \ln(\alpha)}\right) = \Omega\left(\frac{1}{\epsilon \ln(1+\epsilon)}\right) = \Omega(\epsilon^{-2})$ is big enough. We can prove asymptotic version of the inequalities that we checked numerically. We use that $c_1 = \frac{\partial}{\partial \alpha} F_\alpha(R')$ and $c_2 = \frac{\partial}{\partial R'} F_\alpha(R')$ and $1/c_2$ are bounded numbers for α near 1 and R' near 2.56. Then we obtain $F_\alpha(R_\alpha - \epsilon') \approx 1 - \epsilon' c_2$ and $R_\alpha \approx R - (\alpha - 1)\, c_1/c_2$.

Bibliography

[Alb02] Susanne Albers. On randomized online scheduling. In *Proc. 34th Symp. Theory of Computing (STOC)*, pages 134–143. ACM, 2002.

[BCK97] Piotr Berman, Moses Charikar, and Marek Karpinski. On-line load balancing for related machines. In *Proc. 5th Workshop on Algorithms and Data Structures (WADS)*, volume 1272 of *Lecture Notes in Comput. Sci.*, pages 116–125. Springer, 1997.

[BCK00] Piotr Berman, Moses Charikar, and Marek Karpinski. On-line load balancing for related machines. *J. Algorithms*, 35:108–121, 2000.

[BFKV95] Yair Bartal, Amos Fiat, Howard Karloff, and Rakesh Vohra. New algorithms for an ancient scheduling problem. *J. Comput. Systems Sci.*, 51:359–366, 1995.

[CvVW94a] Bo Chen, Andre van Vliet, and Gerhard J. Woeginger. Lower bounds for randomized online scheduling. *Inform. Process. Lett.*, 51:219–222, 1994.

[CvVW94b] Bo Chen, Andre van Vliet, and Gerhard J. Woeginger. New lower and upper bounds for on-line scheduling. *Oper. Res. Lett.*, 16:221–230, 1994.

[CvVW95] Bo Chen, Andre van Vliet, and Gerhard J. Woeginger. An optimal algorithm for preemptive on-line scheduling. *Oper. Res. Lett.*, 18:127–131, 1995.

[DE09] György Dósa and Leah Epstein. Preemptive online scheduling with reordering. In *Proc. 17th European Symp. on Algorithms (ESA)*, volume 5757 of *Lecture Notes in Comput. Sci.*, pages 456–467. Springer, 2009.

[DH04] György Dósa and Yong He. Semi-online algorithms for parallel machine scheduling problems. *Computing*, 72:355–363, 2004.

[Du04] Donglei Du. Optimal preemptive semi-online scheduling on two uniform processors. *Inform. Process. Lett.*, 92:219–223, 2004.

[Ebe10] Tomáš Ebenlendr. Semi-online preemptive scheduling: Study of special cases. In *Proc. 8th Int. Conf. on Parallel Processing and Applied Mathematics (PPAM 2009)*, volume 6068 of *Lecture Notes in Comput. Sci.*, pages 11–20. Springer, 2010.

[EF02a] Leah Epstein and Lene M. Favrholdt. Optimal non-preemptive semi-online scheduling on two related machines. In *Proc. 27th Symp. on Mathematical Foundations of Computer Science (MFCS)*, volume 2420 of *Lecture Notes in Comput. Sci.*, pages 245–256. Springer, 2002.

[EF02b] Leah Epstein and Lene M. Favrholdt. Optimal preemptive semi-online scheduling to minimize makespan on two related machines. *Oper. Res. Lett.*, 30:269–275, 2002.

[EJS06] Tomáš Ebenlendr, Wojciech Jawor, and Jiří Sgall. Preemptive online scheduling: Optimal algorithms for all speeds. In *Proc. 14th European Symp. on Algorithms (ESA)*, volume 4168 of *Lecture Notes in Comput. Sci.*, pages 327–339. Springer, 2006.

[EJS09] Tomáš Ebenlendr, Wojciech Jawor, and Jiří Sgall. Preemptive online scheduling: Optimal algorithms for all speeds. *Algorithmica*, 53:504–522, 2009.

[ENS+01] Leah Epstein, John Noga, Steven S. Seiden, Jiří Sgall, and Gerhard J. Woeginger. Randomized on-line scheduling for two uniform machines. *J. Sched.*, 4:71–92, 2001.

[EÖW08] Matthias Englert, Deniz Özmen, and Matthias Westermann. The power of reordering for online minimum makespan scheduling. In *Proc. 49th Symp. Foundations of Computer Science (FOCS)*, pages 603–612. IEEE, 2008.

[Eps01] Leah Epstein. Optimal preemptive scheduling on uniform processors with non-decreasing speed ratios. *Oper. Res. Lett.*, 29:93–98, 2001.

[Eps03] Leah Epstein. Bin stretching revisited. *Acta Inform.*, 39:97–117, 2003.

[ES00] Leah Epstein and Jiří Sgall. A lower bound for on-line scheduling on uniformly related machines. *Oper. Res. Lett.*, 26:17–22, 2000.

[ES04] Tomáš Ebenlendr and Jiří Sgall. Optimal and online preemptive scheduling on uniformly related machines. In *Proc. 21st Symp. on Theoretical Aspects of Computer Science (STACS)*, volume 2996 of *Lecture Notes in Comput. Sci.*, pages 199–210. Springer, 2004.

[ES09a] Tomáš Ebenlendr and Jiří Sgall. Optimal and online preemptive scheduling on uniformly related machines. *J. Sched.*, 12:517–527, 2009.

[ES09b] Tomáš Ebenlendr and Jiří Sgall. Semi-online preemptive scheduling: Online algorithm for all variants. In *Proc. 26th Symp. on Theoretical Aspects of Computer Science (STACS)*, volume 3 of *LIPIcs*, pages 346–360. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.

[ES10] Tomáš Ebenlendr and Jiří Sgall. Semi-online preemptive scheduling: Online algorithm for all variants. *Theory Comput. Syst.*, pages 1–37, 2010.

[EY07] Leah Epstein and Deshi Ye. Semi-online scheduling with "end of sequence" information. *J. Comb. Optim.*, 14:45–61, 2007.

[FKT89] Ulrich Faigle, Walter Kern, and Gyorgy Turán. On the performane of online algorithms for partition problems. *Acta Cybernet.*, 9:107–119, 1989.

[FW00] Rudolph Fleischer and Michaela Wahl. On-line scheduling revisited. *J. Sched.*, 3:343–353, 2000.

[Gra66] Ronald L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical J.*, 45:1563–1581, 1966.

[Gra69] Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.*, 17:263–269, 1969.

[GS78] Teofilo F. Gonzales and Sartaj Sahni. Preemptive scheduling of uniform processor systems. *J. ACM*, 25:92–101, 1978.

[HJ04a] Yong He and Yiwei Jiang. Optimal algorithms for semi-online preemptive scheduling problems on two uniform machines. *Acta Inform.*, 40:367–383, 2004.

[HJ04b] Yong He and Yiwei Jiang. Preemptive semi-online scheduling with tightly-grouped processing times. *J. Comput. Sci. Technol.*, 19:733–739, 2004.

[HLS77] E. Horwath, E. C. Lam, and R. Sethi. A level algorithm for preemptive scheduling. *J. ACM*, 24:32–43, 1977.

[HS87] Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *J. ACM*, 34:144–162, 1987.

[HS88] Dorit S. Hochbaum and David Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM J. Comput.*, 17, 1988.

[HZ99] Yong He and Guochuan Zhang. Semi on-line scheduling on two identical machines. *Computing*, 62:179–187, 1999.

[JH07] Yiwei Jiang and Yong He. Optimal semi-online algorithms for preemptive scheduling problems with inexact partial information. *Acta Inform.*, 44:571–590, 2007.

[KKST97] 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.

[McN59] R. McNaughton. Scheduling with deadlines and loss functions. *Management Sci.*, 6:1–12, 1959.

[RC03] John F. Rudin, III and R. Chandrasekaran. Improved bound for the online scheduling problem. *SIAM J. Comput.*, 32:717–735, 2003.

[Rud01] John F. Rudin, III. *Improved Bound for the Online Scheduling Problem.* PhD thesis, The University of Texas at Dallas, 2001.

[Sei00] Steve S. Seiden. Randomized online multiprocessor scheduling. *Algorithmica*, 28:173–216, 2000.

[Sga97] Jiří Sgall. A lower bound for randomized on-line multiprocessor scheduling. *Inform. Process. Lett.*, 63:51–55, 1997.

[SSW00] Steve Seiden, Jiří Sgall, and Gerhard J. Woeginger. Semi-online scheduling with decreasing job sizes. *Oper. Res. Lett.*, 27:215–221, 2000.

[TH02] Zhiyi Tan and Yong He. Semi-on-line problems on two identical machines with combined partial information. *Oper. Res. Lett.*, 30:408–414, 2002.

[TH07] Zhiyi Tan and Yong He. Semi-online scheduling problems on two identical machines with inexact partial information. *Theoret. Comput. Sci.*, 377:110–125, 2007.

[Tic04] Tomáš Tichý. Randomized on-line scheduling on 3 processors. *Oper. Res. Lett.*, 32:152–158, 2004.

[Vaz01] Vijay V. Vazirany. *Approximation algorithms.* Springer, 2001.

[WD98] Jianjun Wen and Donglei Du. Preemptive on-line scheduling for two uniform processors. *Oper. Res. Lett.*, 23:113–116, 1998.

[ZY99] Guochuan Zhang and Deshi Ye. A note on on-line scheduling with partial information. *Computers & Mathematics with Applications*, 44:539–543, 1999.

Appendix A

The numerical lower bounds

A.1 Online scheduling

In Table A.1 we list the set of speeds s_i and the sequence of values q_j that witness a lower bound on competitive ratio of 2.112, using linear program (2.7). The values O_k can be computed directly from q_j and s_i.

A file with the solutions for $m = 2, \ldots, 200$ in a format suitable for computer verification is attached in the electronic version of the thesis.

A.2 Known maximal processing time.

Table A.2 shows the set of speeds s_i and the sequence of values q_j that witness a lower bound on competitive ratio of 1.908, using linear program (2.11).

We also attach a file with the solutions for $m = 2, \ldots, 200$ in a format suitable for computer verification in the electronic version of the thesis.

$R > 2.11256$	$q_1 = 0.0947507$	$q_{98} = \cdots = q_{102} = 0.0038075$
	$q_2 = 0.0010607$	$q_{103} = \cdots = q_{107} = 0.0039374$
$m = 200$	$q_3 = 0.0011012$	$q_{108} = \cdots = q_{111} = 0.0040901$
	$q_4 = 0.0011139$	$q_{112} = 0.0041628$
$s_1 = 1.0000000$	$q_5 = 0.0011669$	$q_{113} = 0.0043201$
$s_2 = 0.7746337$	$q_6 = 0.0011810$	$q_{114} = 0.0044834$
$s_3 = 0.6676802$	$q_7 = q_8 = 0.0011949$	$q_{115} = q_{116} = q_{117} = 0.0045524$
$s_4 = 0.5969363$	$q_9 = q_{10} = 0.0012687$	$q_{118} = 0.0046542$
$s_5 = 0.5112385$	$q_{11} = 0.0012841$	$q_{119} = 0.0048408$
$s_6 = 0.4290230$	$q_{12} = 0.0013503$	$q_{120} = \cdots = q_{123} = 0.0050348$
$s_7 = 0.3600290$	$q_{13} = 0.0013664$	$q_{124} = 0.0051044$
$s_8 = 0.3021304$	$q_{14} = q_{15} = 0.0013835$	$q_{125} = 0.0053043$
$s_9 = s_{10} = 0.2805526$	$q_{16} = q_{17} = 0.0014746$	$q_{126} = 0.0055119$
$s_{11} = s_{12} = 0.2603720$	$q_{18} = 0.0015462$	$q_{127} = q_{128} = q_{129} = 0.0055659$
$s_{13} = 0.2287875$	$q_{19} = 0.0015967$	$q_{130} = 0.0057009$
$s_{14} = 0.2010344$	$q_{20} = q_{21} = 0.0016186$	$q_{131} = 0.0059383$
$s_{15} = s_{16} = s_{17} = 0.1900257$	$q_{22} = 0.0016405$	$q_{132} = \cdots = q_{135} = 0.0061856$
$s_{18} = s_{19} = 0.1839019$	$q_{23} = 0.0017927$	$q_{136} = 0.0062397$
$s_{20} = 0.1758213$	$q_{24} = 0.0018091$	$q_{137} = 0.0065516$
$s_{21} = 0.1577303$	$q_{25} = 0.0018187$	$q_{138} = 0.0068792$
$s_{22} = \cdots = s_{27} = 0.1415007$	$q_{26} = 0.0018351$	$q_{139} = q_{140} = q_{141} = 0.0068893$
$s_{28} = 0.1377890$	$q_{27} = 0.0019418$	$q_{142} = 0.0070104$
$s_{29} = 0.1250557$	$q_{28} = 0.0020344$	$q_{143} = 0.0074272$
$s_{30} = \cdots = s_{36} = 0.1153880$	$q_{29} = 0.0020618$	$q_{144} = \cdots = q_{149} = 0.0078689$
$s_{37} = 0.1131405$	$q_{30} = 0.0020656$	$q_{150} = 0.0081527$
$s_{38} = \cdots = s_{44} = 0.1042002$	$q_{31} = 0.0020931$	$q_{151} = 0.0087345$
$s_{45} = 0.0973724$	$q_{32} = 0.0020969$	$q_{152} = \cdots = q_{158} = 0.0093470$
$s_{46} = \cdots = s_{51} = 0.0939827$	$q_{33} = 0.0021243$	$q_{159} = 0.0101489$
$s_{52} = 0.0887078$	$q_{34} = 0.0021281$	$q_{160} = 0.0110197$
$s_{53} = s_{54} = s_{55} = 0.0871756$	$q_{35} = 0.0021984$	$q_{161} = \cdots = q_{167} = 0.0112386$
$s_{56} = 0.0870474$	$q_{36} = 0.0022244$	$q_{168} = 0.0121802$
$s_{57} = 0.0829030$	$q_{37} = q_{38} = q_{39} = 0.0022627$	$q_{169} = 0.0134204$
$s_{58} = \cdots = s_{61} = 0.0821842$	$q_{40} = q_{41} = 0.0023058$	$q_{170} = 0.0147869$
$s_{62} = 0.0788989$	$q_{42} = q_{43} = q_{44} = 0.0023502$	$q_{171} = \cdots = q_{176} = 0.0151853$
$s_{63} = s_{64} = s_{65} = 0.0770295$	$q_{45} = q_{46} = q_{47} = 0.0023948$	$q_{177} = 0.0169269$
$s_{66} = 0.0762824$	$q_{48} = q_{49} = q_{50} = 0.0024396$	$q_{178} = 0.0188684$
$s_{67} = 0.0734089$	$q_{51} = q_{52} = q_{53} = 0.0024845$	$q_{179} = 0.0210325$
$s_{68} = \cdots = s_{71} = 0.0724077$	$q_{54} = q_{55} = q_{56} = 0.0025296$	$q_{180} = q_{181} = 0.0219992$
$s_{72} = 0.0696173$	$q_{57} = \cdots = q_{60} = 0.0025748$	$q_{182} = q_{183} = q_{184} = 0.0227317$
$s_{73} = s_{74} = s_{75} = 0.0680943$	$q_{61} = q_{62} = q_{63} = 0.0026267$	$q_{185} = 0.0240486$
$s_{76} = 0.0670618$	$q_{64} = q_{65} = q_{66} = 0.0026858$	$q_{186} = 0.0273686$
$s_{77} = 0.0646201$	$q_{67} = \cdots = q_{70} = 0.0027531$	$q_{187} = 0.0311469$
$s_{78} = \cdots = s_{95} = 0.0634905$	$q_{71} = \cdots = q_{74} = 0.0028244$	$q_{188} = q_{189} = 0.0354468$
$s_{96} = 0.0621875$	$q_{75} = \cdots = q_{78} = 0.0028963$	$q_{190} = q_{191} = 0.0381941$
$s_{97} = 0.0606841$	$q_{79} = \cdots = q_{82} = 0.0029715$	$q_{192} = 0.0411317$
$s_{98} = 0.0593376$	$q_{83} = 0.0030473$	$q_{193} = 0.0490140$
$s_{99} = 0.0592250$	$q_{84} = 0.0031228$	$q_{194} = 0.0584068$
$s_{100} = \cdots = s_{200} = 0.0550208$	$q_{85} = 0.0032047$	$q_{195} = 0.0695995$
	$q_{86} = 0.0032864$	$q_{196} = 0.0829372$
	$q_{87} = 0.0033610$	$q_{197} = 0.0968398$
	$q_{88} = 0.0034442$	$q_{198} = 0.1083164$
	$q_{89} = 0.0035296$	$q_{199} = 0.1256673$
	$q_{90} = q_{91} = q_{92} = 0.0036035$	$q_{200} = 0.1622280$
	$q_{93} = \cdots = q_{97} = 0.0036971$	

Table A.1: This sequence proves the lower bound for the online scheduling problem.

$R > 1.90844$	$m = 200$	$p = 0.0262381$
$s_1 = 1.0000000$	$q_1 = 0.2144790$	$q_{116} = q_{117} = 0.0066507$
$s_2 = \cdots = s_{30} = 0.0932937$	$q_2 = \cdots = q_{31} = 0.0017624$	$q_{118} = 0.0067819$
$s_{31} = 0.0923561$	$q_{32} = 0.0020982$	$q_{119} = 0.0069072$
$s_{32} = 0.0908911$	$q_{33} = 0.0024423$	$q_{120} = 0.0070292$
$s_{33} = 0.0894493$	$q_{34} = 0.0026849$	$q_{121} = 0.0071739$
$s_{34} = 0.0880304$	$q_{35} = 0.0026988$	$q_{122} = 0.0073215$
$s_{35} = 0.0866850$	$q_{36} = 0.0027361$	$q_{123} = 0.0074551$
$s_{36} = 0.0853100$	$q_{37} = 0.0027703$	$q_{124} = 0.0075571$
$s_{37} = 0.0840503$	$q_{38} = 0.0028833$	$q_{125} = 0.0077018$
$s_{38} = 0.0827658$	$q_{39} = 0.0031002$	$q_{126} = 0.0078565$
$s_{39} = 0.0814044$	$q_{40} = 0.0031025$	$q_{127} = 0.0080069$
$s_{40} = 0.0801131$	$q_{41} = 0.0031405$	$q_{128} = 0.0081364$
$s_{41} = 0.0789302$	$q_{42} = 0.0031696$	$q_{129} = 0.0082768$
$s_{42} = 0.0777239$	$q_{43} = 0.0032036$	$q_{130} = 0.0084153$
$s_{43} = 0.0764454$	$q_{44} = 0.0032202$	$q_{131} = 0.0085459$
$s_{44} = 0.0751484$	$q_{45} = 0.0032646$	$q_{132} = 0.0087007$
$s_{45} = 0.0739564$	$q_{46} = 0.0033055$	$q_{133} = 0.0088718$
$s_{46} = 0.0728644$	$q_{47} = 0.0033089$	$q_{134} = 0.0090376$
$s_{47} = 0.0717508$	$q_{48} = 0.0033531$	$q_{135} = 0.0091613$
$s_{48} = 0.0705706$	$q_{49} = 0.0033785$	$q_{136} = 0.0093367$
$s_{49} = 0.0693733$	$q_{50} = 0.0033888$	$q_{137} = 0.0095155$
$s_{50} = 0.0681637$	$q_{51} = 0.0034335$	$q_{138} = 0.0096694$
$s_{51} = 0.0670920$	$q_{52} = 0.0036358$	$q_{139} = 0.0098363$
$s_{52} = 0.0660823$	$q_{53} = q_{54} = 0.0039484$	$q_{140} = 0.0100008$
$s_{53} = 0.0649061$	$q_{55} = q_{56} = 0.0040020$	$q_{141} = 0.0101560$
$s_{54} = 0.0639142$	$q_{57} = q_{58} = 0.0040568$	$q_{142} = 0.0103400$
$s_{55} = 0.0628628$	$q_{59} = 0.0041126$	$q_{143} = 0.0105333$
$s_{56} = 0.0617963$	$q_{60} = 0.0041476$	$q_{144} = 0.0106943$
$s_{57} = 0.0608128$	$q_{61} = 0.0041690$	$q_{145} = 0.0108651$
$s_{58} = 0.0596702$	$q_{62} = 0.0042266$	$q_{146} = 0.0110579$
$s_{59} = 0.0588647$	$q_{63} = 0.0042795$	$q_{147} = 0.0112488$
$s_{60} = 0.0577844$	$q_{64} = 0.0042839$	$q_{148} = 0.0114369$
$s_{61} = 0.0567559$	$q_{65} = q_{66} = 0.0043411$	$q_{149} = 0.0116144$
$s_{62} = 0.0558886$	$q_{67} = q_{68} = 0.0043984$	$q_{150} = 0.0118266$
$s_{63} = 0.0549692$	$q_{69} = q_{70} = q_{71} = 0.0044557$	$q_{151} = 0.0120038$
$s_{64} = 0.0540366$	$q_{72} = \cdots = q_{75} = 0.0045132$	$q_{152} = 0.0121973$
$s_{65} = 0.0531766$	$q_{76} = q_{77} = q_{78} = 0.0045713$	$q_{153} = 0.0124137$
$s_{66} = 0.0521775$	$q_{79} = \cdots = q_{82} = 0.0046296$	$q_{154} = 0.0126280$
$s_{67} = 0.0511971$	$q_{83} = \cdots = q_{86} = 0.0046886$	$q_{155} = 0.0128392$
$s_{68} = 0.0505061$	$q_{87} = \cdots = q_{90} = 0.0047481$	$q_{156} = 0.0130385$
$s_{69} = 0.0496015$	$q_{91} = \cdots = q_{94} = 0.0048082$	$q_{157} = 0.0132339$
$s_{70} = 0.0486011$	$q_{95} = \cdots = q_{98} = 0.0048689$	$q_{158} = 0.0134472$
$s_{71} = 0.0477580$	$q_{99} = q_{100} = q_{101} = 0.0049303$	$q_{159} = 0.0136793$
$s_{72} = 0.0470281$	$q_{102} = 0.0050276$	$q_{160} = 0.0139080$
$s_{73} = 0.0462545$	$q_{103} = 0.0051416$	$q_{161} = 0.0141239$
$s_{74} = 0.0455774$	$q_{104} = 0.0052366$	$q_{162} = 0.0143355$
$s_{75} = 0.0446384$	$q_{105} = 0.0053210$	$q_{163} = 0.0145666$
$s_{76} = 0.0439054$	$q_{106} = 0.0054386$	$q_{164} = 0.0148102$
$s_{77} = 0.0432176$	$q_{107} = 0.0055589$	$q_{165} = 0.0150401$
$s_{78} = 0.0423053$	$q_{108} = 0.0056818$	$q_{166} = 0.0152825$
$s_{79} = 0.0414123$	$q_{109} = 0.0057422$	$q_{167} = 0.0155289$
$s_{80} = 0.0409761$	$q_{110} = 0.0058661$	$q_{168} = 0.0157792$
$s_{81} = 0.0400899$	$q_{111} = 0.0059926$	$q_{169} = 0.0160335$
$s_{82} = 0.0392228$	$q_{112} = 0.0061218$	$q_{170} = 0.0162544$
$s_{83} = 0.0386007$	$q_{113} = 0.0062192$	$q_{171} = 0.0164476$
$s_{84} = 0.0379005$	$q_{114} = 0.0063230$	$q_{172} = \cdots = q_{184} = 0.0166941$
$s_{85} = \cdots = s_{200} = 0.0371670$	$q_{115} = 0.0064561$	$q_{185} = \cdots = q_{200} = 0.0262381$

Table A.2: This sequence proves the lower bound for the known maximal processing time semi-online scheduling problem.