$\frac{5}{4}(1+\epsilon)$-Approximation Algorithm for Scheduling with Rejection Costs Proportional to Processing Times

Olivier Beaumont [1] Rémi Bouzel[2] Lionel Eyraud-Dubois[1]
Esragul Korkmaz[1] Laercio Lima Pilla[1] Alexandre Van Kempen[2]

24/06/2024, 17th Scheduling for large-scale systems workshop

[1]Inria Center of the University of Bordeaux, LaBRI, UMR 5800, Talence, France

[2]Qarnot Computing, Montrouge, France

Background

Scheduling with Rejection

- Topic of the paper: Scheduling n independent jobs on m machines
    - With a possibility to **reject** some jobs
- Solution ($\mathcal{S}$)
    - Accepted jobs ($A^{\mathcal{S}}$) + associated schedule
    - Rejected jobs ($R^{\mathcal{S}}$)
- Objective
    - minimize($Z^{\mathcal{S}} = C^{\mathcal{S}} + \sum_{j \in R^{\mathcal{S}}} RejectionCost_j$)
    - where $C^{\mathcal{S}}$ denotes the makespan of jobs in $A^{\mathcal{S}}$

Scheduling with Rejection

- Many studies including
  - Bartal et al.[1]: $(2-\frac{1}{m})$-approx algo in $\mathcal{O}(n\log n)$
  - Ou et al.[2]: $(\frac{3}{2}+\epsilon)$-approx algo in $\mathcal{O}(n\log n+\frac{n}{\epsilon})$
  - Liu and Lu[3]: $(\frac{3}{2}-\frac{1}{2m})$-approx algo in $\mathcal{O}(n^3 logn)$

- **Our paper:** $\frac{5}{4}(1+\epsilon)$-approx algo in $\mathcal{O}(m^3(m+n)\log_{1+\epsilon}\rho)$
  - **Assumption**: Rejection costs proportional to processing times

[1] Bartal, Y., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J., Stougie, L.: Multi-processor scheduling with rejection. SIAM Journal Disc Math 13(1), 64-78, 2000

[2] Ou, J., Zhong, X., Wang, G.: An improved heuristic for parallel machine scheduling with rejection. European Journal of Operational Research 241(3), 653-661, 2015

[3] Liu, P., Lu, X.: New approximation algorithms for machine scheduling with rejection on single and parallel machine. Journal of Comb Optimization 40(4), 929-952, 2020

Context – Qarnot platform

- Scheduling pbs in the context of Qarnot platform (PULSE project)
- Qarnot: Cloud provider with highly distributed computational units
  - Recycling heat of computations to provide heat to the hosts
  - Avoiding energy waste on cooling systems

Traditional Datacenter

Jobs

Traditional computer

- Executing jobs

Qarnot Computing

Jobs

Boilers

- Executing jobs
- Heating water

Traditional computer

- Executing jobs

- Today: boiler maintenance scheduled at time T
- Goal: schedule maintenance close to T and minimize energy

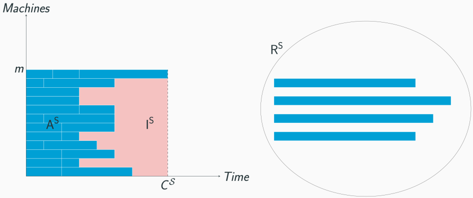
Problem Formulation - Scheduling with Rejection

- Inputs:
  - n non-preemptive independent jobs:
    - $\forall j \in \{1, ..., n\}, p_j$: processing time
    - **$\forall j \in \{1, ..., n\}, \rho p_j$: cost to run on expensive machine (ρ is fixed)**
  - m identical machines on boiler (cheap)
  - ∞ identical machines on e.g. public provider (expensive)
- Constraints:
  - Each job requires exactly one machine
  - Each machine can run at most one job at a time
  - All boiler machines are turned ON when any job running on boiler
- Outputs (Decision variables):
  - $\forall i \in \{1, ..., m\}, X_i$ is the list of jobs assigned to the cheap machine i
- Definitions:
  - $C^{\mathcal{S}} = \max_{i \in \{1,...,m\}} \sum_{j \in X_i} p_j$ (no dependencies)
  - $R^{\mathcal{S}} = \sum_{j \notin (X_i)_{i \in \{1,...,m\}}} p_j$
- Objective:

  $minimize(Z^{\mathcal{S}} = m * C^{\mathcal{S}} + \rho R^{\mathcal{S}})$

Problem Visualization - Scheduling with Rejection

- Left figure shows boiler machines vs time
  - Blue is occupied time; Pink is idle time
- Right figure shows the jobs on expensive machines (called **rejected**)

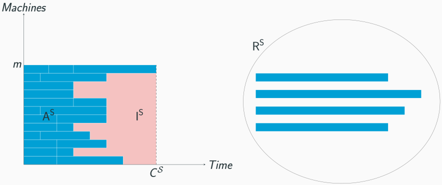


- Objective: $Z^{\mathcal{S}} = m * C^{\mathcal{S}} + \rho R^{\mathcal{S}}$
- $m * C^{\mathcal{S}}$: energy on boiler
- $\rho R^{\mathcal{S}}$: energy on traditional resources

Solution for a makespan bound T

Maintenance with target T

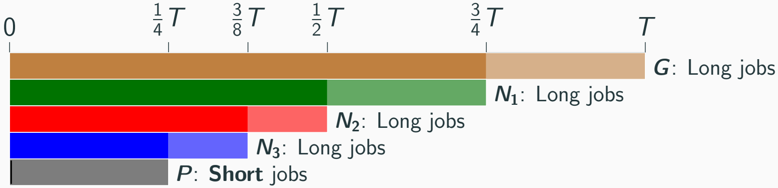
- Optimal Solution if maintenance is scheduled at time T
  - $Z^{opt} = m * T + \rho R^{opt}(T)$
- Our goal: find $C^{\mathcal{S}}$ and $R^{\mathcal{S}}$ such that



  - $Z^{\mathcal{S}} = m * C^{\mathcal{S}} + \rho R^{\mathcal{S}} \leq \frac{5}{4} Z^{opt}$

Our approach for fixed makespan T

- Idea from Caprara et al.[4] for Multiple Subset Sum Problem solution



- For fitting in T, only possible long job combinations on a machine:
  - One job possibilities: $(G), (N_1), (N_2), (N_3)$
  - Two job possibilities: $(N_1, N_2), (N_1, N_3), (N_2, N_2), (N_2, N_3), (N_3, N_3)$
  - Three job possibilities: $(N_2, N_3, N_3), (N_3, N_3, N_3)$
- Our solution for fixed makespan T:
  - Use only these combinations of long jobs (at most $\frac{5}{4}T$ makespan)
  - Finally, assign short jobs greedily within makespan $\leq \frac{5}{4}T$

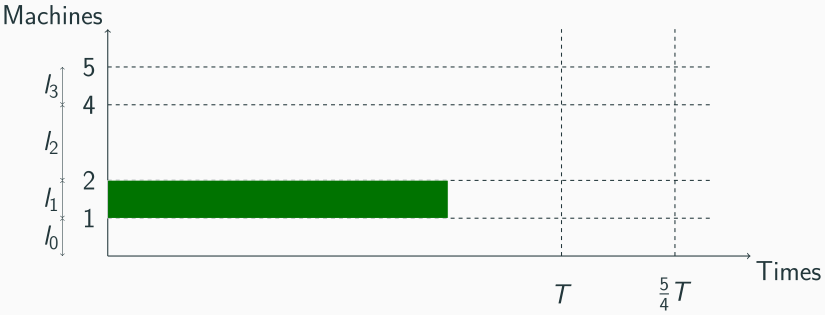
[4] Caprara, A., Kellerer, H., Pferschy, U.: A 3/4-approximation algorithm for multiple subset sum. Journal of Heuristics 9(2), 99–111 (March 2003)

Our algorithm for fixed makespan T : *FillMaxArea*

- Assume: m=5, $|G|$=0, $|N_1|$=5, $|N_2|$=1, $|N_3|$=5, $|P|$=17
- l_0, l_1, l_2 and l_3 stand for the number of machines that run no long job, one long job, two long jobs and three long jobs, respectively
- Fix $(l_0, l_1, l_2, l_3) = (1, 1, 2, 1)$

Our algorithm for fixed makespan T : *FillMaxArea*

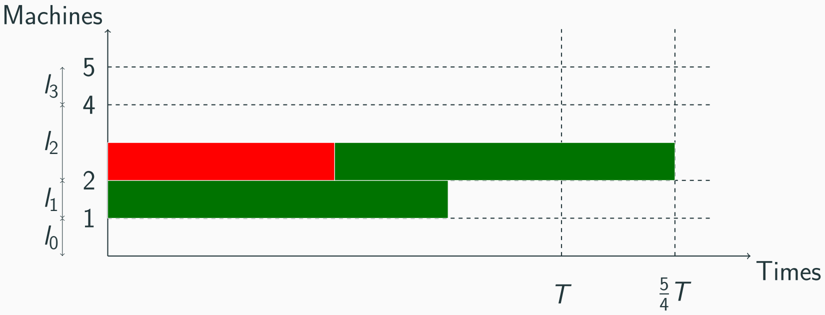
- Assume: m=5, $|G|$=0, $|N_1|$=4, $|N_2|$=1, $|N_3|$=5, $|P|$=17
- l_0, l_1, l_2 and l_3 stand for the number of machines that run no long job, one long job, two long jobs and three long jobs, respectively
- Fix $(l_0, l_1, l_2, l_3) = (1, 1, 2, 1)$



- Start with singletons, i.e. $(G), (N_1), (N_2), (N_3)$
  - No job in G $\implies$ Take the longest job in N_1

Our algorithm for fixed makespan T : *FillMaxArea*

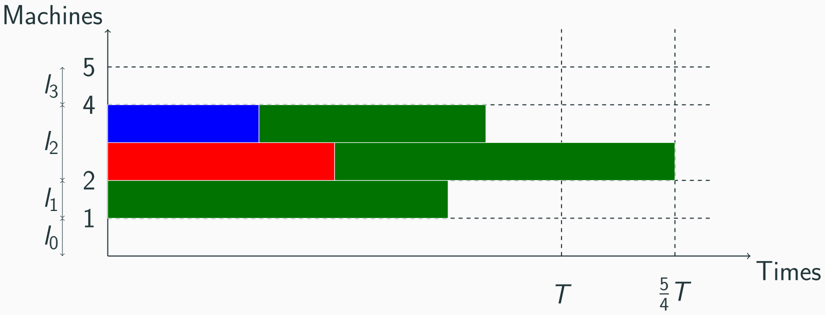
- Assume: m=5, $|G|$=0, $|N_1|$=3, $|N_2|$=0, $|N_3|$=5, $|P|$=17
- l_0, l_1, l_2 and l_3 stand for the number of machines that run no long job, one long job, two long jobs and three long jobs, respectively
- Fix $(l_0, l_1, l_2, l_3) = (1, 1, 2, 1)$



- Now assign pairs,i.e. $(N_1, N_2), (N_1, N_3), (N_2, N_2), (N_2, N_3), (N_3, N_3)$
  - Longest job in N_2 and longest job in N_1

Our algorithm for fixed makespan T : *FillMaxArea*

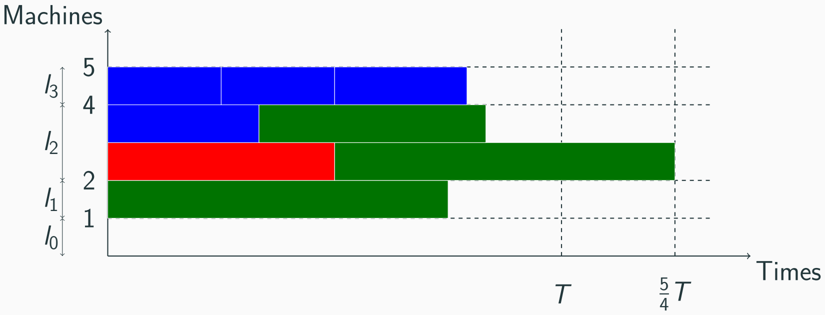
- Assume: m=5, $|G|$=0, $|N_1|$=2, $|N_2|$=0, $|N_3|$=4, $|P|$=17
- l_0, l_1, l_2 and l_3 stand for the number of machines that run no long job, one long job, two long jobs and three long jobs, respectively
- Fix $(l_0, l_1, l_2, l_3) = (1, 1, 2, 1)$



- Assign pairs,i.e. $(N_2, N_1), (N_3, N_1), (N_2, N_2), (N_3, N_2), (N_3, N_3)$
  - No job in N_2 $\implies$ Longest job in N_3 and longest job in N_1

Our algorithm for fixed makespan T : *FillMaxArea*

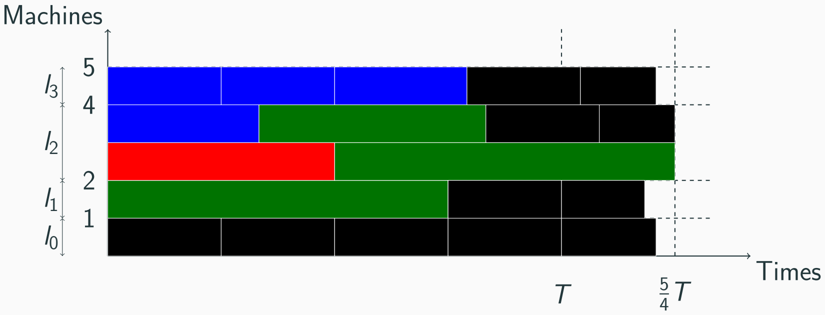
- Assume: m=5, $|G|$=0, $|N_1|$=2, $|N_2|$=0, $|N_3|$=1, $|P|$=17
- l_0, l_1, l_2 and l_3 stand for the number of machines that run no long job, one long job, two long jobs and three long jobs, respectively
- Fix $(l_0, l_1, l_2, l_3) = (1, 1, 2, 1)$



- Assign triplets, i.e. $(N_3, N_3, N_2), (N_3, N_3, N_3)$
  - No job in $N_2 \implies$ 3 longest jobs in N_3

Our algorithm for fixed makespan T : *FillMaxArea*

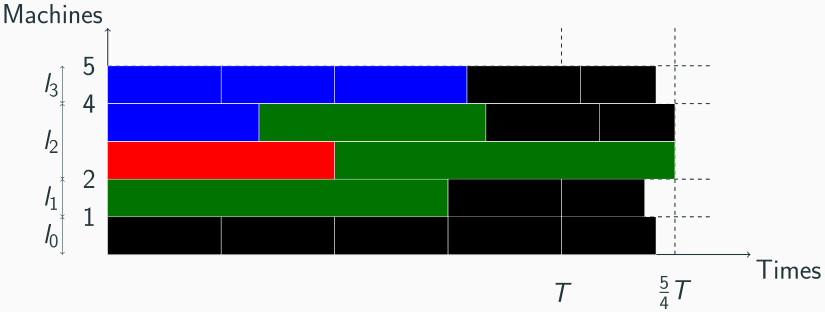
- Assume: m=5, $|G|$=0, $|N_1|$=2, $|N_2|$=0, $|N_3|$=1, $|P|$=6
- l_0, l_1, l_2 and l_3 stand for the number of machines that run no long job, one long job, two long jobs and three long jobs, respectively
- Fix $(l_0, l_1, l_2, l_3) = (1, 1, 2, 1)$



- Assign P jobs greedily while makespan $\leq \frac{5}{4}T$

Our algorithm for fixed makespan T : *FillMaxArea*

- Assume: m=5, $|G|$=0, $|N_1|$=2, $|N_2|$=0, $|N_3|$=1, $|P|$=6
- l_0, l_1, l_2 and l_3 stand for the number of machines that run no long job, one long job, two long jobs and three long jobs, respectively
- Fix $(l_0, l_1, l_2, l_3) = (1, 1, 2, 1)$



- Find solution for each $(l_0, l_1, l_2, l_3) \rightarrow \mathcal{O}(m^3)$ iterations
  - Choose solution with largest $\sum_{j \in AcceptedJobs} p_j$

Overall algorithm:

- Iterate above algorithm for all l_0, l_1, l_2, l_3 such that $\sum l_i = m$
- there are "only" $\Theta(m^3)$ such quadruplets...
- and keep the solution that minimizes $Z^{\mathcal{S}} = m * C^{\mathcal{S}} + \rho R^{\mathcal{S}}$

RESULT

For any T, let $\dagger$ be the solution obtained by *FillMaxArea*(J, m, T). Then, $C^{\dagger} \leq \frac{5}{4} T$ and $R^{\dagger} \leq R^*(T)$, where $R^*(T) = \min_{\mathcal{S}, C^{\mathcal{S}} \leq T} R^{\mathcal{S}}$.

FillMaxArea - **Main result**

Overall algorithm:

- Iterate above algorithm for all l_0, l_1, l_2, l_3 such that $\sum l_i = m$
- there are "only" $\Theta(m^3)$ such quadruplets...
- and keep the solution that minimizes $Z^{\mathcal{S}} = m * C^{\mathcal{S}} + \rho R^{\mathcal{S}}$

RESULT

For any T, let $\dagger$ be the solution obtained by *FillMaxArea*(J, m, T). Then, $C^{\dagger} \leq \frac{5}{4} T$ and $R^{\dagger} \leq R^*(T)$, where $R^*(T) = \min_{\mathcal{S}, C^{\mathcal{S}} \leq T} R^{\mathcal{S}}$.

Sketch of Proof:

- $C^{\dagger} \leq \frac{5}{4} T$: straightforward given the upper bounds of combinations
- *FillMaxArea* allocates as much work as possible on the boiler given l_0, l_1, l_2, l_3
  - $R^{\dagger} \leq R^*(T)$

$\square$

RESULT

For any T, let $\dagger$ be the solution obtained by *FillMaxArea*(J, m, T). Then, $C^\dagger \leq \frac{5}{4}T$ and $R^\dagger \leq R^*(T)$, where $R^*(T) = \min_{\mathcal{S}, C^{\mathcal{S}} \leq T} R^{\mathcal{S}}$.

LEMMA

For any T, let $\dagger$ be the solution obtained by *FillMaxArea*(J, m, T). We can bound its cost by: $Z^\dagger \leq \frac{5}{4}Tm + \rho R^*(T)$.

RESULT

For any T, let $\dagger$ be the solution obtained by *FillMaxArea*(J, m, T). Then, $C^\dagger \leq \frac{5}{4}T$ and $R^\dagger \leq R^*(T)$, where $R^*(T) = \min_{\mathcal{S}, C^{\mathcal{S}} \leq T} R^{\mathcal{S}}$.

LEMMA

For any T, let $\dagger$ be the solution obtained by *FillMaxArea*(J, m, T). We can bound its cost by: $Z^\dagger \leq \frac{5}{4}Tm + \rho R^*(T)$.

Proof:

- Remember: Cost of $\dagger$ is $Z^\dagger = m * C^\dagger + \rho R^\dagger$
- $C^\dagger \leq \frac{5}{4}T$
- $R^\dagger \leq R^*(T)$
    - $Z^\dagger \leq \frac{5}{4}Tm + \rho R^*(T) \leq \frac{5}{4}Z^{opt}$

$\square$

Determining the makespan bound T

Determining Makespan: BEKP method

- If C^{OPT} is known $\implies$ *FillMaxArea*(C^{OPT}) is a $\frac{5}{4}$-approx. algorithm
- In the general problem C^{OPT} is not known...
- Assume: Lower and Upper bound (L and U) on makespan such that
  - Rejecting all jobs when makespan $C^{OPT} \leq L$ and $C^{OPT} \geq U$ is valid
- Iterate over remaining makespans

L $L(1+\epsilon)$ ... $L(1+\epsilon)^k \geq U$

- ϵ: User defined small coefficient
- Iteration number: $k = \lceil \frac{log(\frac{U}{L})}{log(1+\epsilon)} \rceil$

Determining Makespan: BEKP method

Algorithm 1 $\mathcal{BEKP}(J, m)$

1: The solution where all jobs are rejected is X_0
2: Compute U and L
3: Set k as the first value such that $(1+\epsilon)^k L \geq U$
4: **for** each $C_i \in \{L, (1+\epsilon)L, (1+\epsilon)^2 L, ..., (1+\epsilon)^k L\}$ **do**
5: $\quad X_i = \mathit{FillMaxArea}(J, m, C_i)$
6: **return** A schedule with the lowest cost among X_0 and all X_i

Determining Makespan: BEKP method

Algorithm 2 $\mathcal{BEKP}(J, m)$

1. The solution where all jobs are rejected is X_0
2. Compute U and L
3. Set k as the first value such that $(1+\epsilon)^k L \geq U$
4. **for** each $C_i \in \{L, (1+\epsilon)L, (1+\epsilon)^2 L, ..., (1+\epsilon)^k L\}$ **do**
5. $\quad X_i = \mathit{FillMaxArea}(J, m, C_i)$
6. **return** A schedule with the lowest cost among X_0 and all X_i

WHAT IS LEFT?

- Compute L and U
- Proof: *BEKP* is a $\frac{5}{4}(1+\epsilon)$-approximation algorithm.

Computing L and U

- W: sum of all processing times of jobs in ready queue
- $R^*(C)$: minimum possible area for the rejected jobs for makespan C
- Lower bound function on the cost: $f(C) = Cm + \rho R^*(C)$

Computing L and U

- W: sum of all processing times of jobs in ready queue
- $R^*(C)$: minimum possible area for the rejected jobs for makespan C
- Lower bound function on the cost: $f(C) = Cm + \rho R^*(C)$
- $f(C) \geq Cm$

Computing L and U

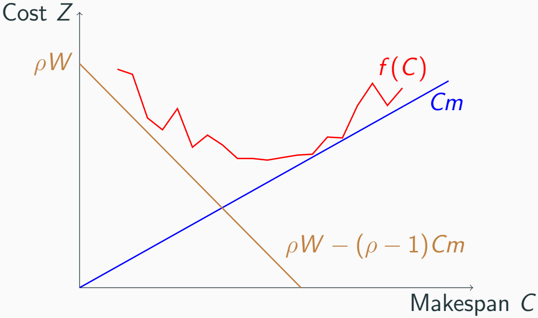
- W: sum of all processing times of jobs in ready queue
- $R^*(C)$: minimum possible area for the rejected jobs for makespan C
- Lower bound function on the cost: $f(C) = Cm + \rho R^*(C)$

- $f(C) \geq Cm$
- $f(C) \geq Cm + \rho(W - Cm)$

Computing L and U

- W: sum of all processing times of jobs in ready queue
- $R^*(C)$: minimum possible area for the rejected jobs for makespan C
- Lower bound function on the cost: $f(C) = Cm + \rho R^*(C)$

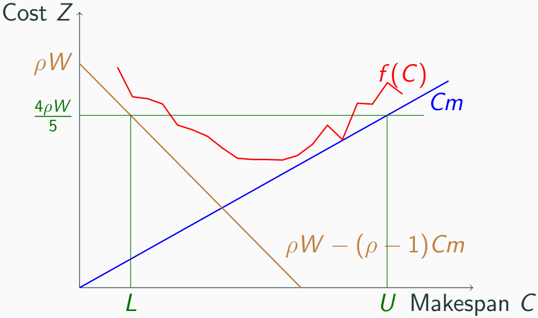
- $f(C) \geq Cm$
- $f(C) \geq Cm + \rho(W - Cm)$



Computing L and U

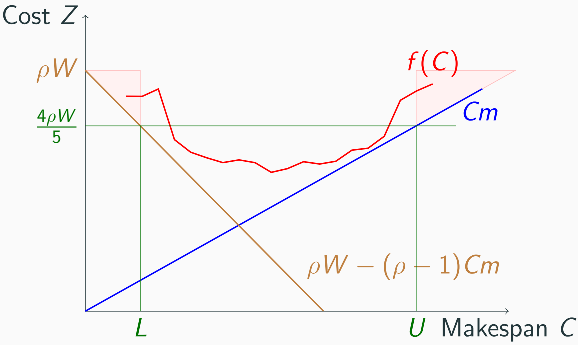
- W: sum of all processing times of jobs in ready queue
- $R^*(C)$: minimum possible area for the rejected jobs for makespan C
- Lower bound function on the cost: $f(C) = Cm + \rho R^*(C)$

- $f(C) \geq Cm$
- $f(C) \geq Cm + \rho(W - Cm)$



- $H = \frac{4\rho W}{5}$ (We set this value)

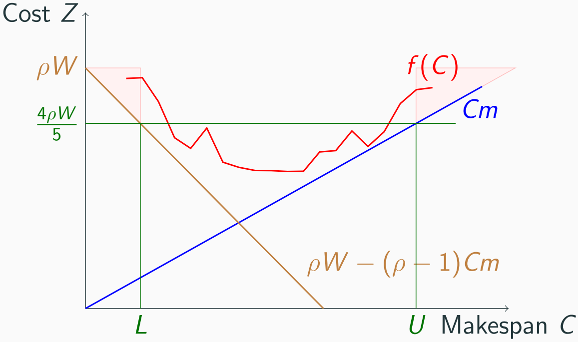
Computing L and U



- If we reject all jobs for makespan values $\leq L$ and $\geq U$, worst cost:

$\frac{Z^{X_0}}{Z^{OPT}} \leq \frac{\rho W}{\frac{4\rho W}{5}} = \frac{5}{4}$ (Valid solution)

Computing L and U



- If we reject all jobs for makespan values $\leq L$ and $\geq U$, worst cost:

$\frac{Z^{X_0}}{Z^{OPT}} \leq \frac{\rho W}{\frac{4\rho W}{5}} = \frac{5}{4}$ (Valid solution)

- Remember: $k = \lceil \frac{log(\frac{U}{L})}{log(1+\epsilon)} \rceil$
- $U = \frac{4\rho W}{5m}$ and $L = \frac{\rho W}{5m(\rho-1)} \implies \frac{U}{L} = 4(\rho - 1)$

For $\rho \leq 10$ and $\epsilon \geq 0.05$, $k = 74$ (Cheap iteration)

Proof of the approximation ratio

THEOREM

For any positive ϵ, $\mathcal{BEKP}$ is a $\frac{5}{4}(1+\epsilon)$-approximation algorithm.

Proof

- For $C^{OPT} < L$ or $C^{OPT} > U$ already proved.

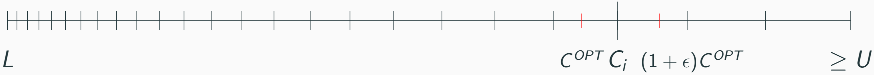
Proof of the approximation ratio

THEOREM

For any positive ϵ, $\mathcal{BEKP}$ is a $\frac{5}{4}(1+\epsilon)$-approximation algorithm.

Proof

- For $C^{OPT} < L$ or $C^{OPT} > U$ already proved. **For $\boldsymbol{L \leq C^{OPT} \leq U}$**:



Proof of the approximation ratio

THEOREM

For any positive ϵ, $\mathcal{BEKP}$ is a $\frac{5}{4}(1+\epsilon)$-approximation algorithm.

Proof

- For $C^{OPT} < L$ or $C^{OPT} > U$ already proved. **For $\boldsymbol{L \leq C^{OPT} \leq U}$:**

- X_i: Solution by using C_i
- $Z^{X_i} \leq \frac{5}{4}mC_i + \rho R^*(C_i)$

Proof of the approximation ratio

THEOREM

For any positive ϵ, $\mathcal{BEKP}$ is a $\frac{5}{4}(1+\epsilon)$-approximation algorithm.

Proof

- For $C^{OPT} < L$ or $C^{OPT} > U$ already proved. **For $\boldsymbol{L \leq C^{OPT} \leq U}$**:

- X_i: Solution by using C_i
- $Z^{X_i} \leq \frac{5}{4}mC_i + \rho R^*(C_i)$
- As $C^{OPT} \leq C_i$
  - $R^*(C_i) \leq R^*(C^{OPT})$

Proof of the approximation ratio

THEOREM

For any positive ϵ, $\mathcal{BEKP}$ is a $\frac{5}{4}(1+\epsilon)$-approximation algorithm.

Proof

- For $C^{OPT} < L$ or $C^{OPT} > U$ already proved. **For $\boldsymbol{L \leq C^{OPT} \leq U}$:**

- X_i: Solution by using C_i
- $Z^{X_i} \leq \frac{5}{4}mC_i + \rho R^*(C_i)$
- As $C^{OPT} \leq C_i$
  - $R^*(C_i) \leq R^*(C^{OPT})$
- As $C_i \leq (1+\epsilon)C^{OPT}$
  - $Z^{X_i} \leq \frac{5}{4}(1+\epsilon)mC^{OPT} + \rho R^*(C^{OPT}) \leq \frac{5}{4}(1+\epsilon)Z^{OPT}$

$\square$

Conclusion - Future work

Conclusion

- In practice (simulations)
  - Very good qualitative results (much lower than $\frac{5}{4}$)
  - Low computational time ($Km^3(m+n)$ where m machines, n jobs)
- $\mathcal{BEKP}$ compared to $\mathcal{LIULU}$
  - Improves approximation ratio (for proportional rejection costs)
  - Proportional rejection costs can be extended to other contexts
  - Improves complexity wrt total number of jobs
- Next step: consider other scheduling problems related to Qarnot
  - With deadlines (each job has a duration and a deadline)
  - the true problem is online and non clairvoyant...
    - probably very difficult to prove something...

Thank You

Backup slides

Experiments

Experimental setting

- Run in sequential on Miriel nodes of Plafrim
  - 2 INTEL Xeon E5-2680v3 12-core 2.50 GHz processors with 128 GB
- Processing times are generated through lognormal distribution
  - Mean 3
  - Standard deviations (σ): "0.5", "0.7", and "1.0"
  - Large σ means higher variance in processing times
- Number of machines: $m = 20$
- Number of jobs: $n = 4m$
- Our simulation code is available as free software in[5]

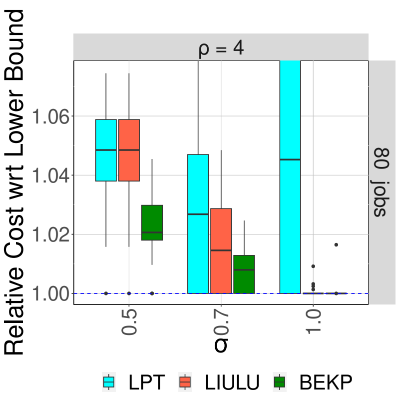
[5] Beaumont, O., Eyraud-Dubois, L., Korkmaz, E., Pilla, L.L.: Experimental codes and results for the paper "a 5/4($1+\epsilon$)-approximation algorithm for scheduling with rejection costs proportional to processing times". `https://inria.hal.science/hal-04517532`, accessed: March 25, 2024

Methods compared

- $\mathcal{BEKP}$: Our method
  - $\mathcal{O}(m^3(m+n)\log_{1+\epsilon}\rho)$
  - $\frac{5}{4}(1+\epsilon)$ approximation
- $\mathcal{LIULU}$: The algorithm proposed by Liu and Lu[6]
  - $\mathcal{O}(n^3 \log n)$
  - $(\frac{3}{2}-\frac{1}{2m})$ approximation
  - Assumes arbitrary rejection costs
- $\mathcal{LPT}$: A cheap and naive solution (No rejection)
  - Uses Longest Processing Time first method

[6] Liu, P., Lu, X.: New approximation algorithms for machine scheduling with rejection on single and parallel machine. Journal of Comb Optimization 40(4), 929-952 (2020)

Comparison of scheduler costs



- Each box-plot represents 30 different experimental cases
- $\mathcal{LPT}$: No guarantee on the cost bound
- $\mathcal{BEKP}$: Better costs in general compared to $\mathcal{LIULU}$

Our algorithm for fixed makespan T

Algorithm 3 *FillMaxArea*(J, m, T)

1: Generate G, N_1, N_2, N_3 and P subsets of J

2: **for** each $j = (l_0, l_1, l_2, l_3)$ such that $l_0 + l_1 + l_2 + l_3 = m$ and $l_1 + 2l_2 + 3l_3 \leq n$ **do**

3: $\quad X_j \leftarrow \emptyset$

4: $\quad X_j \leftarrow X_j \cup AssignFrom(\{(G), (N_1), (N_2), (N_3)\}, l_1)$

5: $\quad X_j \leftarrow X_j \cup AssignFrom(\{(N_2, N_1), (N_3, N_1), (N_2, N_2), (N_3, N_2), (N_3, N_3)\}, l_2)$

6: $\quad X_j \leftarrow X_j \cup AssignFrom(\{(N_3, N_3, N_2), (N_3, N_3, N_3)\}, l_3)$

7: $\quad$ If $l_0 + |X_j| < m$, discard X_j

8: $\quad$ Add jobs from P greedily (in any order) to X_j, keeping makespan $\leq \frac{5}{4}T$

9: $X^* = \{X_j | \max_j A^{X_j}\}$

10: **return** X^*

Our algorithm for fixed makespan T

Algorithm 4 *AssignFrom*(*combs*, *l*)

$Result \leftarrow \emptyset$
Remove all combinations from *combs* where at least one set within the combination is empty
while $|Result| \leq l$ and *combs* is not empty **do**
    Denote by $(K_1, K_2, ..., K_k)$ the first combination in *combs*
    $j_1 \leftarrow$ the largest job from K_1, $j_2 \leftarrow$ the largest remaining job from K_2 ...
    Continue until $j_k \leftarrow$ the largest remaining job from K_k
    $Result = Result \cup (j_1, j_2, ..., j_k)$
    Remove all combinations from *combs* where at least one set within the combination is empty
return *Result*

Comparison of real-life scheduler running times

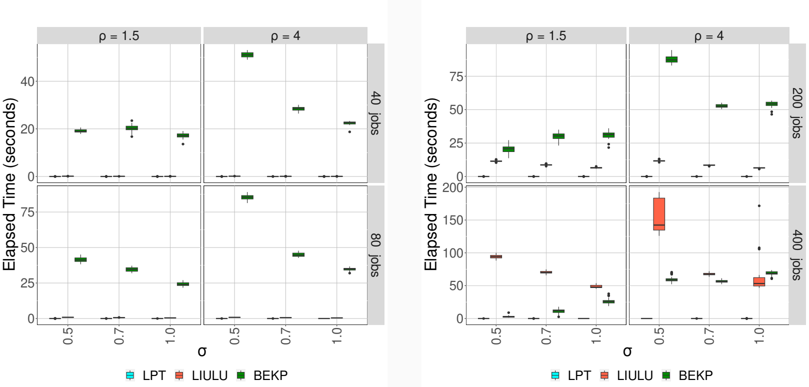


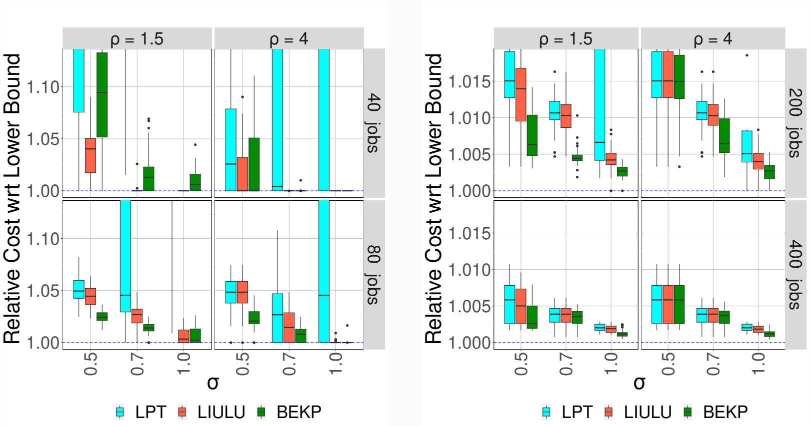
Figure 1: Comparison of $\mathcal{LPT}$, $\mathcal{LIULU}$ and $\mathcal{BEKP}$ using $m = 20$ for different number of jobs, different values for ρ and σ. Each box-plot represents 30 different experimental cases for the corresponding configuration.

Methods compared

- $\mathcal{BEKP}$: Our method
  - $\mathcal{O}(m^3(m+n)\log_{1+\epsilon}\rho)$
  - $\frac{5}{4}(1+\epsilon)$ approximation
- $\mathcal{LIULU}$: The algorithm proposed by Liu and Lu[7]
  - $\mathcal{O}(n^3 \log n)$
  - $(\frac{3}{2} - \frac{1}{2m})$ approximation
  - Assumes arbitrary rejection costs
- $\mathcal{LPT}$: A cheap and naive solution (No rejection)
  - Uses Longest Processing Time first method
- Lower Bound: Reference method
  - ILP: $x_i = 1$ if job is accepted; else $x_i = 0$
  - $\forall i \in J, C \geq x_i p_i$ and $C \geq \sum_{i \in J}(x_i p_i)/m$
  - minimize $Cm + \sum_{i \in J} \rho(1 - x_i)p_i$
- No performance optimization for methods

[7] Liu, P., Lu, X.: New approximation algorithms for machine scheduling with rejection on single and parallel machine. Journal of Comb Optimization 40(4), 929-952 (2020)

Comparison of scheduler costs



- Each box-plot represents 30 different experimental cases
- $\mathcal{LPT}$: No guarantee on the cost bound
- $\mathcal{BEKP}$: Better costs in general compared to $\mathcal{LIULU}$