Worst Case Bound of LRF Schedule for Fully Parallel Jobs

Kai WANG

Department of Computer Science
City University of Hong Kong

March 27, 2016

Joint work with *Vincent Chau*
Supervisor: *Minming Li*

Outline

- Introduction
- Formulation
- Contribution
- Research Work
  - Special case: Instance of jobs with equal density
  - General case: Instance of jobs with arbitrary weights
- Conclusion & Future Work

Introduction

1. Many companies have containers to be shipper out.
2. A cargo ship can delivery C containers per journey.
3. As long as one container can not be shipped out today, the company has to wait another day.

How to delivery the containers from different companies to minimize the total completion time?

Formulation

- Given a set J of n jobs, for each job j,
  - *Fully Parallel*, processed on any machine at any moment.
  - s_j, the workload, $s_j \in \mathbb{N}$.
  - w_j, the weight (importance), $w_j \in \mathbb{R}$.
  - released at time zero.
- Given m identical machines, for each machine i,
  - finish one job of one unit workload during one unit time.
- A feasible schedule is a table M,
  - $M(i,t)$: the job executed on machine i during time unit $[t-1,t)$,
  - Job completion time $C_j = \max_{M(i,t)=j} t$.
- Objective: minimize $T = \sum_{j \in J} w_j C_j$.

Formulation

- Given a set J of n jobs, for each job j,
  - *Fully Parallel*, processed on any machine at any moment.
  - s_j, the workload, $s_j \in \mathbb{N}$.
  - w_j, the weight (importance), $w_j \in \mathbb{R}$.
  - released at time zero.
- Given m identical machines, for each machine i,
  - finish one job of one unit workload during one unit time.
- A feasible schedule is a table M,
  - $M(i,t)$: the job executed on machine i during time unit $[t-1,t)$,
  - Job completion time $C_j = \max_{M(i,t)=j} t$.
- Objective: minimize $T = \sum_{j \in J} w_j C_j$.

Formulation

- Given a set J of n jobs, for each job j,
  - *Fully Parallel*, processed on any machine at any moment.
  - s_j, the workload, $s_j \in \mathbb{N}$.
  - w_j, the weight (importance), $w_j \in \mathbb{R}$.
  - released at time zero.
- Given m identical machines, for each machine i,
  - finish one job of one unit workload during one unit time.
- A feasible schedule is a table M,
  - $M(i,t)$: the job executed on machine i during time unit $[t-1,t)$.
  - Job completion time $C_j = \max\limits_{M(i,t)=j} t$.
- Objective: minimize $T = \sum_{j \in J} w_j C_j$.

Formulation

- Given a set J of n jobs, for each job j,
  - *Fully Parallel*, processed on any machine at any moment.
  - s_j, the workload, $s_j \in \mathbb{N}$.
  - w_j, the weight (importance), $w_j \in \mathbb{R}$.
  - released at time zero.
- Given m identical machines, for each machine i,
  - finish one job of one unit workload during one unit time.
- A feasible schedule is a table M,
  - $M(i,t)$: the job executed on machine i during time unit $[t-1,t)$,
  - Job completion time $C_j = \max_{M(i,t)=j} t$.
- Objective: minimize $T = \sum_{j \in J} w_j C_j$.

Formulation

- Given a set J of n jobs, for each job j,
  - *Fully Parallel*, processed on any machine at any moment.
  - s_j, the workload, $s_j \in \mathbb{N}$.
  - w_j, the weight (importance), $w_j \in \mathbb{R}$.
  - released at time zero.
- Given m identical machines, for each machine i,
  - finish one job of one unit workload during one unit time.
- A feasible schedule is a table M,
  - $M(i,t)$: the job executed on machine i during time unit $[t-1,t)$,
  - Job completion time $C_j = \max\limits_{M(i,t)=j} t$.
- Objective: minimize $T = \sum_{j \in J} w_j C_j$.

Formulation

- Given a set J of n jobs, for each job j,
  - *Fully Parallel*, processed on any machine at any moment.
  - s_j, the workload, $s_j \in \mathbb{N}$.
  - w_j, the weight (importance), $w_j \in \mathbb{R}$.
  - released at time zero.
- Given m identical machines, for each machine i,
  - finish one job of one unit workload during one unit time.
- A feasible schedule is a table M,
  - $M(i,t)$: the job executed on machine i during time unit $[t-1,t)$,
  - Job completion time $C_j = \max_{M(i,t)=j} t$.
- Objective: minimize $T = \sum_{j \in J} w_j C_j$.

Formulation

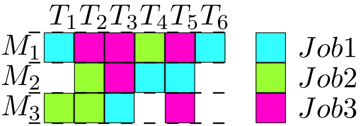
- Given a set J of n jobs, for each job j,
  - *Fully Parallel*, processed on any machine at any moment.
  - s_j, the workload, $s_j \in \mathbb{N}$.
  - w_j, the weight (importance), $w_j \in \mathbb{R}$.
  - released at time zero.
- Given m identical machines, for each machine i,
  - finish one job of one unit workload during one unit time.
- A feasible schedule is a table M,
  - $M(i,t)$: the job executed on machine i during time unit $[t-1,t)$.
  - Job completion time $C_j = \max_{M(i,t)=j} t$.
- Objective: minimize $T = \sum_{j \in J} w_j C_j$.

Formulation

- Given a set J of n jobs, for each job j,
  - *Fully Parallel*, processed on any machine at any moment.
  - s_j, the workload, $s_j \in \mathbb{N}$.
  - w_j, the weight (importance), $w_j \in \mathbb{R}$.
  - released at time zero.
- Given m identical machines, for each machine i,
  - finish one job of one unit workload during one unit time.
- A feasible schedule is a table M,
  - $M(i,t)$: the job executed on machine i during time unit $[t-1,t)$,
  - Job completion time $C_j = \max\limits_{M(i,t)=j} t$.
- Objective: minimize $T = \sum_{j \in J} w_j C_j$.

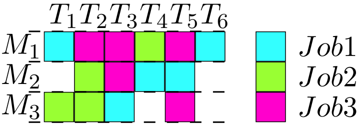
Formulation

- Given a set J of n jobs, for each job j,
  - *Fully Parallel*, processed on any machine at any moment.
  - s_j, the workload, $s_j \in \mathbb{N}$.
  - w_j, the weight (importance), $w_j \in \mathbb{R}$.
  - released at time zero.
- Given m identical machines, for each machine i,
  - finish one job of one unit workload during one unit time.
- A feasible schedule is a table M,
  - $M(i,t)$: the job executed on machine i during time unit $[t-1,t)$,
  - Job completion time $C_j = \max\limits_{M(i,t)=j} t$.
- Objective: minimize $T = \sum_{j \in J} w_j C_j$.



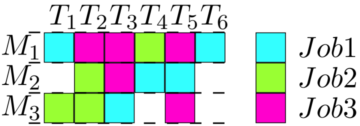
Formulation

- Given a set J of n jobs, for each job j,
  - *Fully Parallel*, processed on any machine at any moment.
  - s_j, the workload, $s_j \in \mathbb{N}$.
  - w_j, the weight (importance), $w_j \in \mathbb{R}$.
  - released at time zero.
- Given m identical machines, for each machine i,
  - finish one job of one unit workload during one unit time.
- A feasible schedule is a table M,
  - $M(i,t)$: the job executed on machine i during time unit $[t-1,t)$,
  - Job completion time $C_j = \max\limits_{M(i,t)=j} t$.
- Objective: minimize $T = \sum_{j \in J} w_j C_j$.

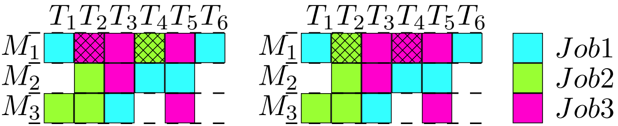


Formulation

- Given a set J of n jobs, for each job j,
  - *Fully Parallel*, processed on any machine at any moment.
  - s_j, the workload, $s_j \in \mathbb{N}$.
  - w_j, the weight (importance), $w_j \in \mathbb{R}$.
  - released at time zero.
- Given m identical machines, for each machine i,
  - finish one job of one unit workload during one unit time.
- A feasible schedule is a table M,
  - $M(i,t)$: the job executed on machine i during time unit $[t-1,t)$,
  - Job completion time $C_j = \max\limits_{M(i,t)=j} t$.
- Objective: minimize $T = \sum_{j \in J} w_j C_j$.

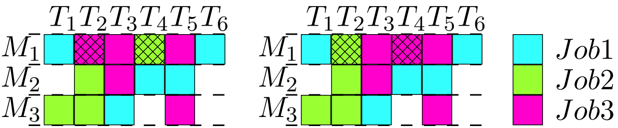


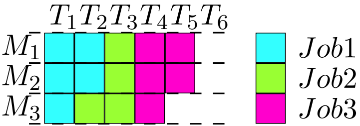
- Observation: Jobs should be scheduled consecutively.
- $T = \sum_{j \in J} w_j \lceil \frac{\sum_{i \preceq j} s_i}{m} \rceil$.



Formulation

- Observation: Jobs should be scheduled consecutively.
- $T = \sum_{j \in J} w_j \lceil \frac{\sum_{i \preceq j} s_i}{m} \rceil$.





Contribution

- Related works: first introduced by Zhang et al. [TCS 2013]
  - Strongly NP-hard when m is input.
  - Proposed a 2-approximation algorithm, *Largest-Ratio-First* (LRF) algorithm.
- LRF algorithm: schedule the job with largest ratio w/s first.
- Our contribution: the LRF algorithm is α - approximation,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$ for instance of jobs with equal density $w_j/s_j = 1$, where $i = \lceil \frac{2n}{m} \rceil$.
    - i) $1 + \frac{1}{i+1} \leq \alpha < 1 + \frac{1}{i}$.
    - ii) α is tight for some group of instance;
    - iii) we give tight upper bound of α for different group of instances.
  - $\alpha = 1 + \frac{m-1}{m+2}$, for general case.

Contribution

- Related works: first introduced by Zhang et al. [TCS 2013]
  - Strongly NP-hard when m is input.
  - Proposed a 2-approximation algorithm, *Largest-Ratio-First* (LRF) algorithm.
- LRF algorithm: schedule the job with largest ratio w/s first.
- Our contribution: the LRF algorithm is α - approximation,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$ for instance of jobs with equal density $w_j/s_j = 1$, where $i = \lceil \frac{2n}{m} \rceil$.
    - i) $1 + \frac{1}{i+1} \leq \alpha < 1 + \frac{1}{i}$.
    - ii) α is tight for some group of instance;
    - iii) we give tight upper bound of α for different group of instances.
  - $\alpha = 1 + \frac{m-1}{m+2}$, for general case.

Contribution

- Related works: first introduced by Zhang et al. [TCS 2013]
  - Strongly NP-hard when m is input.
  - Proposed a 2-approximation algorithm, *Largest-Ratio-First* (LRF) algorithm.
- LRF algorithm: schedule the job with largest ratio w/s first.
- Our contribution: the LRF algorithm is α - approximation,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$ for instance of jobs with equal density $w_j/s_j = 1$, where $i = \lceil \frac{2n}{m} \rceil$.
    - i) $1 + \frac{1}{i+1} \leq \alpha < 1 + \frac{1}{i}$;
    - ii) α is tight for some group of instance;
    - iii) we give tight upper bound of α for different group of instances.
  - $\alpha = 1 + \frac{m-1}{m+2}$, for general case.

Contribution

- Related works: first introduced by Zhang et al. [TCS 2013]
  - Strongly NP-hard when m is input.
  - Proposed a 2-approximation algorithm, *Largest-Ratio-First* (LRF) algorithm.
- LRF algorithm: schedule the job with largest ratio w/s first.
- Our contribution: the LRF algorithm is α - approximation,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$ for instance of jobs with equal density $w_j/s_j = 1$, where $i = \lceil \frac{2n}{m} \rceil$.
    - i) $1 + \frac{1}{i+1} \leq \alpha < 1 + \frac{1}{i}$.
    - ii) α is tight for some group of instance;
    - iii) we give tight upper bound of α for different group of instances.
  - $\alpha = 1 + \frac{m-1}{m+2}$, for general case.

Contribution

- Related works: first introduced by Zhang et al. [TCS 2013]
  - Strongly NP-hard when m is input.
  - Proposed a 2-approximation algorithm, *Largest-Ratio-First* (LRF) algorithm.
- LRF algorithm: schedule the job with largest ratio w/s first.
- Our contribution: the LRF algorithm is α - approximation,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$ for instance of jobs with equal density $w_j/s_j = 1$, where $i = \lceil \frac{2n}{m} \rceil$.
    - i) $1 + \frac{1}{i+1} \le \alpha < 1 + \frac{1}{i}$;
    - ii) α is tight for some group of instance;
    - iii) we give **tight** upper bound of α for different group of instances.
  - $\alpha = 1 + \frac{m-1}{m+2}$, for general case.

Contribution

- Related works: first introduced by Zhang et al. [TCS 2013]
  - Strongly NP-hard when m is input.
  - Proposed a 2-approximation algorithm, *Largest-Ratio-First* (LRF) algorithm.
- LRF algorithm: schedule the job with largest ratio w/s first.
- Our contribution: the LRF algorithm is α - approximation,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$ for instance of jobs with equal density $w_j/s_j = 1$, where $i = \lceil \frac{2n}{m} \rceil$.
    - i) $1 + \frac{1}{i+1} \leq \alpha < 1 + \frac{1}{i}$;
    - ii) α is tight for some group of instance;
    - iii) we give **tight** upper bound of α for different group of instances.
  - $\alpha = 1 + \frac{m-1}{m+2}$, for general case.

Contribution

- Related works: first introduced by Zhang et al. [TCS 2013]
  - Strongly NP-hard when m is input.
  - Proposed a 2-approximation algorithm, *Largest-Ratio-First* (LRF) algorithm.
- LRF algorithm: schedule the job with largest ratio w/s first.
- Our contribution: the LRF algorithm is α - approximation,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$ for instance of jobs with equal density $w_j/s_j = 1$, where $i = \lceil \frac{2n}{m} \rceil$.
    - i) $1 + \frac{1}{i+1} \leq \alpha < 1 + \frac{1}{i}$;
    - ii) α is tight for some group of instance;
    - iii) we give **tight** upper bound of α for different group of instances.
  - $\alpha = 1 + \frac{m-1}{m+2}$, for general case.

Contribution

- Related works: first introduced by Zhang et al. [TCS 2013]
  - Strongly NP-hard when m is input.
  - Proposed a 2-approximation algorithm, *Largest-Ratio-First* (LRF) algorithm.
- LRF algorithm: schedule the job with largest ratio w/s first.
- Our contribution: the LRF algorithm is α - approximation,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$ for instance of jobs with equal density $w_j/s_j = 1$, where $i = \lceil \frac{2n}{m} \rceil$.
    - i) $1 + \frac{1}{i+1} \leq \alpha < 1 + \frac{1}{i}$;
    - ii) α is tight for some group of instance;
    - iii) we give **tight** upper bound of α for different group of instances.
  - $\alpha = 1 + \frac{m-1}{m+2}$, for general case.

Contribution

- Related works: first introduced by Zhang et al. [TCS 2013]
  - Strongly NP-hard when m is input.
  - Proposed a 2-approximation algorithm, *Largest-Ratio-First* (LRF) algorithm.
- LRF algorithm: schedule the job with largest ratio w/s first.
- Our contribution: the LRF algorithm is α - approximation,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$ for instance of jobs with equal density $w_j/s_j = 1$, where $i = \lceil \frac{2n}{m} \rceil$.
    - i) $1 + \frac{1}{i+1} \leq \alpha < 1 + \frac{1}{i}$;
    - ii) α is tight for some group of instance;
    - iii) we give **tight** upper bound of α for different group of instances.
  - $\alpha = 1 + \frac{m-1}{m+2}$, for general case.

Contribution

- Related works: first introduced by Zhang et al. [TCS 2013]
  - Strongly NP-hard when m is input.
  - Proposed a 2-approximation algorithm, *Largest-Ratio-First* (LRF) algorithm.
- LRF algorithm: schedule the job with largest ratio w/s first.
- Our contribution: the LRF algorithm is α - approximation,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$ for instance of jobs with equal density $w_j/s_j = 1$, where $i = \lceil \frac{2n}{m} \rceil$.
    - i) $1 + \frac{1}{i+1} \leq \alpha < 1 + \frac{1}{i}$;
    - ii) α is tight for some group of instance;
    - iii) we give **tight** upper bound of α for different group of instances.
  - $\alpha = 1 + \frac{m-1}{m+2}$, for general case.

Instance of Jobs With Equal Density (overview)

- Equal density: $w_j = s_j, \ \forall j \in J$.
  - LRF algorithm = arbitrary order of jobs.
  - $\alpha(J) = \frac{\max_{S \in permutation(J)} T(S,J)}{\min_{S \in permutation(J)} T(S,J)}$
- Goal: for fixed value of n and m, find maximum α & corresponding instance J^*.

- $\alpha_{max} = \alpha(\mathfrak{J}^{(m,n)})$.
- We prove $J^* \in \mathfrak{J}_{org}^{(m,n)}$

Instance of Jobs With Equal Density (overview)

- Equal density: $w_j = s_j, \ \forall j \in J$.
  - LRF algorithm = arbitrary order of jobs.
  - $\alpha(J) = \frac{\max_{S \in permutation(J)} T(S,J)}{\min_{S \in permutation(J)} T(S,J)}$
- Goal: for fixed value of n and m, find maximum α & corresponding instance J^*.

- $\alpha_{max} = \alpha(\mathfrak{J}^{(m,n)})$.
- We prove $J^* \in \mathfrak{J}_{org}^{(m,n)}$

Instance of Jobs With Equal Density (overview)

- Equal density: $w_j = s_j, \ \forall j \in J$.
  - LRF algorithm = arbitrary order of jobs.
  - $\alpha(J) = \frac{\max_{S \in permutation(J)} T(S,J)}{\min_{S \in permutation(J)} T(S,J)}$
- Goal: for fixed value of n and m, find maximum α & corresponding instance J^*.
- $\alpha_{max} = \alpha(\mathfrak{J}^{(m,n)})$.
- We prove $J^* \in \mathfrak{J}_{org}^{(m,n)}$

Instance of Jobs With Equal Density (overview)

- Equal density: $w_j = s_j, \ \forall j \in J$.
  - LRF algorithm = arbitrary order of jobs.
  - $\alpha(J) = \frac{\max_{S \in permutation(J)} T(S,J)}{\min_{S \in permutation(J)} T(S,J)}$
- Goal: for fixed value of n and m, find maximum α & corresponding instance J^*.

- $\alpha_{max} = \alpha(\mathfrak{J}^{(m,n)})$.
- We prove $J^* \in \mathfrak{J}_{org}^{(m,n)}$

Instance of Jobs With Equal Density (overview)

- Equal density: $w_j = s_j, \ \forall j \in J$.
  - LRF algorithm = arbitrary order of jobs.
  - $\alpha(J) = \frac{\max_{S \in permutation(J)} T(S,J)}{\min_{S \in permutation(J)} T(S,J)}$
- Goal: for fixed value of n and m, find maximum α & corresponding instance J^*.

Definition

Define $\alpha(\mathfrak{J}) = \max_{J' \in \mathfrak{J}} \alpha(J')$ for any set of instances $\mathfrak{J}$.

Define $\mathfrak{J}^{(m,n)}$, $\mathfrak{J}_{one}^{(m,n)}$, $\mathfrak{J}_{org}^{(m,n)}$ s.t. $\mathfrak{J}_{org}^{(m,n)} \subseteq \mathfrak{J}_{one}^{(m,n)} \subseteq \mathfrak{J}^{(m,n)}$.

- $\alpha_{max} = \alpha(\mathfrak{J}^{(m,n)})$.
- We prove $J^* \in \mathfrak{J}_{org}^{(m,n)}$

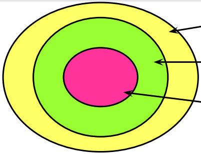
Instance of Jobs With Equal Density (overview)

- Equal density: $w_j = s_j, \ \forall j \in J$.
  - LRF algorithm = arbitrary order of jobs.
  - $\alpha(J) = \frac{\max_{S \in permutation(J)} T(S,J)}{\min_{S \in permutation(J)} T(S,J)}$
- Goal: for fixed value of n and m, find maximum α & corresponding instance J^*.

Definition

Define $\alpha(\mathfrak{J}) = \max_{J' \in \mathfrak{J}} \alpha(J')$ for any set of instances $\mathfrak{J}$.

Define $\mathfrak{J}^{(m,n)}$, $\mathfrak{J}_{one}^{(m,n)}$, $\mathfrak{J}_{org}^{(m,n)}$ s.t. $\mathfrak{J}_{org}^{(m,n)} \subseteq \mathfrak{J}_{one}^{(m,n)} \subseteq \mathfrak{J}^{(m,n)}$.



$\mathfrak{J}^{(m,n)} = \{J \mid \forall j \in J \ \ w_j = s_j, \ |J| = n\}$

$\mathfrak{J}_{one}^{(m,n)} = \{J \mid \forall j \in J \ \ 1 \leq s_j \leq m, \ J \in \mathfrak{J}^{(m,n)}\}$

$\mathfrak{J}_{org}^{(m,n)}$ contains all the *organized instances* of n jobs

- $\alpha_{max} = \alpha(\mathfrak{J}^{(m,n)})$.
- We prove $J^* \in \mathfrak{J}_{org}^{(m,n)}$

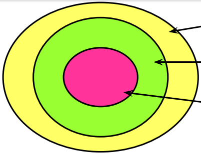
Instance of Jobs With Equal Density (overview)

- Equal density: $w_j = s_j, \ \forall j \in J$.
  - LRF algorithm = arbitrary order of jobs.
  - $\alpha(J) = \frac{\max_{S \in permutation(J)} T(S,J)}{\min_{S \in permutation(J)} T(S,J)}$
- Goal: for fixed value of n and m, find maximum α & corresponding instance J^*.

Definition

Define $\alpha(\mathfrak{J}) = \max_{J' \in \mathfrak{J}} \alpha(J')$ for any set of instances $\mathfrak{J}$.

Define $\mathfrak{J}^{(m,n)}$, $\mathfrak{J}_{one}^{(m,n)}$, $\mathfrak{J}_{org}^{(m,n)}$ s.t. $\mathfrak{J}_{org}^{(m,n)} \subseteq \mathfrak{J}_{one}^{(m,n)} \subseteq \mathfrak{J}^{(m,n)}$.



$\mathfrak{J}^{(m,n)} = \{J \mid \forall j \in J \ \ w_j = s_j, \ |J| = n\}$

$\mathfrak{J}_{one}^{(m,n)} = \{J \mid \forall j \in J \ \ 1 \leq s_j \leq m, \ \ J \in \mathfrak{J}^{(m,n)}\}$

$\mathfrak{J}_{org}^{(m,n)}$ contains all the *organized instances* of n jobs

- $\alpha_{max} = \alpha(\mathfrak{J}^{(m,n)})$.
- We prove $J^* \in \mathfrak{J}_{org}^{(m,n)}$

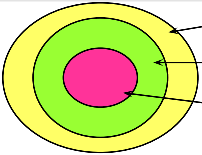
Instance of Jobs With Equal Density (overview)

- Equal density: $w_j = s_j, \ \forall j \in J$.
  - LRF algorithm $=$ arbitrary order of jobs.
  - $\alpha(J) = \frac{\max_{S \in permutation(J)} T(S,J)}{\min_{S \in permutation(J)} T(S,J)}$
- Goal: for fixed value of n and m, find maximum α & corresponding instance J^*.

Definition

Define $\alpha(\mathfrak{J}) = \max_{J' \in \mathfrak{J}} \alpha(J')$ for any set of instances $\mathfrak{J}$.

Define $\mathfrak{J}^{(m,n)}$, $\mathfrak{J}^{(m,n)}_{one}$, $\mathfrak{J}^{(m,n)}_{org}$ s.t. $\mathfrak{J}^{(m,n)}_{org} \subseteq \mathfrak{J}^{(m,n)}_{one} \subseteq \mathfrak{J}^{(m,n)}$.



$\mathfrak{J}^{(m,n)} = \{J \mid \forall j \in J \ \ w_j = s_j, \ |J| = n\}$

$\mathfrak{J}^{(m,n)}_{one} = \{J \mid \forall j \in J \ \ 1 \leq s_j \leq m, \ \ J \in \mathfrak{J}^{(m,n)}\}$

$\mathfrak{J}^{(m,n)}_{org}$ contains all the *organized instances* of n jobs

- $\alpha_{max} = \alpha(\mathfrak{J}^{(m,n)})$.
- We prove $J^* \in \mathfrak{J}^{(m,n)}_{org}$

Splitting of Jobs

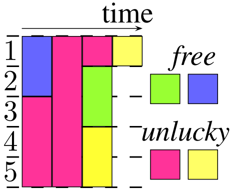
Definition

Free job: be executed within one unit time;
Unlucky job: one unit workload less, finished earlier.
Splitting: replace one job by two jobs, keep both w/s and total workload.

Splitting of Jobs

Definition

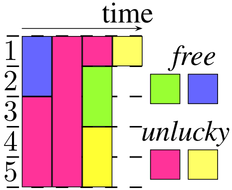
Free job: be executed within one unit time;
Unlucky job: one unit workload less, finished earlier.
Splitting: replace one job by two jobs, keep both w/s and total workload.

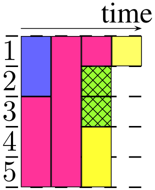


Splitting of Jobs

Definition

Free job: be executed within one unit time;
Unlucky job: one unit workload less, finished earlier.
Splitting: replace one job by two jobs, keep both w/s and total workload.

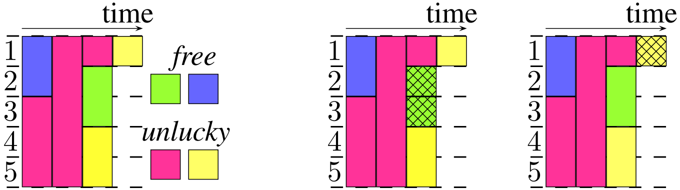




Splitting of Jobs

Definition

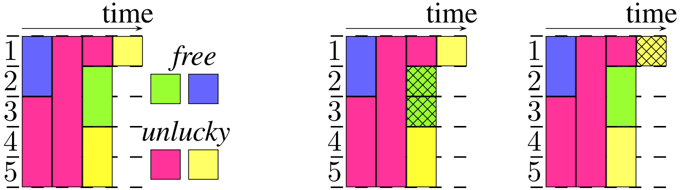
Free job: be executed within one unit time;
Unlucky job: one unit workload less, finished earlier.
Splitting: replace one job by two jobs, keep both w/s and total workload.



Splitting of Jobs

Definition

Free job: be executed within one unit time;
Unlucky job: one unit workload less, finished earlier.
Splitting: replace one job by two jobs, keep both w/s and total workload.

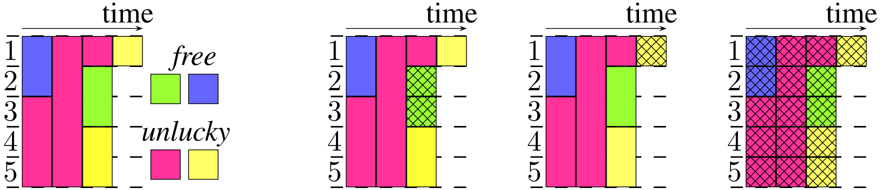


- $T' - T = 0$, for *free* jobs.
- $T' - T < 0$, for *unlucky* jobs.
- $T' - T \leq 0$, generally.

- green: 2×4 vs $1 \times 4 + 1 \times 4$.
- yellow: 3×4 vs $2 \times 3 + 1 \times 4$.

Definition

Free job: be executed within one unit time;
Unlucky job: one unit workload less, finished earlier.
Splitting: replace one job by two jobs, keep both w/s and total workload.



- $T' - T = 0$, for *free* jobs.
- $T' - T < 0$, for *unlucky* jobs.
- $T' - T \leq 0$, generally.

- green: 2×4 vs $1 \times 4 + 1 \times 4$.
- yellow: 3×4 vs $2 \times 3 + 1 \times 4$.

Lemma

For any feasible schedule S, $T(S, J) \geq T(J^{unit})$.

Properties of Organized Instance

Definition

An *organized instance* $J(y, z, k) = m^y + k^1 + 1^z$ s.t. $L(J) > m(y+1)$, where $n = y + z + 1$, $1 < k \leq m$, $0 < z < n$ $(y, z, k \in \mathbb{N})$.

Properties of Organized Instance

Definition

An *organized instance* $J(y, z, k) = m^y + k^1 + 1^z$ s.t. $L(J) > m(y + 1)$, where $n = y + z + 1$, $1 < k \leq m$, $0 < z < n$ $(y, z, k \in \mathbb{N})$.

Lemma

For any organized instance $J = J(y, z, k) \in \mathfrak{J}_{org}^{(m,n)}$,

- *schedule* $S_1 = (m^y, k, 1^z)$ *is optimal and* $T(OPT, J) = T(J^{unit})$.
- *schedule* $S_2 = (1, m^y, 1^{m-k}, k, 1^{z+k-1-m})$ *is an LRF schedule and* $T(LRF, J) = T(J^{unit}) + y(m-1) + (k-1)$.

Properties of Organized Instance

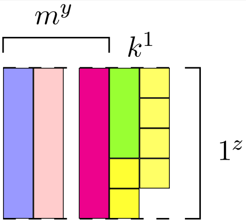
Definition

An *organized instance* $J(y, z, k) = m^y + k^1 + 1^z$ s.t. $L(J) > m(y+1)$, where $n = y + z + 1$, $1 < k \leq m$, $0 < z < n$ $(y, z, k \in \mathbb{N})$.

Lemma

For any organized instance $J = J(y, z, k) \in \mathfrak{J}_{org}^{(m,n)}$,

- *schedule* $S_1 = (m^y, k, 1^z)$ *is optimal and* $T(OPT, J) = T(J^{unit})$.
- *schedule* $S_2 = (1, m^y, 1^{m-k}, k, 1^{z+k-1-m})$ *is an LRF schedule and* $T(LRF, J) = T(J^{unit}) + y(m-1) + (k-1)$.



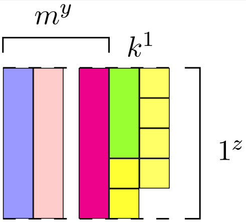
Definition

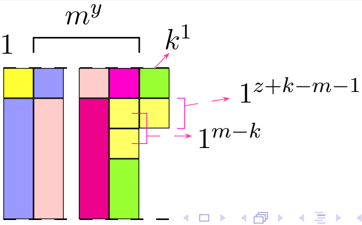
An *organized instance* $J(y,z,k) = m^y + k^1 + 1^z$ s.t. $L(J) > m(y+1)$, where $n = y+z+1$, $1 < k \leq m$, $0 < z < n$ $(y,z,k \in \mathbb{N})$.

Lemma

For any organized instance $J = J(y,z,k) \in \mathfrak{J}_{org}^{(m,n)}$,

- *schedule* $S_1 = (m^y, k, 1^z)$ *is optimal and* $T(OPT, J) = T(J^{unit})$.
- *schedule* $S_2 = (1, m^y, 1^{m-k}, k, 1^{z+k-1-m})$ *is an LRF schedule and* $T(LRF, J) = T(J^{unit}) + y(m-1) + (k-1)$.





Properties of Organized Instance

Definition

An *organized instance* $J(y,z,k) = m^y + k^1 + 1^z$ s.t. $L(J) > m(y+1)$, where $n = y+z+1$, $1 < k \leq m$, $0 < z < n$ $(y,z,k \in \mathbb{N})$.

Lemma

For any organized instance $J = J(y,z,k) \in \mathfrak{J}_{org}^{(m,n)}$,

- *schedule* $S_1 = (m^y, k, 1^z)$ *is optimal and* $T(OPT,J) = T(J^{unit})$.
- *schedule* $S_2 = (1, m^y, 1^{m-k}, k, 1^{z+k-1-m})$ *is an LRF schedule and* $T(LRF,J) = T(J^{unit}) + y(m-1) + (k-1)$.

Lemma

Given $m, n \geq 2$, $\alpha(\mathfrak{J}_{org}^{(m,n)}) > \alpha(\mathfrak{J}_{org}^{(m,n+1)})$.

Properties of Organized Instance

Definition

An *organized instance* $J(y,z,k) = m^y + k^1 + 1^z$ s.t. $L(J) > m(y+1)$, where $n = y+z+1$, $1 < k \leq m$, $0 < z < n$ $(y,z,k \in \mathbb{N})$.

Lemma

For any organized instance $J = J(y,z,k) \in \mathfrak{J}_{org}^{(m,n)}$,

- *schedule* $S_1 = (m^y, k, 1^z)$ *is optimal and* $T(OPT, J) = T(J^{unit})$.
- *schedule* $S_2 = (1, m^y, 1^{m-k}, k, 1^{z+k-1-m})$ *is an LRF schedule and* $T(LRF, J) = T(J^{unit}) + y(m-1) + (k-1)$.

Lemma

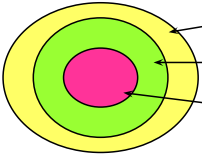
Given $m, n \geq 2$, $\alpha(\mathfrak{J}_{org}^{(m,n)}) > \alpha(\mathfrak{J}_{org}^{(m,n+1)})$.

$\forall J = J(y,z,k) \in \mathfrak{J}_{org}^{(m,n)}$ s.t. $n = y+z+1$, $\alpha(J) = 1 + \frac{y(m-1)+(k-1)}{T(J^{unit})}$

Instance Achieving Maximum Approximation Ratio

Next, we prove that

$$\alpha(\mathfrak{J}_{org}^{(m,n)}) = \alpha(\mathfrak{J}_{one}^{(m,n)}) = \alpha(\mathfrak{J}^{(m,n)})$$



$$\mathfrak{J}^{(m,n)} = \{J \mid \forall j \in J \;\; w_j = s_j,\ |J| = n\}$$

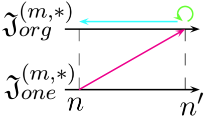
$$\mathfrak{J}_{one}^{(m,n)} = \{J \mid \forall j \in J \;\; 1 \le s_j \le m,\ J \in \mathfrak{J}^{(m,n)}\}$$

$\mathfrak{J}_{org}^{(m,n)}$ contains all the *organized instances* of n jobs

Instance Achieving Maximum Approximation Ratio

Lemma

Given $m, n \geq 2$, $\alpha(\mathfrak{J}_{one}^{(m,n)}) = \alpha(\mathfrak{J}_{org}^{(m,n)})$.



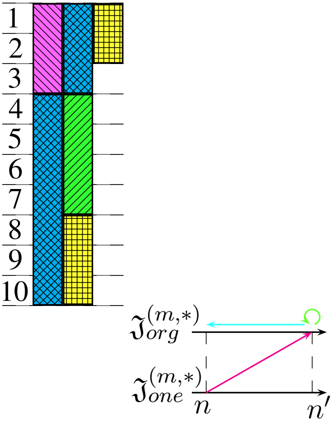
$\alpha(J) \leq \alpha(J')$

$J' \in \mathfrak{J}_{org}^{(m,n')}$

$\alpha(\mathfrak{J}_{org}^{(m,n')}) \leq \alpha(\mathfrak{J}_{org}^{(m,n)})$

Lemma

Given $m, n \geq 2$, $\alpha(\mathfrak{J}_{one}^{(m,n)}) = \alpha(\mathfrak{J}_{org}^{(m,n)})$.



$\alpha(J) \leq \alpha(J')$

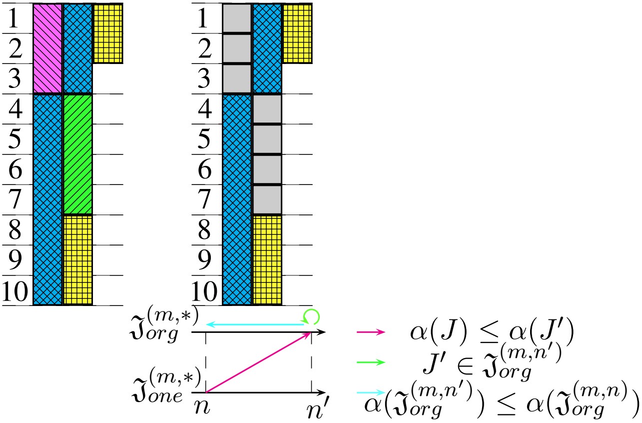
$J' \in \mathfrak{J}_{org}^{(m,n')}$

$\alpha(\mathfrak{J}_{org}^{(m,n')}) \leq \alpha(\mathfrak{J}_{org}^{(m,n)})$

Instance Achieving Maximum Approximation Ratio

Lemma

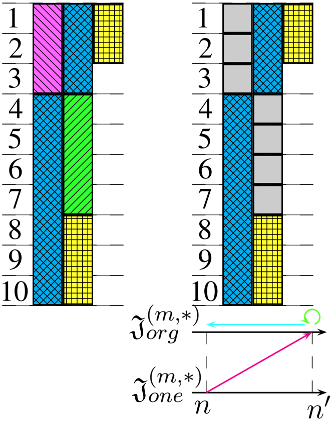
Given $m, n \geq 2$, $\alpha(\mathfrak{J}_{one}^{(m,n)}) = \alpha(\mathfrak{J}_{org}^{(m,n)})$.



Instance Achieving Maximum Approximation Ratio

Lemma

Given $m, n \geq 2$, $\alpha(\mathfrak{J}_{one}^{(m,n)}) = \alpha(\mathfrak{J}_{org}^{(m,n)})$.



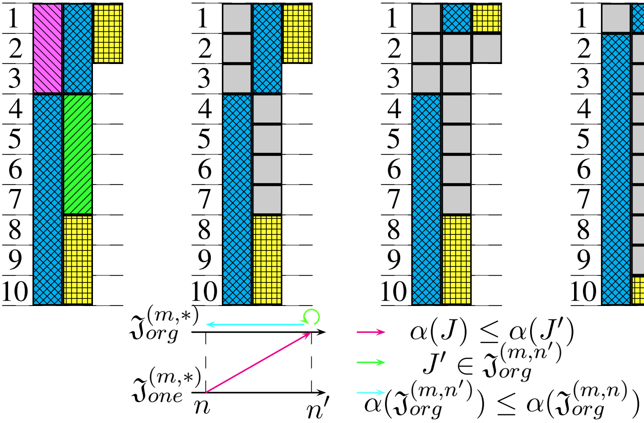
$\alpha(J) \leq \alpha(J')$

$J' \in \mathfrak{J}_{org}^{(m,n')}$

$\alpha(\mathfrak{J}_{org}^{(m,n')}) \leq \alpha(\mathfrak{J}_{org}^{(m,n)})$

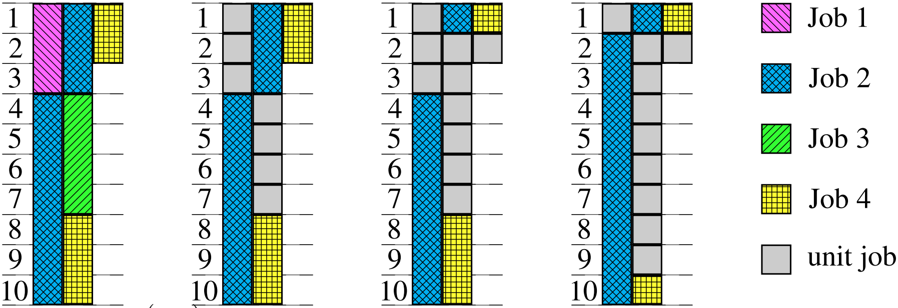
Lemma

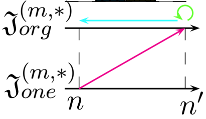
Given $m, n \geq 2$, $\alpha(\mathfrak{J}_{one}^{(m,n)}) = \alpha(\mathfrak{J}_{org}^{(m,n)})$.



Lemma

Given $m, n \geq 2$, $\alpha(\mathfrak{J}_{one}^{(m,n)}) = \alpha(\mathfrak{J}_{org}^{(m,n)})$.





$\alpha(J) \leq \alpha(J')$

$J' \in \mathfrak{J}_{org}^{(m,n')}$

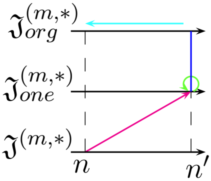
$\alpha(\mathfrak{J}_{org}^{(m,n')}) \leq \alpha(\mathfrak{J}_{org}^{(m,n)})$

Instance Achieving Maximum Approximation Ratio

Lemma

Given $n, m \geq 2$, $\alpha(\mathfrak{J}^{(m,n)}) = \alpha(\mathfrak{J}^{(m,n)}_{org})$.

- split s.t. $s_{j_2} = m$, objective value reduces by $s_{j_1} \cdot 1$.
- $T(OPT, J') \leq T(OPT, J) - s_{j_1}$.
- $T(LRF, J') \geq T(LRF, J) - s_{j_1}$.
- Consequently, $\alpha(J') = \frac{T(LRF,J')}{T(OPT,J')} > \frac{T(LRF,J)}{T(OPT,J)} = \alpha(J)$.



$\alpha(J) < \alpha(J')$

$J' \in \mathfrak{J}^{(m,n')}_{one}$

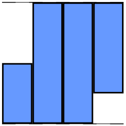
$\alpha(\mathfrak{J}^{(m,n')}_{one}) = \alpha(\mathfrak{J}^{(m,n')}_{org})$

$\alpha(\mathfrak{J}^{(m,n')}_{org}) \leq \alpha(\mathfrak{J}^{(m,n)}_{org})$

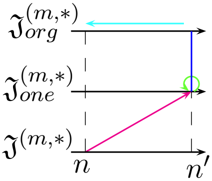
Instance Achieving Maximum Approximation Ratio

Lemma

Given $n, m \geq 2$, $\alpha(\mathfrak{J}^{(m,n)}) = \alpha(\mathfrak{J}_{org}^{(m,n)})$.



- split s.t. $s_{j_2} = m$, objective value reduces by $s_{j_1} \cdot 1$.
- $T(OPT, J') \leq T(OPT, J) - s_{j_1}$.
- $T(LRF, J') \geq T(LRF, J) - s_{j_1}$.
- Consequently, $\alpha(J') = \frac{T(LRF,J')}{T(OPT,J')} > \frac{T(LRF,J)}{T(OPT,J)} = \alpha(J)$.



$\alpha(J) < \alpha(J')$

$J' \in \mathfrak{J}_{one}^{(m,n')}$

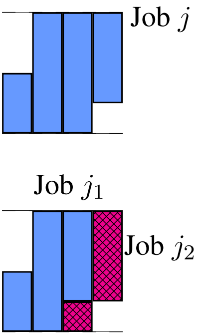
$\alpha(\mathfrak{J}_{one}^{(m,n')}) = \alpha(\mathfrak{J}_{org}^{(m,n')})$

$\alpha(\mathfrak{J}_{org}^{(m,n')}) \leq \alpha(\mathfrak{J}_{org}^{(m,n)})$

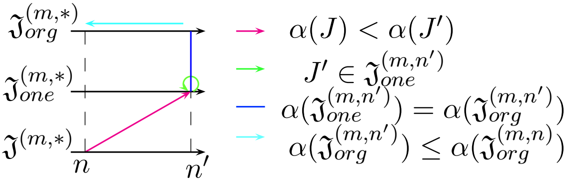
Instance Achieving Maximum Approximation Ratio

Lemma

Given $n, m \geq 2$, $\alpha(\mathfrak{J}^{(m,n)}) = \alpha(\mathfrak{J}_{org}^{(m,n)})$.



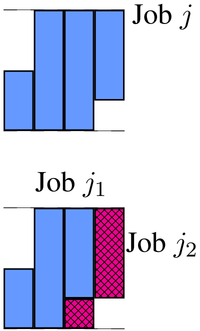
- split s.t. $s_{j_2} = m$, objective value reduces by $s_{j_1} \cdot 1$.
- $T(OPT, J') \leq T(OPT, J) - s_{j_1}$.
- $T(LRF, J') \geq T(LRF, J) - s_{j_1}$.
- Consequently, $\alpha(J') = \frac{T(LRF,J')}{T(OPT,J')} > \frac{T(LRF,J)}{T(OPT,J)} = \alpha(J)$.



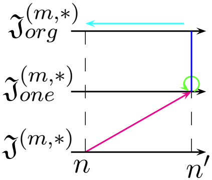
Instance Achieving Maximum Approximation Ratio

Lemma

Given $n, m \geq 2$, $\alpha(\mathfrak{J}^{(m,n)}) = \alpha(\mathfrak{J}^{(m,n)}_{org})$.



- split s.t. $s_{j_2} = m$, objective value reduces by $s_{j_1} \cdot 1$.
- $T(OPT, J') \leq T(OPT, J) - s_{j_1}$.
- $T(LRF, J') \geq T(LRF, J) - s_{j_1}$.
- Consequently, $\alpha(J') = \frac{T(LRF,J')}{T(OPT,J')} > \frac{T(LRF,J)}{T(OPT,J)} = \alpha(J)$.



$\alpha(J) < \alpha(J')$

$J' \in \mathfrak{J}^{(m,n')}_{one}$

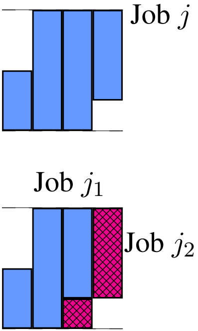
$\alpha(\mathfrak{J}^{(m,n')}_{one}) = \alpha(\mathfrak{J}^{(m,n')}_{org})$

$\alpha(\mathfrak{J}^{(m,n')}_{org}) \leq \alpha(\mathfrak{J}^{(m,n)}_{org})$

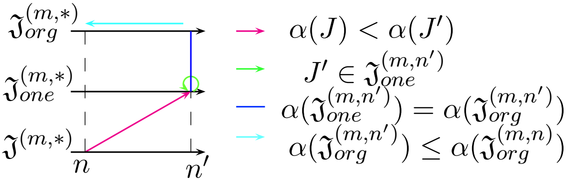
Instance Achieving Maximum Approximation Ratio

Lemma

Given $n, m \geq 2$, $\alpha(\mathfrak{J}^{(m,n)}) = \alpha(\mathfrak{J}^{(m,n)}_{org})$.



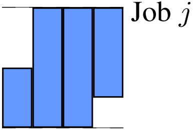
- split s.t. $s_{j_2} = m$, objective value reduces by $s_{j_1} \cdot 1$.
- $T(OPT, J') \leq T(OPT, J) - s_{j_1}$.
- $T(LRF, J') \geq T(LRF, J) - s_{j_1}$.
- Consequently, $\alpha(J') = \frac{T(LRF,J')}{T(OPT,J')} > \frac{T(LRF,J)}{T(OPT,J)} = \alpha(J)$.

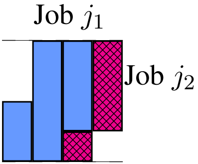


Instance Achieving Maximum Approximation Ratio

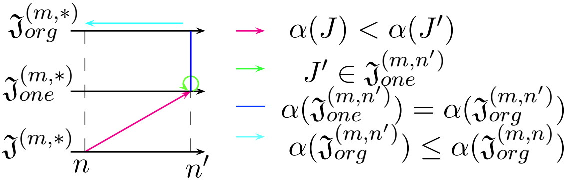
Lemma

Given $n, m \geq 2$, $\alpha(\mathfrak{J}^{(m,n)}) = \alpha(\mathfrak{J}_{org}^{(m,n)})$.





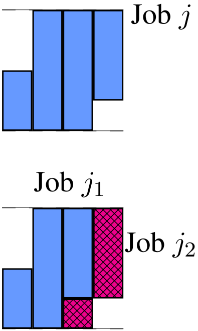
- split s.t. $s_{j_2} = m$, objective value reduces by $s_{j_1} \cdot 1$.
- $T(OPT, J') \leq T(OPT, J) - s_{j_1}$.
- $T(LRF, J') \geq T(LRF, J) - s_{j_1}$.
- Consequently, $\alpha(J') = \frac{T(LRF,J')}{T(OPT,J')} > \frac{T(LRF,J)}{T(OPT,J)} = \alpha(J)$.



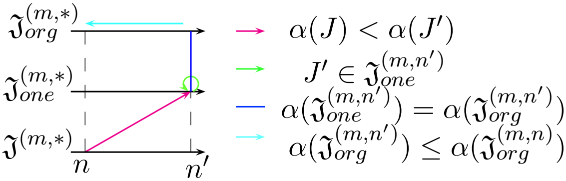
Instance Achieving Maximum Approximation Ratio

Lemma

Given $n, m \geq 2$, $\alpha(\mathfrak{J}^{(m,n)}) = \alpha(\mathfrak{J}^{(m,n)}_{org})$.



- split s.t. $s_{j_2} = m$, objective value reduces by $s_{j_1} \cdot 1$.
- $T(OPT, J') \leq T(OPT, J) - s_{j_1}$.
- $T(LRF, J') \geq T(LRF, J) - s_{j_1}$.
- Consequently, $\alpha(J') = \frac{T(LRF,J')}{T(OPT,J')} > \frac{T(LRF,J)}{T(OPT,J)} = \alpha(J)$.



Previously, $\forall J = J(y,z,k) \in \mathfrak{J}_{org}^{(m,n)}$ s.t. $n = y + z + 1$, $\alpha(J) = 1 + \frac{y(m-1)+(k-1)}{T(J^{unit})}$.

- $L(J) = y \cdot m + 1 \cdot k + z \cdot 1$ is the total workload of J.
- Define $a = \lfloor \frac{L(J)}{m} \rfloor$, $b = L(J) - am$.
- $T(J^{unit}) = m(1 + 2 + ... + a) + b(a+1)$.
- $\alpha(J) = \frac{T(LRF,J)}{T(OPT,J)} = 1 + \frac{am+b-n}{ma(a+1)/2+b(a+1)}$

Previously, $\forall J = J(y,z,k) \in \mathfrak{J}_{org}^{(m,n)}$ s.t. $n = y + z + 1$,
$\alpha(J) = 1 + \frac{y(m-1)+(k-1)}{T(J^{unit})}$.

- $L(J) = y \cdot m + 1 \cdot k + z \cdot 1$ is the total workload of J.
- Define $a = \lfloor \frac{L(J)}{m} \rfloor$, $b = L(J) - am$.
- $T(J^{unit}) = m(1 + 2 + ... + a) + b(a+1)$.
- $\alpha(J) = \frac{T(LRF,J)}{T(OPT,J)} = 1 + \frac{am+b-n}{ma(a+1)/2+b(a+1)}$

Previously, $\forall J = J(y,z,k) \in \mathfrak{J}_{org}^{(m,n)}$ s.t. $n = y + z + 1$,
$\alpha(J) = 1 + \frac{y(m-1)+(k-1)}{T(J^{unit})}$.

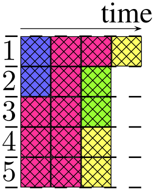
- $L(J) = y \cdot m + 1 \cdot k + z \cdot 1$ is the total workload of J.
- Define $a = \lfloor \frac{L(J)}{m} \rfloor,\ b = L(J) - am$.
- $T(J^{unit}) = m(1+2+...+a) + b(a+1)$.
- $\alpha(J) = \frac{T(LRF,J)}{T(OPT,J)} = 1 + \frac{am+b-n}{ma(a+1)/2+b(a+1)}$

Previously, $\forall J = J(y,z,k) \in \mathfrak{J}_{org}^{(m,n)}$ s.t. $n = y + z + 1$,
$\alpha(J) = 1 + \frac{y(m-1)+(k-1)}{T(J^{unit})}$.

- $L(J) = y \cdot m + 1 \cdot k + z \cdot 1$ is the total workload of J.
- Define $a = \lfloor \frac{L(J)}{m} \rfloor,\ b = L(J) - am$.
- $T(J^{unit}) = m(1 + 2 + ... + a) + b(a+1)$.
- $\alpha(J) = \frac{T(LRF,J)}{T(OPT,J)} = 1 + \frac{am+b-n}{ma(a+1)/2+b(a+1)}$

Previously, $\forall J = J(y,z,k) \in \mathfrak{J}_{org}^{(m,n)}$ s.t. $n = y+z+1$, $\alpha(J) = 1 + \frac{y(m-1)+(k-1)}{T(J^{unit})}$.

- $L(J) = y \cdot m + 1 \cdot k + z \cdot 1$ is the total workload of J.
- Define $a = \lfloor \frac{L(J)}{m} \rfloor$, $b = L(J) - am$.
- $T(J^{unit}) = m(1+2+...+a) + b(a+1)$.
- $\alpha(J) = \frac{T(LRF,J)}{T(OPT,J)} = 1 + \frac{am+b-n}{ma(a+1)/2+b(a+1)}$



Approximation Ratio Bound (Organized Instance)

- Define function $g(a,b) = \frac{am+b-n}{ma(a+1)/2+b(a+1)}$ $(a > 0,\ b \geq 0)$.

- Define $i = \lceil \frac{2n}{m} \rceil$,

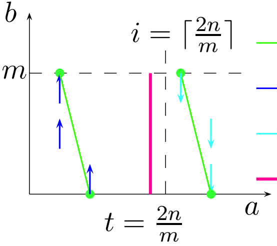
$$g(a,b) \leq g(i,0),\ \forall a > 0, 0 \leq b < m, a, b \in \mathbb{N}$$

- Eventually,

$$\alpha(J) \leq 1 + g(i,0) = 1 + \frac{2(im-n)}{i(i+1)m}$$

Approximation Ratio Bound (Organized Instance)

- Define function $g(a,b) = \frac{am+b-n}{ma(a+1)/2+b(a+1)}$ $(a > 0,\ b \geq 0)$.



$g(a,m) = g(a+1,0),\ \forall a > 0$

$g(a,b) < g(a,b+1),\ \forall 0 < a < t,\ 0 \leq b < m$

$g(a,b) > g(a,b+1),\ \forall a > t,\ 0 \leq b < m$

$g(a,b) = g(a+1,0),\ a = t,\ 0 \leq b \leq m$

- Define $i = \lceil \frac{2n}{m} \rceil$,

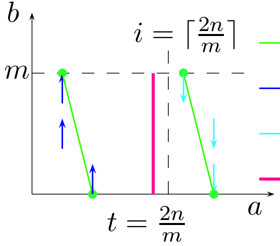
$$g(a,b) \leq g(i,0),\ \forall a > 0, 0 \leq b < m, a, b \in \mathbb{N}$$

- Eventually,

$$\alpha(J) \leq 1 + g(i,0) = 1 + \frac{2(im-n)}{i(i+1)m}$$

Approximation Ratio Bound (Organized Instance)

- Define function $g(a,b) = \frac{am+b-n}{ma(a+1)/2+b(a+1)}$ $(a > 0,\ b \geq 0)$.



— $g(a,m) = g(a+1,0),\ \forall a > 0$

→ $g(a,b) < g(a,b+1),\ \forall 0 < a < t,\ 0 \leq b < m$

→ $g(a,b) > g(a,b+1),\ \forall a > t,\ 0 \leq b < m$

— $g(a,b) = g(a+1,0),\ a = t,\ 0 \leq b \leq m$

- Define $i = \lceil \frac{2n}{m} \rceil$,

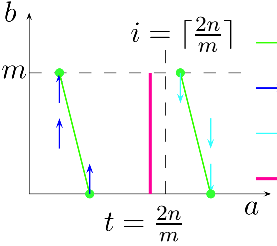
$$g(a,b) \leq g(i,0),\ \forall a > 0, 0 \leq b < m, a, b \in \mathbb{N}$$

- Eventually,

$$\alpha(J) \leq 1 + g(i,0) = 1 + \frac{2(im-n)}{i(i+1)m}$$

Approximation Ratio Bound (Organized Instance)

- Define function $g(a,b) = \frac{am+b-n}{ma(a+1)/2+b(a+1)}$ $(a > 0,\ b \geq 0)$.



— $g(a,m) = g(a+1,0),\ \forall a > 0$

→ $g(a,b) < g(a,b+1),\ \forall 0 < a < t,\ 0 \leq b < m$

→ $g(a,b) > g(a,b+1),\ \forall a > t,\ 0 \leq b < m$

— $g(a,b) = g(a+1,0),\ a = t,\ 0 \leq b \leq m$

- Define $i = \lceil \frac{2n}{m} \rceil$,

$$g(a,b) \leq g(i,0),\ \forall a > 0, 0 \leq b < m, a, b \in \mathbb{N}$$

- Eventually,

$$\alpha(J) \leq 1 + g(i,0) = 1 + \frac{2(im-n)}{i(i+1)m}$$

Lemma

For any organized instance $J \in \mathfrak{J}_{org}^{(m,n)}$, $L(J) = im$ *if and only if* $i \leq n + 1 - m$, *where* $i = \lceil \frac{2n}{m} \rceil$.

- how about $i > n + 1 - m$, $L(J) = im - 1,\ im + 1, ..?$
- Group the instance, in each group (region) of instance we find the tight bound.
- Roughly, for region B_i, $B_i = \{(m, n) \mid i = \lceil \frac{2n}{m} \rceil, m \geq 3\}$

Approximation Ratio Bound (Organized Instance)

<table>
<tr><th>Regions</th><th>Approximation Ratio</th><th>function $g(a,b)$</th></tr>
<tr><td>$B_0 = \{m = 2\}$</td><td>$1 + \frac{n-1}{n^2}$ (tight)</td><td>$g(n-1, m-1)$</td></tr>
<tr><td>$B_1 = \{m \geq 2n, m \geq 3\}$</td><td>$1 + \frac{m-n+1}{m+2}$ (tight)</td><td>$g(i, 1)$</td></tr>
<tr><td>$B_2^* = \{m = 2n-1, m \geq 3\}$</td><td>$1 + \frac{m-1}{2m-1}$ (tight)</td><td>$g(i-1, n-1)$</td></tr>
<tr><td>$B_2 = \{n \leq m \leq 2n-2, m \geq 3\}$</td><td>$1 + \frac{2m-n+1}{3m+3}$ (tight)</td><td>$g(i, 1)$</td></tr>
<tr><td>$B_3^* = \{m = n-1, m \geq 3, n \neq 4\}$</td><td>$1 + \frac{2m-2}{6m-3}$ (tight)</td><td>$g(i-1, m-1)$</td></tr>
<tr><td>$B_3 = \{\frac{2n}{3} \leq m \leq n-2, m \geq 3\}$</td><td rowspan="3">$1 + \frac{2(im-n)}{i(i+1)m}$ (tight)</td><td rowspan="3">$g(i, 0)$</td></tr>
<tr><td>$B_4 = \{3 < \frac{2n}{m} \leq 4, m \geq 3, n \neq 5\}$</td></tr>
<tr><td>$B_i = \{i-1 < \frac{2n}{m} \leq i, m \geq 3\}, \forall i \geq 5$</td></tr>
<tr><td>$B^* = \{m = 3, 4 \leq n \leq 5\}$</td><td>$1 + \frac{2(im-n)}{i(i+1)m}$</td><td></td></tr>
</table>

Table: Approximation ratio bound of instances in different regions.

Approximation Ratio Bound (Organized Instance)

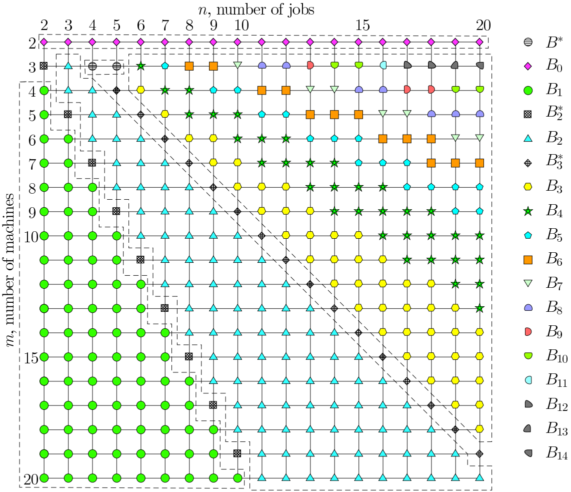


Figure: Example of the region division for $2 \le n, m \le 20$

Instance of Jobs with Arbitrary Weights

Definition

$\forall J$, let $J^{(e)} = \{(w'_j, s_j) \; | w'_j = s_j, j \in J\}$ be the corresponding job set of J.

Instance of Jobs with Arbitrary Weights

Definition

$\forall J$, let $J^{(e)} = \{(w'_j, s_j) \mid w'_j = s_j, j \in J\}$ be the corresponding job set of J.

Lemma

$\forall J$, there always exists a subset $J_s \subseteq J$ such that $\alpha(J_s^{(e)}) \geq \alpha(J)$.

Instance of Jobs with Arbitrary Weights

Definition

$\forall J$, let $J^{(e)} = \{(w'_j, s_j) \mid w'_j = s_j, j \in J\}$ be the corresponding job set of J.

Lemma

$\forall J$, there always exists a subset $J_s \subseteq J$ such that $\alpha(J_s^{(e)}) \geq \alpha(J)$.

Theorem

$\forall J,\ \alpha(J) \leq \alpha(\mathfrak{J}^{(m,2)}) = 1 + \frac{m-1}{m+2}$.

Conclusion

- We prove that LRF algorithm is α - approximation ($\alpha < 2$).
- For instance of jobs with equal density $w_j/s_j = 1$,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$, where $i = \lceil \frac{2n}{m} \rceil$
  - we give tight upper bound for different group of instances.
- For general case, $\alpha = 1 + \frac{m-1}{m+2}$.

Conclusion

- We prove that LRF algorithm is α - approximation ($\alpha < 2$).
- For instance of jobs with equal density $w_j/s_j = 1$,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$, where $i = \lceil \frac{2n}{m} \rceil$
  - we give tight upper bound for different group of instances.
- For general case, $\alpha = 1 + \frac{m-1}{m+2}$.

Conclusion

- We prove that LRF algorithm is α - approximation ($\alpha < 2$).
- For instance of jobs with equal density $w_j/s_j = 1$,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$, where $i = \lceil \frac{2n}{m} \rceil$
  - we give tight upper bound for different group of instances.
- For general case, $\alpha = 1 + \frac{m-1}{m+2}$.

Conclusion

- We prove that LRF algorithm is α - approximation ($\alpha < 2$).
- For instance of jobs with equal density $w_j/s_j = 1$,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$, where $i = \lceil \frac{2n}{m} \rceil$
  - we give tight upper bound for different group of instances.
- For general case, $\alpha = 1 + \frac{m-1}{m+2}$.

Conclusion

- We prove that LRF algorithm is α - approximation ($\alpha < 2$).
- For instance of jobs with equal density $w_j/s_j = 1$,
  - $\alpha = 1 + \frac{i+(i-2n/m)}{i(i+1)}$, where $i = \lceil \frac{2n}{m} \rceil$
  - we give tight upper bound for different group of instances.
- For general case, $\alpha = 1 + \frac{m-1}{m+2}$.

Future Work

- what is the complexity of the problem when m is fixed?
- Considering release time?
- Considering machine reservation time period?
- unrelated machines, i.e. different machine takes different processing time.

Future Work

- what is the complexity of the problem when m is fixed?
- Considering release time?
- Considering machine reservation time period?
- unrelated machines, i.e. different machine takes different processing time.

Future Work

- what is the complexity of the problem when m is fixed?
- Considering release time?
- Considering machine reservation time period?
- unrelated machines, i.e. different machine takes different processing time.

Future Work

- what is the complexity of the problem when m is fixed?
- Considering release time?
- Considering machine reservation time period?
- unrelated machines, i.e. different machine takes different processing time.

Question

Question?