# Efficient approximation schemes for scheduling on a stochastic number of machines

***Leah Epstein***
University of Haifa
Haifa, Israel

***Asaf Levin***
The Technion
Haifa, Israel

# Makespan scheduling with an unknown number of identical machines

- Introduced by **Stein and Zhong, 2020.**

An input consists of a set of jobs, where each job has a rational size associated with it, and an integer $M \geq 2$.

The final number of identical machines will be $\boldsymbol{m} \leq M$.

**An algorithm has two stages:**

1. **Partition jobs to M sets called bags such that each bag will be unsplittable.**
2. **After m becomes known, merge bags such that there are exactly m bags, and assign the jobs of each bag to a different machine.**

For each machine, its completion time is **the total size of jobs** of all bags assigned to it.

The objective **to be minimized** is the **makespan** which is the maximum completion time of any machine.

# A small example

Let M=4 and ten jobs of sizes **{1,2,4,7,9,11,12,14,15,16}**

- Assume that an algorithm creates the bags

{2,9,15} with the sum 26, {4,16} with the sum 20

{7,11} with the sum 18, {1,12,14} with the sum 27

- If m=M=4, the makespan is 27.
- If m=3, two bags are to be merged.

**If {4,16} is merged with {7,11}, the resulting bag has sum 38, and this is the makespan.**

- If m=2, bags are to be merged to obtain two bags.

**If the first two bags are merged and the next two bags are also merged, the makespan is 46.**

**If (instead) the first three bags are merged (and the fourth one is not merged), the makespan is 64.**

# Work on the subject

- Various simple and also non-trivial algorithms were designed. The study is of the approximation ratio: the worst-case ratio between the cost (makespan) of the algorithm and that of an optimal solution.
- The problem was generalized to **uniformly related machines** (machines with speeds), such that the number of machines is known, but the speeds are unknown.
- Identical machines are a special case where speed are binary (0 or 1).
- The problem was studied also with respect to jobs that can be split arbitrarily (so the only input is M or M and the number of bags b, if b>M). The case of jobs of size 1 was studied as well.
- Stein & Zhong 2020, Eberle, Hoeksma, Megow, Nölke, Schewior, & Simon, 2021,2023, Balkanski, Ou, Stein, & Wei 2023, Minařík & Sgall, 2023.

# An example for the adversarial model

Consider the case **M=5** where $m \in \{2, 3, 4, 5\}$

**There are ten jobs of size 1 each.**

$OPT_5 = 2$ using $\{1,1\}, \{1,1\}, \{1,1\}, \{1,1\}, \{1,1\}$.

$OPT_4 = 3$ using $\{1,1,1\}, \{1,1,1\}, \{1,1\}, \{1,1\}$.

**$OPT_i$ is the optimal makespan for i machines.**

**Case 1:** There is **a bag with at least three jobs.**

For example: $\{1,1,1\},\{1,1,1\},\{1,1\},\{1\},\{1\}$. The **makespan** for five machines will be at least **3**.

**The approximation ratio is at least 1.5.**

**Case 2:** All **five bags have the form $\{1,1\}$.**

For four machines there will be a machine with (at least) two bags, and **makespan at least 4.**

**The approximation ratio is at least 4/3.**

# Is it possible to find an optimal solution?

**A solution consists of a set of bags, where for every value of m, the algorithm computes a schedule using the bags (by assigning them to machines).**

- The computation has to run in polynomial time.
- Even if computation in exponential time is allowed, the previous example shows that it is not possible to obtain optimal solutions.
- The limitations are based on the lack of knowledge of m (similarly to online algorithms), and on running times.
  - But jobs are not presented one by one.
- Known lower bounds of the approximation ratio are based on the lack of knowledge of m.

# The stochastic model

- It is not possible to win in the adversarial model, because the adversary is strong.
  - It knows the number of machines, while the algorithm does not know it.
  - An optimal solution does not need to create bags, but it just assigns jobs to machines, and those are the bags that it uses
- In the stochastic model, every number of machines m has a probability. The cost is the expected value of the makespan, based on solutions that the algorithm would create for different values of m.
- An optimal solution also does not necessarily have an optimal solution for every value of m, since its objective is defined as for the algorithm.
- The difficulty of an algorithm is computational.

# An example

For **M=5** and **10 jobs of size 1** each:

Assume that the probability vector for **m=2,3,4,5** is

**(0.15,0.35,0.2,0.3)**

- Assume that the algorithm creates the bags:

**{1,1,1},{1,1},{1,1},{1,1},{1} (or {1,1,1},{1,1,1},{1,1},{1},{1})**

The four makespans are: 5, 4, 3, 3, and the cost of the algorithm is $0.15 \cdot 5 + 0.35 \cdot 4 + 0.2 \cdot 3 + 0.3 \cdot 3 = 3.65$.

- Assume that the algorithm creates the bags:

**{1,1},{1,1},{1,1},{1,1},{1,1}**

The four makespans are: 6, 4, 4, 2, and the cost of the algorithm is $0.15 \cdot 6 + 0.35 \cdot 4 + 0.2 \cdot 4 + 0.3 \cdot 2 = 3.7$.

- But for the probabilities (0.4, 0.2, 0.2, 0.2) the first partition into bags is better (cost 4 versus 4.4).

# Approximation schemes

**An approximation scheme** is a class of algorithms of cost (makespan) not exceeding the cost of an optimal solution times $1 + \varepsilon$ for any $\varepsilon > 0$.

- PTAS – the running time is polynomial in the input size but $\varepsilon$ is seen as a constant.

**For example, $n^{\frac{1}{\varepsilon^3}}$ is allowed.**

- EPTAS – the running time is a function of $\varepsilon$ times a polynomial in the input size.

**Here, $n^{\frac{1}{\varepsilon^3}}$ is not allowed but $\frac{1}{\varepsilon^2}^{\frac{1}{\varepsilon^3}} \cdot n^2$ is allowed.**

- FPTAS – the running time is polynomial in the input size and in $\frac{1}{\varepsilon}$

- Every algorithm for the adversarial model can be used for the stochastic model without increasing the approximation ratio, but can one do much better?
- **We saw that an approximation scheme cannot be found in the adversarial model.**
  - **But in the stochastic model it turns out to be possible**
- An FPTAS does not exist in the stochastic model unless P=NP
  - The special case with one value of m with a non-zero probability is equivalent to the problem where the number of machines is known (to be this value of m).
  - The problem is NP-hard in the strong sense.
  - This holds also for other objectives

# Approximation schemes for standard scheduling on identical machines

- The algorithm knows the number of machines and simply assigns the input jobs.
- PTAS's/EPTAS's for various objectives including makepsan: Hochbaum and Shmoys 1987, Woeginger, 1997, Alon, Azar, Woeginger, and Yadid, 1997, 1998.
  - It is not possible to use one solution for all objectives, but if the objective is known and satisfies reasonable properties, it is possible to approximate within $1+\varepsilon$ in polynomial time.
- The difficulty of finding an optimal solution is due to NP-hardness.
  - If exponential time can be used, an optimal solution can be obtained (for example by testing all possible solutions).

# Previous work and our results

Buchem, Eberle, Kasuya Rosado, Schewior, & Wiese, 2024 designed a PTAS (which is not an EPTAS) for makepsan minimization.

**We design an EPTAS using other methods.**

They also studied the maximization problem also called "The Santa Claus Problem" and designed a PTAS for this problem. In this problem the objective is to maximize the minimum machine completion time.

**We design an EPTAS for the maximization problem and also for a third objective, which is $\ell_p$ norm minimization.**

**We will discuss makespan minimization first.**

# What can be said about the cost of an optimal solution?

- This cost OPT is a convex combination of values of makespan.
- Here, $\boldsymbol{OPT_i}$ is not the optimal cost for i machines, but it is the optimal cost for i machines for an optimal partition into bags, which may be much larger for some values of i.
- Still, we can show for the largest job size $p_{\max}$ and for the number of jobs, n, that $p_{\max} \leq \boldsymbol{OPT_i} \leq \boldsymbol{n \cdot p_{max}}$
- This means in particular that $p_{\max} \leq \boldsymbol{OPT} \leq \boldsymbol{n \cdot p_{max}}$
  - Which allows us to guess OPT up to a multiplicative factor of $1 + \varepsilon$.
- This is helpful, but only the beginning of the sequence of transformations.

# Rounding and reducing the number of bag sizes and job sizes

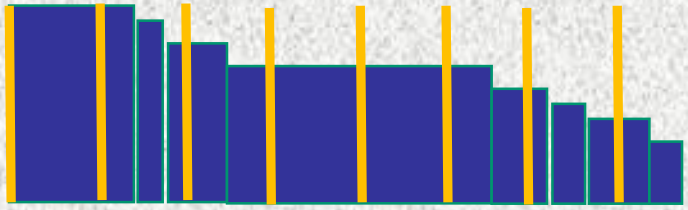
- For jobs, rounding is standard.
- Since we guessed OPT, it is not possible to have larger jobs, because this means that for every number of machines the makespan is larger than OPT, and the convex combination is larger.
- A bag size also cannot be larger than OPT, by a similar reasoning.
- A size of a bag is the total size of its jobs. It can be rounded up slightly so that there is a constant number of bag sizes that are at least $\varepsilon \cdot \mathbf{OPT}$.
- We would like to have bag sizes that are at least $\varepsilon \cdot \mathbf{OPT}$, which is achieved by merging bags. As a result some bags become empty, and we allow bags of size zero.

# Small numbers of machines

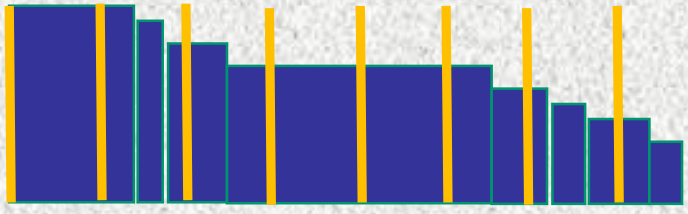
- For very small numbers of machines, a greedy assignment of every set of bags (after rounding and reducing the list of possible bag sizes) leads to a very good solution.
  - The reason is that all bags are very small compared to the makespan.
  - When the number of machines is small, every machine gets a large total size.
- A small number of machines is defined based on parameters of the problem and the required solution: Total size of jobs, $\varepsilon$, the guess of OPT.
  - Those machines are not taken into account in the following steps.

# Histogram of makespans

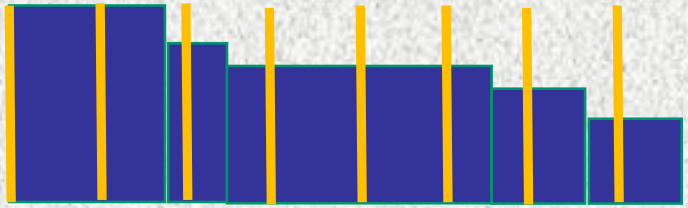
- Seeing $OPT_i$ as a function of i, this is a monotonically non-increasing function.
- It is possible to select representative values of i to reduce the number of different values of makespan.
- However, the regularity of representatives should not be based on differences between number of machines.
  - It has to be based on probabilities.
- It is possible to draw a histogram with rectangles for machines, where the width of a value of i is its probability, and select makespans with equal distances.



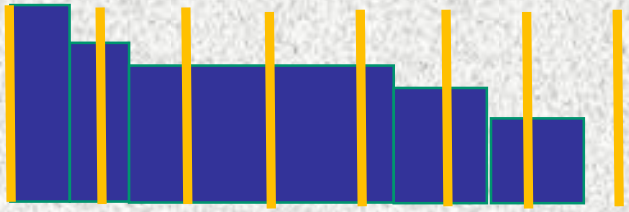
The algorithm will increase makespans slightly by extending makespans to the right. A lower bound is found by extending to the left and finding the difference.



Rounding the histogram via “Linear grouping”



The goal is to have a short list of makespans that can be guessed (enumerated)



The difference between the areas which correspond to costs is $O(\varepsilon) \cdot OPT$

# Towards an IP

- There is a constant (a function of $\varepsilon$) number of job sizes (after rounding).
- **A template of a bag consists of the number of jobs of each size that it should contain**
  - The number of templates is bounded from above because the number of bag sizes is a constant.
- A configuration of a machine consists of the number of bags of each template that it has.

- For a fixed number of machines (any $m \in \{2,3,\dots,M\}$) we **fixed a makespan**, so the configuration of each machine has to have total size not exceeding the makespan.

- Every job has to be assigned to a template of a bag, and for every m, every copy of a template needs be assigned to a configuration of the corresponding value of m.

- Expressing this as an IP allows us to solve it and find a solution (if it exists for the specific guess) efficiently.

# Differences with the maximization problem

- The objective function value for m machines is the minimum completion time of any machine.
  - The convex combination which is the final profit of the algorithm is of these values for m=2,3,…,M.
- The function $OPT_i$ is still monotonically non-increasing.
- **An easier part** is that neglecting specific machine numbers just means that bags can be assigned arbitrarily – we can assume that the profit is simply zero.
- **A harder part** is that for a value of m that we expect to get profit from, during rounding one has to keep a total sufficient size of jobs for each machine.
- It is also harder to find **suitable bounds on OPT**. The bounds are per value of m. In particular, one large job does not mean that the profit will be large.

# Additional tricks for maximization

Buchem, Eberle, Kasuya Rosado, Schewior, & Wiese, 2024 designed a PTAS, but our EPTAS uses a different approach.

**The "scenarios" for different machine numbers have four parts rather than two:**

1. **Scenarios with very large numbers of machines**, for which the profit is very small, and no gain is planned for them. This is a suffix of the list of scenarios.
2. **A consecutive sequence of scenarios with small total probability** that will separate a prefix from the most important scenarios creating a gap for the profits. It is selected based on specific guesses.
3. The resulting **prefix** for which the number of machines is small for each scenario and **a greedy schedule is sufficiently good.**
4. **The remaining scenarios** which have to be dealt with carefully.

# The differences with $\ell_p$ norm minimization

- The objective function value for a fixed value of m is the $\ell_p$ norm (for p>1) of the vector of machine completion times.
- The function $OPT_i$ is **monotonically non-increasing.**
- It is easier to find a good solution for a specific number of machines because one machine does not increase the objective much (unlike the case of makespan).
- However, bounding OPT based on job sizes is harder.
  - **A configuration of a machine of a large total could still be useful.**
  - **A large bag template can still be useful.**
- For multiple number of machines, it becomes much harder to see what is happening.

# Additional tricks for $\ell_p$ norm minimization

- Here, we design the first approximation scheme for the problem.
- Based on a histogram of costs (which are $\ell_p$ norms) for different numbers of machines, reasonable total sizes are defined for machines that have multiple bags.
  - Jobs that are very large have their own bags.
- **The scenarios are split into three parts.**
  - There is no suffix as for the maximization problem
  - The prefix is not fixed (based on parameters including guesses) as for makespan minimization, and it is found carefully but finding a middle subsequence of small total probability that will create a gap.
  - The middle sequence has to be scheduled such that it does not add a cost that is too large, unlike the maximization problem where the schedule for those scenarios does not matter.

# Discussion

- The studied problem is different from other scheduling problems:
  - The schedule is defined in two stages.
  - For the stochastic variant, a convex combination of makespan values is considered.
  - For other objectives a convex combination is analyzed as well
- Can the method of rounding the histogram be used for other problems?
- Are there other interesting objectives for the same problem?