Scheduling When You Don't Know the Number of Machines*

Clifford Stein† Mingxian Zhong‡

Abstract

Often in a scheduling problem, there is uncertainty about the jobs to be processed. The issue of uncertainty regarding the machines has been much less studied. In this paper, we study a scheduling environment in which jobs first need to be grouped into some sets before the number of machines is known, and then the sets need to be scheduled on machines without being separated. In order to evaluate algorithms in such an environment, we introduce the idea of an α-robust algorithm, one which is guaranteed to return a schedule on any number m of machines that is within an α factor of the optimal schedule on m machine, where the optimum is not subject to the restriction that the sets cannot be separated. Under such environment, we give a $(\frac{5}{3}+\epsilon)$-robust algorithm for scheduling on parallel machines to minimize makespan, and show a lower bound $\frac{4}{3}$. For the special case when the jobs are infinitesimal, we give a 1.233-robust algorithm with an asymptotic lower bound of 1.207. We also study a case of fair allocation, where the objective is to minimize the difference between the maximum and minimum machine load.

1 Introduction

For many problems, one does not know the entire input accurately and completely in advance. There are different ways of addressing such uncertainty, e.g. via online algorithms (assuming the input arrives over time), dynamic algorithms (assuming the input changes over time), stochastic optimization (assuming the input includes random variables) or robust optimization (assuming that there is bounded uncertainty in the data). Another way of addressing uncertainty is to require one solution that is good against all possible values of the uncertain parameters. Examples of work in this direction include the universal traveling salesman problem (one tour that is good no matter which subset of points arrive) [15], robust matchings (one matching is chosen and then evaluated by its top k edges, where k is unknown) [9, 13], a knapsack of unknown capacity(one policy of packing that is good irrespective of the actual capacity) [6] and 2-stage scheduling (some decisions must be made before the actual scenario is known) [5, 17]. In scheduling problems, there are many ways to model uncertainty in the jobs, including online algorithms [1, 2], in which the set of jobs is not known in advance, stochastic scheduling [12], in which the jobs are modeled as random variables, and work on schedules that are good against multiple objective functions [4, 14, 16]. But there is much less work studying the possibility of uncertainty in the machines, and the work we are aware of studies uncertainty in speed or reliability (breakdowns) [3, 7].

Motivated by the need to understand how to make scheduling decisions without knowing how many machines we will have, we consider a different notion of uncertainty – a scenario in which you don't know how many machines you are going to have, but you still have to commit (partially) to a schedule by making significant decisions about partitioning the jobs before knowing the number of machines.

This type of decision arises in a variety of settings. For example, many scheduling problems are fundamentally about packing items onto machines and there are many examples of problems that concern packing items where there are multiple levels of commitment to be made with partial information. For example, in a warehouse, a large order may need to be placed into multiple boxes, without knowing exactly how many trucks there will be to ship the items. You therefore want to be able to pack the items well, given the various possible number of trucks. Another example involves problems in modern data centers. In data centers, there are some systems which require you to group work together into "bundles" without knowing exactly how many machines will be available. For example, in a map-reduce type computation, the mapping function naturally breaks the data into some number of groups g. However, there are some unknown number of available machines m, and you typically have to design your mapping function, choosing a g and associated grouping, without knowing m. You may know a range of possible values for m, or it may vary widely depending on the availability of machines at the time you run the map-reduce computation (and the availability is typically not under your control). As more and more computing moves to the "cloud", that is, moves to large shared data centers,

*Supported in part by NSF grants CCF-1421161 and CCF-1714818.

†Columbia University, cliff@ieor.columbia.edu

‡Columbia University, mz2325@columbia.edu.

we anticipate that this problem of grouping work without knowing the number of machines will become more widespread.

In this paper, we consider one of the simplest scheduling problems – minimizing makespan on identical parallel machines. We choose this problem partly as a proof of concept for our two stage model. We consider the following specific model. We are given a set of n jobs, J, with a known processing times $p(j)$ for each job j, and a number M, which is an upper bound on the number of machines we might have. An algorithm must commit, before knowing how many machines there are, to grouping the jobs into M *bags*, where each job is assigned to exactly one of the bags. We call this step the *packing* step. Only after completing the packing step do we learn the number of machines m. We now need to compute a schedule, with the restriction that we must keep the bags together, that is, we will assign one or more bags to each machine. We call this step the *scheduling* step. As in other robust problems, we want to do well against all possible numbers of machines. We therefore evaluate our schedule by the ratio of the makespan of our schedule, $ALG(m, M)$, to the makespan of a schedule that knew m in advance, opt_m, taking the worst case over all possible values of m. If an algorithm always provides a ratio of at most α, where $\alpha = \max_{1 \leq m \leq M} \frac{ALG(m,M)}{\text{opt}_m}$, we call it α-robust. (We may also consider scenarios in which there are different upper and lower bounds on the range of m; the definition of robustness extends in the obvious way.)

Our main result is an algorithm for minimizing makespan on parallel machines. which is $(\frac{5}{3}+\epsilon)$-robust; and we show a lower bound of $4/3$ on the robustness of any algorithm for minimizing makespan on parallel machines. As with many scheduling problems, there are two different aspects to address. One is the load-balancing aspect, but in this two-stage problem, it seems that one wants to create bags of a variety of different sizes, in order to allow a more balanced final schedule. The second is to deal with large jobs, and to handle cases where one or several large jobs are the dominant term in the makespan. Large jobs seem to provide a particular challenge in this problem, and much of our algorithm and analysis are devoted to handling various cases involving large jobs.

In order to focus on the load balancing issues, we consider the "continuous" case where we have a set of infinitesimal jobs. That is, in the packing stage, we simply need to divide our total load into M bags. In traditional makespan scheduling, this case is trivial, we would just divide the load into M equal pieces and achieve an optimal makespan. But in this two stage-problem, even the continuous case is challenging. **We can, however, obtain significantly stronger results than in the discrete case, showing an upper bound of** 1.233 **and a lower bound of** 1.207 **on the robustness.**

The continuous case also models a problem in *fair allocation*. In fair allocation, you typically have resources that you want to split "fairly" among several parties. The literature on this problem is vast and we will not attempt to summarize it here. We will only observe that we are solving a problem in fair allocation that has not previously been studied, to our knowledge. We are given some objects to share, and everyone agrees on the values, but we don't know how many people will be sharing them. We place the objects into bags, and have the restriction that each person must take a subset of the bags. In the makespan variant, we are minimizing the maximum amount that anyone gets. Motivated by fairness, we also consider a version where you want to minimize the difference between the maximum allocation and the minimum allocation (this objective makes sense in fair allocation, but not necessarily in scheduling). Here we consider a case where we know a lower bound of αM on the eventual number of machines, and can show a lower bound of $\min\{2/3, 2/(4\alpha+1)\}A$ on the difference and we can obtain an upper bound of $\min\{2/3, 1/(\alpha+1)\}A$, where A is the average load.

1.1 Overview of the Paper and a Lower Bound

We give a brief overview of our paper, and for intuition, a simple lower bound. The order of our paper is different than the order presented in the introduction. We present the continuous case first, because the proofs are simpler and it gives some intuition for the discrete case.

In Section 2, we consider the case when all jobs are infinitesimal. We give an algorithm which is approximately 1.233 robust. And we also show a lower bound which is approximately 1.207. For the infinitesimal case, we also consider the objective of minimizing the maximum difference between the most loaded and least loaded machine. For this case, and the average load is A, if we know that the eventual number of machines is in the range $[\alpha M, M]$, we can show an asymptotic lower bound of $\min\{2/3, 2/(4\alpha+1)\}A$ on the difference and we can obtain an upper bound of $\min\{2/3, 1/(\alpha+1)\}A$.

In Section 3, we consider the general case with arbitrary sized jobs. We give an algorithm which gives a robust ratio of $\frac{5}{3}+\epsilon$, breaking the simple 2 bound that can be obtained by running LPT on M sets and then repeatedly merging the two smallest sets until m sets remain. In our algorithm, we first calculate the optimal

schedule for $m \in \{\frac{M}{2}, \frac{3M}{4}, M\}$ within a factor of $1+\epsilon$ using the algorithm in [11]. We then take one of these schedules and partition the jobs that were scheduled on some machines into a larger number of sets, which we call *bags*. After learning how many machines we have, we place the bags on the machines. This algorithm is more involved than the previous ones, and has several cases, based on the values of opt_M, $\text{opt}_{3M/4}$ and $\text{opt}_{M/2}$ and demonstrates how a more careful investigation of the packing and scheduling steps can lead to improved bounds.

We conclude the introduction with a simple lower bound. Consider the following example. Let the upper bound on the number of machines, M, be 3 and assume we have $n = 6$ identical jobs, each of which has processing time 1. We will prepare 3 bags which will ultimately have to be scheduled on either 1, 2 or 3 machines. Note that the unconstrained optimal makespan for 1, 2 and 3 machines are 6, 3 and 2 respectively. First, consider the packing which gives each bag 2 jobs. Then the makespans are 6, 4, and 2 respectively if there are 1, 2 and 3 machines. Hence this algorithm is $\max\{6/6, 4/3, 2/2\} = 4/3$-robust. Next consider the packing which places $\{1,2,3\}$ jobs in each bag respectively. Then the makespans are 6, 3, and 3 respectively if there are 1, 2 and 3 machines, which makes this algorithm $\max\{6/6, 3/3, 3/2\} = 3/2$-robust. In this example, the former algorithm is better. Moreover, this example demonstrates that $4/3$ is a lower bound on the robustness of any algorithm. Note that there exists a same lower bound for any number of M: Consider an arbitrary M and $n = 2M$ identical jobs with each processing time 1, if we put at least one bag with at least 3 jobs, then the robust ratio is at least $3/2$; otherwise we put 2 jobs in each bag and it provides a robust ratio which is at least $4/3$.

2 Scheduling infinitesimal jobs

Throughout this paper, we use $p(j)$ to denote the processing time of job j. For any set of jobs S, we use $p(S) = \sum_{p_i \in S} p(i)$ to denote the sum of the processing times of jobs in S. We informally say a job set S is *big* if the value of $p(S)$ is large and *small* otherwise.

We now consider the case of infinitesimal jobs. Suppose we are given a job set J with all infinitesimal jobs such that $p(J) = s$, for some $s > 0$. We first pack the jobs to $M \geq 3$ sets and then schedule the bags on m machines, where $m \in [1, M]$ is only known after we pack the jobs. Let $ALG(m, M)$ be the makespan of scheduling the bags on m machines, then our objective is to minimize the robust ratio, $\alpha = \max_m \left\{ \frac{ALG(m,M)}{\text{opt}_m} \right\}$. Recall opt_m is the makespan of a schedule that knew m in advance, and specifically $\text{opt}_m = s/m$ when all jobs are infinitesimal.

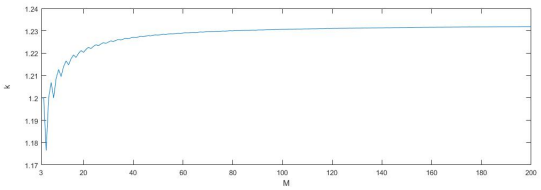

Figure 1: k as a function of M.

The main idea in the packing is to produce a set of bags with a diverse set of sizes. More precisely, we consider the following packing, which we call packing PC. $S_1, S_2, \ldots, S_M$ are the bags: for $i = 1, 2, 3, \ldots, 2\lfloor M/3 \rfloor$, $p(S_i) = \frac{ks}{M - \lceil \frac{i}{2} \rceil} - \frac{ks}{2(M-1)}$; for $j = 2\lfloor M/3 \rfloor + 1, 2\lfloor M/3 \rfloor + 2, \ldots, M$, $p(S_j) = \frac{ks}{M}$, were k is a parameter which only depends on M, chosen to ensure that $\sum_{i=1}^{M} p(S_i) = s$. Specifically, $k = 1/(2 \cdot \sum_{i=1}^{\lfloor M/3 \rfloor} \frac{1}{M-i} - \left\lfloor \frac{M}{3} \right\rfloor \frac{1}{M(M-1)} + 3 \left\{ \frac{M}{3} \right\} \frac{1}{M})$ ≈ 1.233 when M is large (Here we use $\{\}$ to denote the integer remainder). And we will show that for all M, $k \leq 1.2333$. See Figure 1 to see how k changes with M.

Using packing PC to put the jobs into bags, we obtain the following theorem.

THEOREM 2.1. *For $m \in [1, M]$, there exists a schedule with makespan at most $k\text{opt}_m \leq 1.2333\text{opt}_m$ which schedules $\{S_1, S_2, \ldots, S_M\}$ on m machines.*

Proof. We first consider the case when $m \geq M/2$, and we schedule the bags as follows. Let $t = M - m$. For machine $i = 1, 2, \ldots, t$, we schedule bags S_i and S_{2t-i+1} on machine i; for machine $j = t+1, \ldots, m$, we schedule bag S_{j+t} on machine i . The machines with one bag are all within the bound, since for $i \leq 2\left\lfloor \frac{M}{3} \right\rfloor$, $p(S_i) = \frac{ks}{M - \lceil \frac{i}{2} \rceil} - \frac{ks}{2(M-1)} \leq \frac{ks}{M - M/3} - \frac{ks}{2M} = \frac{ks}{M} \leq \frac{ks}{M} \leq \frac{ks}{m} = k\text{opt}_m$. Therefore it remains to bound the load on machines with two bags. We will use $L(i,t)$ to denote the load on the machine i for a particular value of t. We therefore need to prove that $L(i,t) = p(S_i) + p(S_{2t-i+1}) \leq k\text{opt}_m$ for $1 \leq i \leq t$, $t = M - m \leq \left\lfloor \frac{M}{2} \right\rfloor$. Observe that when i is even, $L(i,t) = p(S_i) + p(S_{2t-i+1}) = p(S_{i-1}) + p(S_{2t-i+2}) =$

$L(i-1,t)$, hence we may assume that i is odd. Since $i \le \left\lfloor \frac{M}{2} \right\rfloor \le 2\left\lfloor \frac{M}{3} \right\rfloor$, $p(S_i) = \frac{ks}{M - \frac{i+1}{2}} - \frac{ks}{2(M-1)}$.

We first consider the subcase when $2t - i + 1 \ge 2\left\lfloor \frac{M}{3} \right\rfloor + 1$, then $p(S_{2t-i+1}) = \frac{ks}{M}$. Since i is odd, $i \le 2t - 2\left\lfloor \frac{M}{3} \right\rfloor - 1$. Hence we have

$$\begin{aligned} L(i,t) &= \frac{ks}{M - \frac{i+1}{2}} - \frac{ks}{2(M-1)} + \frac{ks}{M} \\ &\le \frac{ks}{M - t + \lfloor M/3 \rfloor} + \frac{ks(M-2)}{2(M-1)M} \end{aligned} \tag{2.1}$$

Recall that $\mathrm{opt}_m = \frac{ks}{m} = \frac{ks}{M-t}$. Since $t \ge \left\lfloor \frac{M}{3} \right\rfloor + \frac{i}{2}$, $t \ge \left\lfloor \frac{M}{3} \right\rfloor + 1$. Using (2.1), it follows that

$$\begin{aligned} & L(i,t) - \frac{ks}{M-t} \\ &= \frac{ks}{M + \lfloor M/3 \rfloor - t} + \frac{ks(M-2)}{2(M-1)M} - \frac{ks}{M-t} \\ &\le \frac{ks(M-2)}{2(M-1)M} - \frac{ks\lfloor M/3 \rfloor}{(M + \lfloor M/3 \rfloor - t)(M-t)} \\ &\le \frac{ks(M-2)}{2(M-1)M} - \frac{ks\lfloor M/3 \rfloor}{(M-1)(M - \lfloor M/3 \rfloor - 1)} \\ &= ks\frac{M^2 - 3M + 2 - (3M-2)\lfloor M/3 \rfloor}{2M(M-1)(M - \lfloor M/3 \rfloor - 1)} \\ &\le ks\frac{M^2 - 3M + 2 - (3M-2)(M/3 - \frac{2}{3})}{2M(M-1)(M - \lfloor M/3 \rfloor - 1)} \\ &= ks\frac{-M/3 + \frac{2}{3}}{2M(M-1)(M - \lfloor M/3 \rfloor - 1)} \; < \; 0 \; . \end{aligned}$$

That is, $L(i,t) \le \frac{ks}{M-t} = k\mathrm{opt}_m$. Next, we consider the subcase that that $2t - i + 1 \le 2\left\lfloor \frac{M}{3} \right\rfloor$. Now, we have $p(S_{2t-i+1}) = \frac{ks}{M - \frac{2t-i+1}{2}} - \frac{ks}{2(M-1)} = \frac{ks}{M + (i+1)/2 - (t+1)} - \frac{ks}{2(M-1)}$ (Recall that we assume i is odd then $2t - i + 1$ is even). Hence we have

$$\begin{aligned} L(i,t) &= \frac{ks}{M - \frac{i+1}{2}} + \frac{ks}{M + \frac{i+1}{2} - (t+1)} - \frac{ks}{M-1} \\ &= \frac{ks(2M-t-1)}{(M - \frac{i+1}{2})(M + \frac{i+1}{2} - (t+1))} - \frac{ks}{M-1} \\ &= \frac{ks(2M-t-1)}{-\frac{1}{4}(i-t)^2 + \frac{1}{4}(t+1)^2 + M^2 - (t+1)M} \\ &\quad - \frac{ks}{M-1} \\ &\le \frac{ks(2M-t-1)}{-\frac{1}{4}(1-t)^2 + \frac{1}{4}(t+1)^2 + M^2 - (t+1)M} \\ &\quad - \frac{ks}{M-1} \\ &= \frac{ks(2M-t-1)}{(M-1)(M-t)} - \frac{ks}{M-1} = \frac{ks}{M-t} \; . \end{aligned}$$

The inequality holds because $1 \le i \le t$. This proves that for the case $m \ge M/2$, the makespan of our schedule is at most $\frac{ks}{M-t} = k\mathrm{opt}_m$.

Next, we consider the case when $1 \le m < M/2$. Let $x \ge 2$ be the integer such that $\frac{M}{x+1} \le m < \frac{M}{x}$. Let $t = M - mx$. For machine $i = 1, 2, \ldots, t$, we schedule bags S_i, S_{2t-i+1}, $S_{i+2t}, \ldots, S_{i+xt}$ on such machine; for machine $j = t+1, \ldots, m$, we schedule bags $S_{j+xt}, S_{j+xt+(m-t)}$, $S_{j+xt+2(m-t)} \cdots, S_{j+xt+(x-1)(m-t)}$ on such machine. Recall that $\forall i \le M$, $p(S_i) \le \frac{ks}{M}$. Observe that we schedule x bags on the last $m-t$ machines, hence the processing times of jobs on such those machines are at most $\frac{xks}{M} = \frac{ks}{M/x} \le \frac{ks}{m} = k\mathrm{opt}_m$. The processing time of the bags scheduled on machine $i \le t$ is (note that $m = (M-t)/x$):

$$\begin{aligned} p(S_i) + p(S_{2t-i+1}) + \sum_{j=2}^{x} p(S_{i+jt}) &= L(i,t) + \sum_{j=2}^{x} p(S_{i+jt}) \\ &\le \frac{ks}{M-t} + (x-1)\frac{ks}{M} \\ &\le \frac{kxs}{M-t} = k\mathrm{opt}_m \end{aligned}$$

The last is to show a bound of k. Recall that k is chosen to ensure that $\sum_{i=1}^{M} p(S_i) = s$, so we have (here we use $\{\}$ to denote the integer remainder),

$$\begin{aligned} \sum_{i=1}^{M} p(S_i) &= 2 \cdot \sum_{i=1}^{\lfloor M/3 \rfloor} \left(\frac{ks}{M-i} - \frac{ks}{2(M-1)}\right) \\ &\quad + (M - 2\lfloor M/3 \rfloor) \cdot \frac{ks}{M} \\ &= 2 \cdot \sum_{i=1}^{\lfloor M/3 \rfloor} \frac{ks}{M-i} - \lfloor M/3 \rfloor \frac{ks}{M(M-1)} \\ &\quad + 3\{M/3\}\frac{ks}{M} \; . \end{aligned} \tag{2.2}$$

We can solve (2.2) for k and obtain that $k = 1/b(M)$ where $b(M) = 2 \cdot \sum_{i=1}^{\lfloor M/3 \rfloor} \frac{1}{M-i} - \left\lfloor \frac{M}{3} \right\rfloor \frac{1}{M(M-1)} + 3\{\frac{M}{3}\}\frac{1}{M}$. Note that when M is large,

$$b(M) \approx 2 \sum_{i=M-\lfloor M/3 \rfloor}^{M-1} \frac{1}{i} \approx 2(\ln(M-1) - \ln(2M/3)) \approx 2\ln 1.5$$

Hence $k = 1/b(M) \approx 1/(2\ln 1.5) \approx 1.233$ when M is large. We verify by computer that when $M < 10000$. $k \leq 1.2333$. And for $M \geq 10000$,

$$b(M) \geq 2 \cdot \int_0^{\lfloor M/3 \rfloor} \frac{1}{M-i} - \frac{1}{3(M-1)} \geq 2ln\frac{M - \lfloor M/3 \rfloor}{M} - 4 \cdot 10^{-5} \geq 0.81089$$

So $k \leq 1/0.81089 \leq 1.2333$.

We can also show a lower bound. That is, we can show that, no matter how you divide the jobs, you cannot achieve a robust ratio below 1.207, which is close to (but does not exactly match) our upper bound. We state the theorem in terms of a function $Q(M)$ which we define precisely below and which, for large M is 1.207.

THEOREM 2.2. *Let $S_1, S_2, \ldots, S_M$ be M bags of infinitesimal jobs such that $\sum_{i=1}^{M} p(S_i) = s$ for some $s > 0$. Assume there exists a constant k' such for all $m \in [1, M]$, we can schedule $\{S_1, S_2, \ldots, S_M\}$ on m machines with makespan at most $k'opt_m$. Then $k' \geq Q(M)$, where $Q(M) = \max_{t \leq \frac{M}{2}, t \in N} \frac{1}{\frac{t}{M-t} + \frac{M-2t}{M}} \approx 1.207$.*

Proof. We may assume that $p(S_1) \leq p(S_2) \leq \cdots \leq p(S_M)$. We first prove the following statement.

(2.3)
For $t \leq M/2$, there exists a schedule of $\{S_1, S_2, \ldots, S_M\}$ on $m = M - t$ machines with minimum makespan such that $S_1, S_2, \ldots, S_{2t}$ are on the first t machines.

Let $\{T_1, T_2, \ldots, T_m\}$ be a schedule of $\{S_1, S_2, \ldots, S_M\}$ on m machines with minimum makespan, where T_i is a set of bags scheduled on machine i. We may assume that every T_i contains at least one bag. Since we schedule M bags on $M - t$ machines, there are at least $M - 2t$ machines contain exactly one bag. We rename the machines such that $T_{t+1}, T_{t+2}, \ldots, T_{t+(M-2t)} = T_m$ contain only one bag. Suppose that at least one of $S_1, S_2, \ldots, S_{2t}$ is not scheduled on the first t machines, that is, there exists $i \leq 2t$, $j \geq t+1$ such that $T_j = \{S_i\}$. Then since $|\bigcup_{i=1}^{t} T_i| = M - |\bigcup_{i=t+1}^{m} T_i| = 2t$, there exists $i' \geq 2t+1$, $j' \leq t$ such that $S_{i'} \in T_{j'}$. Define $T'_j = \{S_{i'}\}$, $T'_{j'} = T_{j'} \setminus \{S_{i'}\} \cup \{S_i\}$, and $T'_\ell = T_\ell$ for $\ell \neq j, j'$. Since $p(S_{i'}) \geq p(S_i)$, $p(T'_{j'}) \leq p(T_{j'})$. Note also that $p(T'_j) \leq \max_{i=1}^{M}\{p(S_i)\} \leq \max_{i=1}^{m}\{p(T_i)\}$. Hence $\max_{i=1}^{m}\{p(T'_i)\} \leq \max_{i=1}^{m}\{p(T_i)\}$. This implies that $\{T'_1, \ldots, T'_m\}$ is also a schedule with minimum makespan. Note that the first t machine of schedule $\{T'_1, \ldots, T'_m\}$ contains more bags from $\{S_1, S_2, \ldots, S_{2t}\}$ than the first t machine of schedule $\{T_1, \ldots, T_m\}$. By repeating the above process, we can get a schedule with minimum makespan such that $S_1, S_2, \ldots, S_{2t}$ are all scheduled on the first t machines. This completes the proof of (1).

By choosing $m = M$, we know that $p(S_i) \leq k'\text{opt}_M = \frac{k's}{M}$ for any $1 \leq i \leq M$. For $t \leq M/2$, consider the schedule with minimum makespan on $m = M - t$ machines such that $S_1, S_2, \ldots, S_{2t}$ are scheduled on the first t machines. It follows that $\sum_{i=1}^{2t} p(S_i) \leq t \cdot k'\text{opt}_m = \frac{tk's}{M-t}$

Hence for any $t \leq M/2$, we have

$$s = \sum_{i=1}^{M} p(S_i) = \sum_{i=1}^{2t} p(S_i) + \sum_{i=2t+1}^{M} p(S_i) \leq \frac{tk's}{M-t} + (M-2t) \cdot \frac{k's}{M}$$

It follow that $k' \geq Q(M) = \max_{t \leq \frac{M}{2}, t \in N} \frac{1}{\frac{t}{M-t} + \frac{M-2t}{M}}$. Let $L(t) = \frac{t}{M-t} + \frac{M-2t}{M}$. Then by taking the derivative, it is not hard to obtain that $\min_{0 \leq t \leq \frac{M}{2}} L(t) = L((1-\frac{\sqrt{2}}{2})M) = 2(\sqrt{2}-1)$. And $Q(M) \approx \frac{1}{\min_{0 \leq t \leq \frac{M}{2}} L(t)} = \frac{\sqrt{2}+1}{2} \approx 1.207$.

Figure 2 shows how $Q(M)$ changes with M.

2.1 Minimizing the Maximum Difference Another objective we consider is to minimize the difference between the load of the most loaded and least loaded machines. This objective is particularly relevant to settings in which we want to achieve *fairness*. First of all, minimizing the maximum difference is a well-studied scheduling objective in situations where fairness is important. Second, it models a type of fair allocation

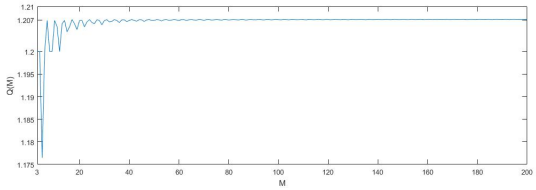

Figure 2: $Q(M)$ as a function of M

problem. Suppose that we want to split some goods among m people, where we don't know what m is in advance. In order to simplify the process, we can first divide the goods into M groups or bags, and then have the restriction that each person must take a subset of the bags.

For this problem, we consider a case where after packing the jobs into M bags, we are given $m \in [\alpha M, M]$ machines on which to schedule. Our bounds (see below) will depend on α, with, not surprisingly, better bounds for larger α. Since we assume every job is infinitesimal, the minimum difference between the load of the most loaded and least loaded machines is always 0 if we know the number of machines in advance. So we can not use the α-robust settings to evaluate our algorithms in this section and we introduce some new definitions here. Let s denote the sum of processing time of all jobs and let $A = s/M$. For a set of bags $S = \{S_1, S_2, \ldots, S_M\}$, we use $D_m(S)$ to denote the minimum difference between the load of the most loaded and least loaded machines when we schedule S on m machines. Our goal is to choose S so as to minimize $D(M, \alpha, S, A) = \max_{m\in[\alpha M, M]}\{D_m(S)\}$.

Assume that αM is an integer with value at most $M-1$ and $M \geq 3$. Let $D^*(M, \alpha, A) = \min_S\{D(M, \alpha, S, A)\}$. We now give a lower bound of $D^*(M, \alpha, A)$, and the proof is included in the appendix.

THEOREM 2.3. *If $\alpha M > \frac{M}{2}$, then*

$$D^*(M, \alpha, A) \geq \frac{2(1-\alpha)M}{1+(4\alpha+1)(1-\alpha)M} \cdot A \ .$$

If $\alpha M \leq \frac{M}{2}$, then

$$D^*(M, \alpha, A) \geq \frac{2M^2-2M}{3M^2+M-2} \cdot A, \textit{ if } M \textit{ is odd;}$$

$$D^*(M, \alpha, A) \geq \frac{2M^2-4M}{3M^2-8} \cdot A, \textit{ if } M \textit{ is even.}$$

Proof. We may assume that $p(S_1) \leq p(S_2) \leq \cdots \leq p(S_M)$. For $i = 1, 2, \ldots, M$, let $A_i = \sum_{j=1}^{i} p(S_j)/i$. We first prove the following statement.

For $m > M/2$,

$$D_m(M, \alpha, S, A) \geq 2A_{2M-2m} - p(S_{2M-2m+1}). \tag{2.4}$$

It is sufficient to proof the following: there exists a scheduling to minimize the differences by schedule $S_1, S_2, \ldots, S_{2M-2m}$ on the first $M-m$ machines and schedule $S_{2M-2m+1}, \ldots, S_M$ to the rest of the $2m-M$ machines. Since then the maximum load on a machine is at least $\sum_{i=1}^{2M-2m} S_i/(M-m) = 2A_{2M-2m}$ and the minimum load on a machine is at most $S_{2M-2m+1}$. Suppose not. Let the optimal way to schedule $\{S_i\}$ on m machines is by scheduling a set of bags, Y_i, on the i-th machine. We may assume that every Y_i contains at least one bag. Note that are at least $2m-M$ machine receive exactly one bag. We rename the machines so that $Y_{M-m+1}, Y_{M-m+2}, \ldots, Y_{M-m+(2m-M)} = Y_m$ contain only one bag. Suppose there exists $i \leq 2M-2m$, $j \geq M-m+1$ such that $Y_j = \{S_i\}$. Then there exists $i' \geq 2M-2m+1$, $j' \leq M-m$ such that $S_{i'} \in Y_{j'}$. Define $Y'_j = \{S_{i'}\}$, $Y'_{j'} = Y_{j'} \setminus S_{i'} \cup \{S_i\}$, and $Y'_\ell = Y_\ell$ for $\ell \neq j, j'$. Since $p(S_{i'}) \geq p(S_i)$, $p(Y'_{j'}) \geq p(Y_{j'})$. Note also that $p(Y'_j) \leq \max_{i=1}^{M}\{p(S_i)\} \leq \max_{i=1}^{M}\{p(Y_i)\}$. Hence $\max_{i=1}^{m}\{p(Y'_i)\} \leq \max_{i=1}^{m}\{p(Y_i)\}$. Since $p(Y'_{j'}) \geq p(S_i) = p(Y_j)$ and $p(Y'_j) = p(S_{i'}) \geq p(Y_j)$, $\min_{i=1}^{m}\{p(Y'_i)\} \geq \min_{i=1}^{m}\{p(Y_i)\}$. This implies that $\{Y'_i\}$ is also an optimal way of scheduling the bags. So we may assume that $\{Y_i\}$ satisfies that $\bigcup_{i=M-m+1}^{m} Y_i = \{S_{2M-2m+1}, S_{2M-2m+2} \ldots S_{2M-2m+2m-M} = S_M\}$. This completes the proof of (4).

Let $D = D^*(M, \alpha, A)$ and let $S = \{S_i\}$ be the set of bags that reaches the optima. First we assume that $\alpha M > \frac{M}{2}$. Let $A' = A_{2M-2\alpha M}$. By (2.4), $p(S_{(2-2\alpha)M+1}) \geq 2A' - D_{\alpha M}(S) \geq 2A' - D$. Since $D \geq D_M(S) = p(S_M) - p(S_1) \geq p(S_{(2-2\alpha)M+1}) - p(S_1)$, $p(S_1) \geq 2A' - 2D$ and $p(S_M) \geq p(S_{(2-2\alpha)M+1}) \geq 2A' - D$.

We know that $A_2 \geq p(S_1) \geq 2A' - 2D$. For $1 \leq k < (1-\alpha)M$, assume we know that $A_{2k} \geq 2A' - 2D + \frac{k-1}{2}(2A' - 3D)$. Then by (2.4), we have $p(S_{2k+1}) \geq 2A_{2k} - D_{M-k}(S) \geq 2(2A' - 2D + \frac{k-1}{2}(2A' - 3D)) - D = 2A' - 2D + k(2A' - 3D)$. Also $p(S_{2k+2}) \geq p(S_{2k+1}) \geq 2A' - 2D + k(2A' - 3D)$. Therefore $A_{2k+2} \geq A_{2k}\frac{2k}{2k+2} + (2A' - 2D + k(2A' - 3D))\frac{2}{2k+2} \geq 2A' - 2D + \frac{k}{2}(2A' - 3D)$. Inductively, this implies that $A' = A_{2M-2\alpha M} \geq 2A' - 2D + \frac{(1-\alpha)M-1}{2}(2A' - 3D)$, which is equivalent to

$$D \geq \frac{2(1-\alpha)M}{1+3(1-\alpha)M} A'$$

Recall that $p(S_i) \leq p(S_1) + D \leq A' + D$ for any i, we have

$$\begin{aligned} MA = s &\leq 2(1-\alpha)MA' + (2\alpha-1)M(A'+D) \\ &= MA' + (2\alpha-1)MD \\ &\leq \frac{1+3(1-\alpha)M}{2(1-\alpha)}D + (2\alpha-1)MD \end{aligned}$$

This is equivalent to

$$D \geq \frac{2(1-\alpha)M}{1+(4\alpha+1)(1-\alpha)M} \cdot A \ .$$

For the case of $\alpha M \leq \frac{M}{2}$, if M is odd, then

$$D^*(M,\alpha,A) \geq D^*(M, \frac{M+1}{2M}, A) \geq \frac{2M^2-2M}{3M^2+M-2} \cdot A \ .$$

If M is even, then

$$D^*(M,\alpha,A) \geq D^*(M, \frac{M+2}{2M}, A) \geq \frac{2M^2-4M}{3M^2-8} \cdot A \ .$$

These bounds can most simply be parsed by saying that, if M is large then the lower bound is given by $\min\{2/3, 2/(4\alpha+1)\}A$. Note that when $\alpha \leq \frac{1}{2}$, the lower bound goes to $\frac{2}{3}A$ as M goes large. If M is even, then the bound of $\frac{2}{3}A$ can be reached by setting the bags as follows. Choose $S = \{S_1, S_2, \ldots, S_M\}$ such that $p(S_1) = p(S_2) = \cdots = p(S_{M/2}) = \frac{2}{3}A$ and $p(S_{M/2+1}) = p(S_{M/2+2}) = \cdots = p(S_M) = \frac{4}{3}A$, then it is easy to verify that $D_m(S) \leq \frac{2}{3}A$ for any $m \in [1, M]$.

For $\alpha > \frac{1}{2}$, we can set $D = \dfrac{(1-\alpha)M}{\alpha+(1-\alpha)(\alpha+1)M} \cdot A$ and $A' = \dfrac{1+3(1-\alpha)M}{2(1-\alpha)M} \cdot D$. Now we can choose the bags, $T = \{T_1, T_2, \ldots, T_M\}$, as follows. For $k \leq (1-\alpha)M$, set $p(T_{2k+1}) = p(T_{2k+2}) = 2A' - 2D + k(2A' - 3D)$; for $i \geq 2(1-\alpha)M + 1$, set $p(T_i) = 2A' - D$. The idea of this construction follows from the proof of Theorem 2.3 in the appendix, and by following this proof, it is not hard to verify that $D_m(S) \leq D$ for any $m \in [\alpha M, M]$. Note that $D' \approx \frac{1}{\alpha+1} \cdot A$ if M is large.

3 Scheduling discrete jobs

In this section we will provide an algorithm which, for any set of jobs, gives a $\frac{5}{3}+\epsilon$ robust ratio for $m \in [1, M]$. For ease of presentation, in this section we assume M is divisible by 4. In Appendix B, we sketch the changes that are needed when M is not divisible by 4.

Our algorithm will start by computing minimum makespan schedules for M, $3M/4$ and $M/2$ machines. We can use a PTAS for minimizing makespan on parallel machines to compute a $(1+\epsilon)$-approximate schedule.

Partition $I(S)$

Input: A set of jobs $S = \{j_1, j_2, .., j_K\}$ with $p(j_1) \geq p(j_2) \geq .. \geq p(j_K)$ and $p(S) \leq b$.
Main Process: Run LPT on 2 machines, A and B, breaking ties in favor of A. If $p(A) \leq p(B)$, swap the names of A and B.
Output: (A, B).

Figure 3: Partition I

We use opt'_i to denote the value of the makespan obtained by running the PTAS on a scheduling instance with i machines. We will especially use $\text{opt}'_{M/2}$ and will denote this value by b. We use $PTAS(i) = \{S_1, S_2, ..S_i\}$ to denote the result of the PTAS on a scheduling instance with i machines, where S_j is the set of jobs assigned to machine j.

Based on the values opt'_M, $\text{opt}'_{3M/4}$ and $\text{opt}'_{M/2}$, we will consider several cases. Each case will have the same structure. First, we will compute sets of jobs S_i from a PTAS. Second we will use *partitioning* routines to split the job set into M bags. We then learn the number of machines and need to schedule the bags on machines. In different cases, we will use different combinations of partitioning and scheduling routines. We then show that, for each case, the resulting schedule is $\frac{5}{3}(1+\epsilon)$ robust.

In Section 3.1, we will give our partitioning routines and prove some properties about each one. In Section 3.2, we will describe how to assemble the jobs into bags and then schedule them on machines.

3.1 Partitioning In this subsection we will describe three useful partitioning algorithms, which will be used as subroutines in the main algorithm. Each one will take one or four sets of jobs and partition them into multiple bags with special properties, achieving some balance between the sizes of the bags and controlling the placement of large jobs. In all routines we will assume that we process the jobs in sorted size order, largest-to-smallest.

We begin with the first partitioning algorithm, Partition I, which partitions a job set into two bags such that neither of them is too big. This partitioning algorithm will be useful when we want to partition a job set "evenly". It implements the LPT algorithm, sorting the jobs in non-increasing order and then repeatedly placing the next job in the least loaded bag. It appears in Figure 3.

We now bound the sizes of the bags. Recall that the

standard analysis of LPT shows that it is a $4/3-1/3m = 7/6$-approximation algorithm on 2 machines [8]. The bounds that we give are tight and are not implied by the standard analysis of LPT.

LEMMA 3.1. *Let (A, B) be the output of Partition I, then $p(A) = \max\{p(A), p(B)\} \leq \max\{p(j_1), 2b/3\}$. If the maximum is achieved by $p(j_1)$ then $A = \{j_1\}$.*

Proof. If j_1 has larger processing time than all the other jobs combined, i.e. $p(j_1) \geq \sum_{i=2}^{K} p(j_i)$, then for $i = 2, .., K$, LPT will put j_i in B and $\max\{p(A), p(B)\} = p(A) = p(j_1)$.

Next consider the case when $p(j_1) < \sum_{i=2}^{K} p(j_i)$. We use d_t to denote the difference in the loads of A and B after adding job j_t (Note that $d_K = p(A) - p(B)$) Pick k as the smallest integer such that $p(j_1) < \sum_{i=2}^{k} p(j_i)$, clearly $k \geq 3$. Then $p(j_k) \geq d_k = \sum_{i=2}^{k} p(j_i) - p(j_1)$ since $\sum_{i=2}^{k-1} p(j_i) \leq p(j_1)$. When we decide where to put j_{i+1}, the difference between the loads of A and B is d_i, and we put j_{i+1} in the bag with less load. Hence we must have $d_{i+1} \leq \max\{p(j_{i+1}), d_i\}$. Then inductively $d_K \leq \max\{p(j_K), p(j_{K-1}), .., p(j_{k+1}), p(j_k), d_k\}$. Because the jobs are indexed largest to smallest, and because $k \geq 3$, we can simplify the previous inequality to $d_K \leq p(j_3) \leq b/3$ (recall that $d_k \leq j_k$). Since $p(A) + p(B) = b$, it immediately follows that $\max\{p(A), p(B)\} \leq 2b/3$.

Since we almost always need to put multiple bags on one machine, we want to create enough small bags to control the makespan. This demand leads to the next two partitioning algorithms, in which we partition job sets to bags such that half of them are small enough and the others are not too big. The first one, Partition II, works for the case when we have a job set with at most one large job. We will place the job with the largest processing time in set A and then fill set B greedily up to $b/3$ and then place the remaining jobs in A. The details appear in Figure 4.

LEMMA 3.2. *Let (A, B) be the output of Partition II, then $p(A) \leq 5b/6$ and $p(B) \leq b/3$.*

Proof. By line 2 clearly we have $p(B) \leq b/3$. Suppose, for a contradiction, that $p(A) > 5b/6$. Then $t \neq K$ and it follows that $p(j_1) + \sum_{i=t+1}^{K} p(j_i) > 5b/6$. Since $\sum_{i=1}^{K} p(j_i) \leq b$, we must have $\sum_{i=2}^{t} p(j_i) = \sum_{i=1}^{K} p(j_i) - (p(j_1) + \sum_{i=t+1}^{K} p(j_i)) < b/6$.

Partition II(S)

Input: A set of jobs $S = \{j_1, j_2, \ldots, j_K\}$ with $5b/6 \geq p(j_1) \geq p(j_2) \geq \cdots \geq p(j_K)$, $p(j_2) \leq b/3$ and $p(S) \leq b$.

Main Process:

1 $B = \emptyset$, $A = \{j_1\}$.

2 Let t be the largest integer such that $p(j_2) + p(j_3) + \cdots + p(j_t) \leq \frac{b}{3}$; put $j_2, \ldots, j_t$ in B.

3 Add $j_{t+1}, \ldots, j_K$ into A if $t \neq K$.

Output: (A, B).

Figure 4: Partition II

But since $\sum_{i=2}^{t+1} p(j_i) > b/3$, we also have $p(j_{t+1}) = \sum_{i=2}^{t+1} p(j_i) - \sum_{i=2}^{t} p(j_i) > b/3 - b/6 = b/6$. Note that $t \geq 2$, hence $p(j_2) \geq p(j_{t+1}) > b/6 > \sum_{i=2}^{t} p(j_i)$, a contradiction.

The last partitioning algorithm, Partition III, will handle the case when we have at least two large jobs and Partition II is not working. Specifically, we will take four job sets as input; three of them, S_1, S_3, S_4, have two large ($\geq b/3$) jobs and the other, S_2, has all small ($< b/3$) jobs. We partition them into eight bags such that we have four small bags and four bags that are not too big. The algorithm first puts the second biggest job from S_1 into A_1 and the remaining jobs from S_1 into A_2. It then greedily puts jobs from S_2 into A_1 and A_2 as long as they don't cause the load to go over $5b/6$. By greedily, here, we mean that it goes through the jobs in non-decreasing order of processing time, and if the job can fit on A_1 or A_2, we place it on one which it fits. We then use Partition I on the remaining jobs of S_2, running LPT to place jobs on B_1 and B_2. We also use Partition I to partition S_3 to (A_3, B_3), and S_4 to (A_4, B_4). We will show that either $p(B_1) + p(B_3)$ and $p(B_2) + p(B_4)$ are both not too big or we can switch jobs to ensure that neither is. The details of the algorithm appear in Figure 5.

LEMMA 3.3. *Let $(A_1, A_2, A_3, A_4, B_1, B_2, B_3, B_4)$ be the output of Partition III. These bags satisfy:*

1. *$p(A_i) \leq 5b/6$, for $i = 1, 2, 3, 4$;*
2. *$p(B_i) \leq b/2$, for $i = 1, 2, 3, 4$;*
3. *$p(B_1) + p(B_3) \leq 5b/6$, $p(B_2) + p(B_4) \leq 5b/6$.*

Partition III(S_1, S_2, S_3, S_4)

Input: Four sets of jobs: $S_1 = \{x_1, \ldots, x_p\}$ such that $p(x_1) \geq p(x_2) \geq b/3$, $p(S_1) \leq b$, $S_2 = \{y_1, \ldots, y_q\}$ such that $b/3 \geq p(y_1) \geq .. \geq p(y_q)$, $p(S_2) \leq b$, S_3 with $p(S_3) \leq b$ and S_4 with $p(S_4) \leq b$. Moreover for $i = 1, 2, 3, 4$, $p(j) \leq 2b/3$ for every $j \in S_i$.

Main Process:

1 $A_1 = \{x_2\}$. $A_2 = S_1 - \{x_2\}$, $i = 1$.
2 **while** ($(p(A_1) + p(y_i) \leq 5b/6)$ or $(p(A_2) + p(y_i) \leq 5b/6)$) and $(i \geq 1)$
3 if $p(A_1) + p(y_i) \leq 5b/6$, add y_i to A_1 else add y_i to A_2,
4 $i++$
5 Let S' be the jobs remaining in S_2.
6 Set (B_1, B_2) = Partition I(S'), (A_3, B_3) = Partition I(S_3), (A_4, B_4) = Partition I(S_4).
7 **if** $B_1 \cup B_2$ contains 3 jobs and B_2 contains only one job, let j_3 be the job with the least processing time in B_1.
if $p(j_3) + p(A_3) \leq 5b/6$, move j_3 from B_1 to A_3.
else if $p(j_3) + p(A_4) \leq 5b/6$, move j_3 from B_1 to A_4;
else Set $temp = B_1$, $B_1 = B_4$, $B_4 = temp$.
8 **else if** $B_1 \cup B_2$ contains more than 3 jobs and B_2 contains one job.
Let j' denotes the job in B_2 with the least processing time. Move j' from B_2 to A_3 if $p(j') + p(A_3) \leq 5b/6$.

Output: $(A_1, A_2, A_3, A_4, B_1, B_2, B_3, B_4)$.

Figure 5: Partition III

Proof. Let k_{ij} denote the load of jobs from S_i that are placed in bag A_j. Note that the jobs in bags B_1 and B_2 all come from S_2. Thus $k_{11} + k_{12} \leq b$ and $k_{21} + k_{22} + p(B_1) + p(B_2) \leq b$. For $i = 1, 2$, define the non-negative slack in each A_i as $\delta_i = 5b/6 - k_{1i} - k_{2i}$ and $\delta = \max\{\delta_1, \delta_2\}$.

Note that at line 1, $p(A_1) = p(x_2) \leq b/2$ and $p(A_2) = p(S_1) - p(x_2) \leq b - b/3 \leq 2b/3$. Since we call Partition I to separate S_3 and S_4, by Lemma 3.1, $p(A_3) \leq 2b/3$, $p(A_4) \leq 2b/3$ at line 6. Since for $i = 1, 2, 3, 4$, we will only add job to A_i as long as the load does not exceed $5b/6$, condition 1 holds.

We now consider conditions 2 and 3. By line 2, any job in $B_1 \cup B_2$ has load greater than δ. First consider the case when $B_1 \cup B_2$ contains at most two job. Then $p(B_1) \leq b/3$ and $p(B_2) \leq b/3$. Note that also $p(B_3) \leq p(S_3)/2 \leq b/2$ and similarly $p(B_4) \leq b/2$, hence the claim follows.

Next we consider the case when $B_1 \cup B_2$ contains $l \geq 3$ job before line 7. Note that,

$$\begin{aligned} p(B_1) + p(B_2) &\leq b - k_{21} - k_{22} \\ &= b - \left(\frac{5b}{6} - \delta_1 - k_{11}\right) - \left(\frac{5b}{6} - \delta_2 - k_{12}\right) \\ &\leq \frac{b}{3} + 2\delta \ . \end{aligned}$$

If both B_1 and B_2 contain at least 2 jobs, then $p(B_1) \geq 2\delta$ and $p(B_2) \leq b/3 + 2\delta - p(B_1) \leq b/3$. Similarly $p(B_2) \leq b/3$, hence the claim follows.

Next we consider the case when $l = 3$. If B_1 contains only one job, then $b/3 \geq p(B_1) \geq p(B_2)$ and the claim follows. We may assume $B_1 = \{j_2, j_3\}$, $B_2 = \{j_1\}$ and $p(j_1) \geq p(j_2) \geq p(j_3)$. Since $k_{11} = p(x_2) \leq b/2$ and $5b/6 - k_{11} \geq b/3 \geq p(y_1)$, we will always put y_1 in A_1. It implies that S_2 contains at least 4 jobs and $j_3 \leq p(S_2)/4 \leq b/4$. If in line 7 we move j_3 to either A_3 or A_4, then clearly claim follows since after line 7, both B_1 and B_2 contains one job and thus have load at most $b/3$. So we may assume that before line 7, $p(A_3) > 5b/6 - p(j_3) \geq 5b/6 - b/4 = 7b/12$, and then $p(B_3) \leq b - p(A_3) < 5b/12$. Similarly, we have $p(B_4) < 5b/12$. Hence $p(B_3) + p(B_4) < 5b/6$ before line 7. Note also that $p(j_2) + p(j_3) \leq p(j_1) + p(y_1)$, hence $p(j_2) + p(j_3) \leq p(S_2)/2 \leq b/2$. So before line 7, $p(B_1) \leq b/2$ and $p(B_2) \leq b/3$. Recall that we will swap the name of B_1 and B_4, hence the claim follows.

The last case is when $l \geq 4$ and one of B_1 and B_2 contains only one job. If B_1 contains only one job, then $b/3 \geq p(B_1) \geq p(B_2)$ and the claim follows. We may assume that $B_2 = \{j_1\}$ and $B_1 = \{j_2, j_3, \ldots, j_l\}$ with $p(j_1) \geq p(j_2) \geq \cdots \geq p(j_l)$. Then by Partition I, $p(j_2) + \cdots + p(j_{l-1}) \leq p(B_1) \leq b/3$. Recall that $l \geq 4$, hence $p(j_l) \leq (p(j_2) + \cdots + p(j_{l-1}))/(l-2) \leq b/6$. By Lemma 3.1, $p(A_3) \leq 2b/3$. Thus $p(A_3) + p(j_k) \leq 5b/6$. So in line 8 we will always move j_k to A_3. After line 8, $p(B_2) = p(j_2) + \cdots + p(j_{l-1}) \leq b/3$. Hence the claim also follows.

3.2 Packing and Scheduling In the previous subsection, we gave different algorithms to partition jobs into bags. We will now show how to use these algorithms in conjunction with additional algorithms to schedule the bags (and thus jobs) onto machines. Recall that $\text{opt}'_{M/2}$ is the value of the makespan obtained by running the PTAS on a scheduling instance with $M/2$ machines and $b = \text{opt}'_{M/2}$. Two simple facts we will use throughout this section are that opt_m is non-increasing with m, the number of machine, and that $2\text{opt}_M \geq \text{opt}_{M/2}$, which implies that $\text{opt}_M \geq \frac{\text{opt}'_{M/2}}{2(1+\epsilon)} = \frac{b}{2(1+\epsilon)}$.

We now prove our bound of $\frac{5}{3} + \epsilon$ by considering

three different cases. The first case is when $opt'_M \geq 3b/5$, which implies that the PTAS of the jobs on $M/2$ is good enough for M machines. We remark that if this case does not hold then there is no job with processing time greater than $3b/5$. The next case is when $opt'_{3M/4} \leq 4b/5$, which implies that we can start with $opt'_{3M/4}$ and do not need to split all job sets. The last and the hardest case is when $opt'_{3M/4} > 4b/5$ and $opt'_M < 3b/5$. In this case we will start with the PTAS on $M/2$ and split each job sets according to its structure. Before going to the detail of those three cases, we start with a lemma which gives a sufficient condition for bags to give a $\frac{5}{3}(1+\epsilon)$ robust ratio when the number of machine is less than $M/2$. It will be used in both the second and the last case.

LEMMA 3.4. *Let $\{S_1, \ldots, S_M\}$ denotes M bags such that for $i = 1, 2, \ldots, M/2$, $p(S_i) \leq 5b/6$, and for $i = M/2+1, M/2+2, \ldots, M$, $p(S_i) \leq 2b/3$. Then the following algorithm yields a schedule for the bags with a makespan of at most $5(1+\epsilon)\text{opt}_m/3$ for any $m \in [1, M/2)$.*

1 **if** $M/4 \leq m < M/2$. For $i = 1, 2, \ldots, m$, put S_i on machine i; for $i = m+1, m+2, \ldots, M/2$, put S_i on machine $i-m$; for $i = M/2+1, \ldots, M$, put S_i on the machine with the least load.
2 **if** $\frac{M}{2(k+1)} \leq m < \frac{M}{2k}$ for some integer $k \geq 2$, sort $\{S_1, S_2, \ldots, S_M\}$ in decreasing order according to their processing times. For $i = 1, \ldots, m$, put S_i on machine i; for $i = m+1, \ldots, 2m$, put S_i on machine $i-m$; for $i = 2m+1, \ldots, M$, put S_i on the machine with the least load.

Proof. First note that since $m < M/2$, $\text{opt}_m \geq \text{opt}_{M/2} \geq b/(1+\epsilon)$. Let machine r be the machine with the largest load and among the bags scheduled on machine r, S_t is the one with the largest index. For the case $M/4 \leq m < M/2$, if $t > M/2$, then we know $p(S_t) \leq 2b/3 \leq 2(1+\epsilon)\text{opt}_m/3$ and the makespan of our algorithm is at most $p(S_t) + \text{opt}_m \leq 5(1+\epsilon)\text{opt}_m/3$. Else $t \leq M/2$, then we know that we only put two bags on machine r, and therefore the makespan is at most $2 \cdot 5b/6 = 5b/3 \leq 5(1+\epsilon)\text{opt}_m/3$.

For the case $\frac{M}{2(k+1)} \leq m < \frac{M}{2k}$ with $k \geq 2$, if $t \leq 2m$, then similarly we know that we only put two bags on machine r, and therefore the makespan is at most $2 \cdot 5b/6 = 5b/3 \leq 5(1+\epsilon)\text{opt}_m/3$. So we may assume that $t \geq 2m+1$. Since $\sum_{i=1}^{2m} p(S_i) \geq \sum_{i=1}^{2m} p(S_t) = 2mp(S_t)$, $\text{opt}_m \geq \sum_{i=1}^{2m} p(S_i)/m \geq 2p(S_t)$. This implies that $p(S_t) \leq \text{opt}_m/2$ and the makespan of our algorithm is at most $p(S_t) + \text{opt}_m \leq 3\text{opt}_m/2$.

Algorithm 1

Packing:
1 Let $S = PTAS(M/2) = \{S_1, \ldots, S_{M/2}\}$.
2 **for** $k = 1 \ldots, M/2$, let $(A_k, B_k) = \text{Partition I}(S_k)$.
3 Return the M bags $\{A_1, \ldots, A_{M/2}, B_1, \ldots, B_{M/2}\}$.

Scheduling:
1 **if** $m \geq M/2$, schedule A_i and B_i on machine i for $i \leq M/2$. Leave the remaining machines empty.
2 **if** $m < M/2$, run LPT to schedule the bags on m machines.

Figure 6: Algorithm 1

3.2.1 Case I: $\text{opt}'_M \geq 3b/5$. The first case is quite simple. We start with $PTAS(M/2)$, and partition each job set into two bags using Partition I. Those are our bags. For scheduling, if $m \geq M/2$, we just revert to the $M/2$ machine schedule, leaving the remaining machines empty. If $m < M/2$, we can run LPT to the bags on the machines. The details appear in Figure 6.

LEMMA 3.5. *If $\text{opt}'_M \geq 3b/5$, Algorithm 1 is $\frac{5}{3}(1+\epsilon)$-robust.*

Proof. For $m \geq M/2$,

$$\frac{ALG(m,M)}{\text{opt}_m} \leq \frac{\text{opt}'_{M/2}}{\text{opt}_M} \leq \frac{b}{\frac{\text{opt}'_M}{1+\epsilon}} \leq \frac{b}{\frac{3b/5}{1+\epsilon}} = (1+\epsilon)\frac{5}{3}.$$

For $m < M/2$, $\text{opt}_m \geq \text{opt}_{M/2} \geq b/(1+\epsilon)$. We may assume the bags are $S'_1, \ldots, S'_M$ such that $p(S'_1) \geq p(S'_2) \cdots \geq p(S'_M)$. Let k be the number such that for $i \leq k$, $p(S'_i) > 2b/3$ and for $i > k$, $p(S'_i) \leq 2b/3$. Recall that each bag S'_i comes from running Partition I on some job set S_j with $p(S_j) \leq b$. Hence by Lemma 3.1, either $p(S'_i) \leq 2b/3$ or it is a bag with a single job. Specifically, $S'_1, \ldots, S'_k$ all contain only one job. Let the jobs be $j_1, j_2, \ldots, j_k$ respectively. Let machine r be the machine with the largest load and among the bags scheduled on machine r, S_t is the one with the largest index. If $t > k$, then we know $p(S_t) \leq 2b/3 \leq 2(1+\epsilon)\text{opt}_m/3$. By the property of LPT, the makespan of our algorithm is at most $p(S_t) + \text{opt}_m \leq 5(1+\epsilon)\text{opt}_m/3$. If $t \leq k$, then the makespan of our algorithm is equal to the makespan of running LPT on jobs $\{j_1, j_2, \ldots, j_k\}$, which is at most 4/3 times the minimum makespan of scheduling $\{j_1, j_2, \ldots, j_k\}$ on m machines [8]. Since the minimum makespan of scheduling $\{j_1, j_2, \ldots, j_k\}$ on m machines is at most opt_m, our makespan in this case is at most $4\text{opt}_m/3$.

Algorithm 2

Packing:

1 Let $S = PTAS(3M/4) = \{S_1, .., S_{3M/4}\}$. Let v be the number of bad sets in S. Rename the sets so that $S_1, \ldots, S_v$ are bad sets.

2 For $\frac{M}{2} < k \leq \frac{3M}{4}$, let $(A_k, B_k) =$ Partition I(S_k).

3 Return the M bags $\{S_1, S_2, .., S_{M/2}, A_{M/2+1}, \ldots, A_{3M/4}, B_{M/2+1}, .., B_{3M/4}\}$.

Scheduling:

1 **if** $m \geq \frac{3M}{4}$, schedule S_i on machine i for $i \leq \frac{M}{2}$, and schedule A_j, B_j on machine j for $j > \frac{M}{2}$. Leave the remaining machines empty.

2 **if** $v > \frac{M}{2}$ and $v \leq m < \frac{3M}{4}$, first put $S_1, S_2, \ldots, S_{M/2}, A_{M/2+1} \cup B_{M/2+1}, \ldots, A_v \cup B_v$ on v machines respectively, then schedule the rest of the bags with at most one in each machine.

3 **if** $v > \frac{M}{2}$ and $\frac{M}{2} \leq m < v - 1$, or $v \leq \frac{M}{2}$ and $\frac{M}{2} \leq m < \frac{3M}{4}$, schedule the bags arbitrarily such that each machine contains at most two bags and at most one bag comes from $\{S_1, \ldots, S_{M/2}\}$.

4 **if** $m < \frac{M}{2}$, we follow the strategy in Lemma 3.4.

Figure 7: Algorithm 2

3.2.2 Case II: $\text{opt}'_M < \frac{3b}{5}$ and $\text{opt}'_{3M/4} \leq 4b/5$. In this case, we will use the PTAS schedule for $3M/4$ machines as input to Partition I. For $S_i \in PTAS(3M/4)$, we call S_i *bad* if S_i contains a job with processing time at least $\frac{2}{3}\text{opt}'_{3M/4}$. For the packing step, We will run Partition I on $M/4$ job sets, try to avoid a bad set if possible and then schedule the bags on machines. For the scheduling step, we will have several cases, based on how many bad machines that we have. The details appear in Figure 7.

LEMMA 3.6. *If $\text{opt}'_M < \frac{3b}{5}$ and $\text{opt}'_{3M/4} \leq 4b/5$ then Algorithm 2 is $\frac{5}{3}(1+\epsilon)$-robust.*

Proof. For $m \leq 3M/4$, the load on each machine is at most $\text{opt}'_{3M/4} \leq 4b/5$, hence we have,

$$\frac{ALG(m,M)}{\text{opt}_m} \leq \frac{\text{opt}'_{3M/4}}{\text{opt}_M} \leq \frac{4b/5}{\frac{b}{2(1+\epsilon)}} < (1+\epsilon)\frac{5}{3}.$$

By Lemma 3.1, for $k > \max\{v, M/2\}$, $p(B_k) \leq p(A_k) \leq \frac{2}{3}\text{opt}'_{3M/4}$. If $v > M/2$, and $v \leq m < 3M/4$ or $v \leq M/2$ and $M/2 \leq m < 3M/4$, the load on each machine is at most $\text{opt}'_{3M/4} + \max_{i>v}\{p(A_i)\} \leq \frac{5}{3}\text{opt}'_{3M/4}$, so we have,

$$\frac{ALG(m,M)}{\text{opt}_m} \leq \frac{\frac{5}{3}\text{opt}'_{3M/4}}{\text{opt}_{3M/4}} \leq (1+\epsilon)\frac{5}{3}.$$

If $v > M/2$ and $M/2 \leq m < v - 1$, then since the number of jobs with processing time at least $\frac{2}{3}\text{opt}'_{3M/4}$ is at least v, $\text{opt}_{v-1} \geq \frac{4}{3}\text{opt}'_{3M/4}$. On the other hand, each machine has load at most $2\text{opt}'_{3M/4}$. So we obtain

$$\frac{ALG(m,M)}{\text{opt}_m} \leq \frac{2\text{opt}'_{3M/4}}{\text{opt}_{v-1}} \leq \frac{2\text{opt}'_{3M/4}}{\frac{4}{3}\text{opt}'_{3M/4}} < (1+\epsilon)\frac{5}{3}\ .$$

Since $\text{opt}'_M < 3b/5$, every job has processing time at most $3b/5$. Notice that for $i \leq M/2$ and $j > M/2$, $p(S_i) \leq \text{opt}'_{3M/4} \leq 4b/5$ and $p(B_j) \leq p(A_j) \leq \max\{3b/5, 2\text{opt}'_{3M/4}/3\} = 3b/5 < 2b/3$, we can apply Lemma 3.4 on the case $m < M/2$.

3.2.3 Case III: $\text{opt}'_M < \frac{3b}{5}$ and $\text{opt}'_{3M/4} > 4b/5$. This final case is the most complicated one. We say a job is *big* if it has processing time at least $b/3$. Given a set of jobs, we call the set *2-big* if it has two big jobs. We call the set *1-big* if it is not 2-big and has one big job. We call a set that is neither 1-big nor 2-big, *0-big*. Give a set S of job sets, we let $v_i(S)$ denote the number of i-big sets and we assume that we have functions *i-big(S)* which return an i-big set from S, assuming one exists. For a job set S_i, we use $S_i(j)$ to denote the jth largest job in set S_i.

The main idea for packing here is to use the optimal schedule for $M/2$ machines, but to split the jobs assigned to each machine into 2 bags. For each S_i, we want to partition it into two sets such that one is small enough and the other is not too big. We can use Partition II to achieve this goal, if and only if S_i contains at most one big job, that is, S_i is 0-big or 1-big. If S_i is 2-big and thus contain two big jobs, we try to group three 2-big job sets with one 0-big job set, and then partition them into eight sets such that four are small enough and four not too big, using Partition III. Roughly speaking, if many S_i contain two big jobs, then we can give a good lower bound on opt, else we will have a good partition. The details appear in Figure 8.

LEMMA 3.7. *If $\text{opt}'_M < 3b/5$ and $\text{opt}'_{3M/4} > 4b/5$, Algorithm 3 is $\frac{5}{3}(1+\epsilon)$-robust.*

Proof. Since $\text{opt}'_M < 3b/5$, every job has processing time at most $3b/5$. For $k = 1, 2, 3, \ldots, 4u$, by Lemma 3.2, we have $p(A_k) \leq 5b/6$ and $p(B_k) \leq b/2$ and if moreover $k = 4t+1$ or $4t+2$ for an integer t, $p(B_k) + p(B_{k+2}) \leq 5b/6$ and we call such B_k and B_{k+2}

Algorithm 3:

Packing:

1 Let $S = PTAS(\frac{M}{2}) = \{S_1, .., S_{\frac{M}{2}}\}$,
$u = \min\{\left\lfloor \frac{v_2(S)}{3} \right\rfloor, v_0(S)\}$,
$w = \max\{v_2(S) - 3u, 0\}$ and $S' = S$.

2 **if** $u \geq 1$, **for** $k = 1, 5, 9, \ldots, 4u-3$,
$C = 2\text{-big}(S'); D = 0\text{-big}(S'); E = 2\text{-big}(S')$;
$F = 2\text{-big}(S')$; $(A_k, \ldots, A_{k+3}, B_k, \ldots, B_{k+3})$
= Partition III(C, D, E, F); $S' = S' - C - D - E - F$.

3 **if** $w \geq 1$, **for** $k = 4u+1, 4u+2, \ldots, 4u+w$,
$C = 2\text{-big}(S'); B_k = \{C(2)\}; A_k = C/B_k$;
$S' = S' - C$.

4 **for** $k = 4u+w+1, 4u+w+2, \ldots, \frac{M}{2}$.
Let C be any set in S'. (A_k, B_k) = Partition II(C); $S' = S' - C$.

5 Return the bags $\{A_1, A_2, \ldots, A_{\frac{M}{2}}, B_1, \ldots, B_{\frac{M}{2}}\}$.

Scheduling:

1 **if** $m \geq \max\{3M/4, M/2 + w + 2u\}$
for $i \leq M/2$, schedule A_i on machine i;
for $i = M/2+1, M/2+3, \ldots, M/2+2u-1$;
schedule B_{2i-M-1} and B_{2i-M+1} on machine i and B_{2i-M} and B_{2i-M+2} on machine $i+1$;
for $i = M/2+2u+1, \ldots, m$, schedule the rest of the bags on those machines such that each machine contains at most two bags and if there are two bags on a machine, at most one of them is from $B_{4u+1}, \ldots, B_{4u+w}$;

2 **if** $w + 2u > M/4$ and $3M/4 \leq m < M/2 + w + 2u$
for $i \leq M/2$, schedule A_i on machine i;
for $i = M/2+1, \ldots, m$, schedule the rest of the bags on machines so that each machines contains at most two bags;

3 **if** $M/2 \leq m < 3M/4$
schedule all bags amongst the machines so that there are at most two bags on each machine and if there are 2 bags on a machine, at most one of the bag is A_i for some i.

4 **if** $m < M/2$, we follow the strategy in Lemma 3.4.

Figure 8: Algorithm 3

paired bags. For $k = 4u+1, 4u+2, .., 4u+w$, we partition a 2-big job set S_i into $\{A_k, B_k\}$ by putting the second biggest job in B_k and the rest in A_k. Hence $b/3 \leq p(B_k) \leq b/2$, $p(A_k) = p(S_i) - p(B_k) \leq b - b/3 = 2b/3$. For $k = 4u+w+1, 4u+w+2, .., \frac{M}{2}$, by Lemma 3.3 we have $p(A_k) \leq 5b/6, p(B_k) \leq b/3$ (note that after line 3 there is no 2-big job set left and hence we can use Partition II in line 4).

Next consider the scheduling process in Algorithm 4. First consider the case $m \geq \max\{3M/4, M/2 + w + 2u\}$. For $i \leq M/2$, the load of each machine is at most $\max_i\{p(A_i)\} \leq 5b/6$. For $i = M/2+1, M/2+2, \ldots, M/2+2u$, we always schedule a paired bags on the machines, so the load is still at most $5b/6$. For the rest of the machines, we schedule $B_{4u+1}, \ldots, B_{M/2}$ on them such that at most two bags on each machines and if there are two bags, at most one of them is from $B_{4u+1}, \ldots, B_{4u+w}$, hence the load is at most $b/3 + b/2 = 5b/6$. Therefore

$$\frac{ALG(m,M)}{\text{opt}_m} \leq \frac{5b/6}{\text{opt}_M} \leq \frac{5b/6}{\frac{b}{2(1+\epsilon)}} = (1+\epsilon)\frac{5}{3}.$$

Next consider the case $w + 2u > M/4$ and $3M/4 \leq m < M/2 + w + 2u$. Since $4u \leq v_2(S) + v_0(S) \leq M/2$, $2u \leq M/4$ and $v_2(S) - 3u = w \geq 1$. If $u = \lfloor v_2(S)/3 \rfloor < v_0(S)$, since $v_2(S) \geq 3u+1$, $v_2(S) = 3u+1$ or $3u+2$. If $v_2(S) = 3u+1$, then $M/4 < w + 2u = v_2(S) - u = 2u + 1$ and $M/4 \leq 2u$, but $M/2 \geq v_2(S) + v_0(S) > 3u + 1 + u$, a contradiction. If $v_2(S) = 3u+2$, then $M/4 < v_2(S) - u = 2u + 2$ and $M/4 \leq 2u + 1$, but $M/2 \geq v_2(S) + v_0(S) > 3u + 2 + u$, a contradiction. So we have $u \geq v_0(S)$ and $w + 2u = v_2(S) - u \leq v_2(S) - v_0(S)$. Note that in total we have at least $2v_2(S) + v_1(S) = M/2 + v_2(S) - v_0(S)$ jobs with processing time at least $b/3$. Hence $\text{opt}_{M/2+w+2u-1} \geq 2b/3$. On the other hand in this case the load on each machine is at most $\max\{\max_i\{A_i\}, 2\max_i\{B_i\}\} \leq b$.

$$\frac{ALG(m,M)}{\text{opt}_m} \leq \frac{b}{\text{opt}_{M/2+w+2u-1}} \leq \frac{b}{2b/3} < (1+\epsilon)\frac{5}{3}\ .$$

In the case $M/2 \leq m < 3M/4$, the load on each machine is at most $\max_i\{A_i\} + \max_i\{B_i\} \leq 5b/6 + b/2 = 4b/3$. Therefore we have

$$\frac{ALG(m,M)}{\text{opt}_m} \leq \frac{4b/3}{\text{opt}_{3M/4}} \leq (1+\epsilon)\frac{4b/3}{4b/5} = (1+\epsilon)\frac{5}{3}\ .$$

For the case when $m < M/2$, by Lemma 3.4 the claim follows.

By combining Lemma 3.5, 3.6 and 3.7, we can deduce the following lemma. The running time comes from calling the PTAS [11] 3 times, various sorting and heap data structure operations needed to implement LPT and the other algorithms. The M term is in the running time because of the need to perform operations on the M bags.

LEMMA 3.8. *When M is divisible by 4, Algorithm 1-3 together gives a $\frac{5}{3}(1+\epsilon)$-robust algorithm with running time of $O((M+n)\log n)$, where the hidden constant depends exponentially on $1/\epsilon$.*

Recall that the algorithm uses the terms $M/2$ and $3M/4$ and assumes that they are integers. When M is not divisible by 4, the algorithm and analysis still work with some small changes, which we omit here. In sum, we have the following main theorem.

THEOREM 3.1. *There exists a $(\frac{5}{3}+\epsilon)$-robust algorithm with running time of $O((M+n)\log n)$, where the hidden constant depends exponentially on $1/\epsilon$.*

4 Conclusion and Open Problems

We initiate the idea of scheduling with uncertainty in the number of machines and give several results. In this paper, we focus on the case when $m \in [1, M]$, but there still exists a gap between the lower bound of $4/3$ and the upper bound of $5/3$. It would be very interesting to close the gap. We can also generalize the idea to consider the case when $m \in [\alpha M, \beta M]$ to see whether we can give a better upper bound if we know a restriction on the possible number of machines in advance. Some of our partitioning lemmas in Section 3.1 can be generalized and may be useful in such analysis. For example, if we have a set $S = \{j_1, j_2, .., j_K\}$, with jobs satisfying $1 - \alpha b/2 \geq p(j_1) \geq p(j_2) \geq .. \geq p(j_K)$, $p(j_2) \leq \alpha b$ and $p(S) \leq b$, we can use the idea of Partition II to partition S to one set with load at most $1 - \alpha b/2$ and another set with load at most αb. Partition III can also be generalized similarly.

It would be natural to consider other scheduling problems and other objectives, such as average completion time or flow time. Makespan is really a partitioning problem, but other objectives raise additional questions regarding the ordering of the jobs, which would be interesting to study. Finally, it might be interesting to consider the case where the number of machines come from a distribution and you need to schedule to minimize the expected makespan.

Acknowledgement

We are thankful to Tsvi Kopelowitz for many helpful discussions. We thank the anonymous reviewer who pointed out the simple 2-robust algorithm mentioned in Section 1.

References

[1] S. Albers, Recent advances for a classical scheduling problem. In Proceedings of ICALP, Springer LNCS 7966, 4-14, 2013.

[2] S. Albers and M. Hellwig, Online makespan minimization with parallel schedules. In Proceedings of SWAT , Springer LNCS 8513, 13-25, 2014.

[3] S. Albers and G. Schmidt, Scheduling with unexpected machine breakdowns. Discrete Applied Mathematics, 110(2-3):85-99, 2001.

[4] J. A. Aslam, A. Rasala, C. Stein and N. E. Young, Improved Bicriteria Existence Theorems for Scheduling. In Proceedings of SODA, 846-847, 1999.

[5] L. Chen, N. Megow, R. Rischke and L. Stougie, Stochastic and Robust Scheduling in the Cloud. In Proceedings of APPROX-RANDOM, 175-186, 2015.

[6] Y. Disser, M. Klimm, N. Megow, S. Stiller, Packing a Knapsack of Unknown Capacity. In Proceedings of STACS, 276-287, 2014.

[7] L. Epstein, A. Levin, A. Marchetti-Spaccamela, N. Megow, J. Mestre, M. Skutella, and L. Stougie, Universal Sequencing on an Unreliable Machine. SIAM Journal on Computing, 41:565-586, 2012.

[8] R. L. Graham, Bounds for certain multiprocessing anomalies. Bell Syst. Tech. J. 45 , 1563-1581, 1966.

[9] R. Hassin and S. Rubinstein, Robust matchings. SIAM Journal on Discrete Mathematics 15(4):530-537, 2002.

[10] D. S. Hochbaum and D. B. Shmoys, Using dual approximation algorithms for scheduling problems: theoretical and practical results. Journal of the ACM, 34:144-162, 1987.

[11] K. Jansen, K. M. Klein, and J. Verschae, Closing the gap for makespan scheduling via sparsification technique. In Proceedings of ICALP, 72:1-72:13, 2016.

[12] J. Niño-Mora, Stochastic scheduling. Encyclopedia of Optimization, 2nd edition, C.A. Floudas and P.M. Pardalos, eds., 3818-3824. Springer, New York, 2009.

[13] J. Matuschke, M. Skutella, and J. A. Soto, Robust Randomized Matchings. In Proceedings of SODA. 1904-1915,2015

[14] N. Megow, Robustness and Approximation for Universal Sequencing. Gems of Combinatorial Optimization and Graph Algorithms 2015, 133-141.

[15] L. K. Platzman and I. John J. Bartholdi, Spacefilling curves and the planar travelling salesman problem. Journal of the ACM, 36, 719-737, 1989.

[16] A. Rasala, C. Stein, E. Torng, and P. Uthaisombut, Existence theorems, lower bounds and algorithms for scheduling to meet two objectives. Proceedings of SODA. 723-731, 2002.

[17] D. B. Shmoys and M. Sozio, Approximation algorithms for 2-stage stochastic scheduling problems. In Integer Programming and Combinatioral Optimization, 145-157, 2007.