Trade-offs in Resource Allocation Problems

Abhinav Srivastav

Thesis Advisors:
Dr. Oded Maler   *Prof. Denis Trystram*

February 16, 2017

UNIVERSITÉ Grenoble Alpes

PERSYVAL-Lab

Agenda

Multi-Objective Optimization

- Formulating trade-offs
- Solution Methods
- Background
- Our algorithm
- Experimental results
- Conclusion

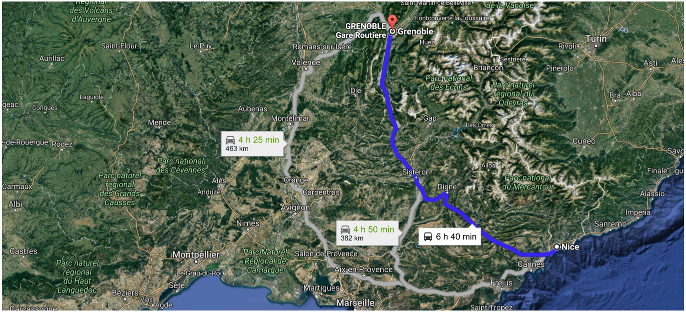
Trade-offs in Scheduling

- Theoretical guarantees
- Scheduling
- Resource augmentation
- Previous work
- Our approach
- Results
- Conclusion

Motivation

- Many real-life optimization problems involve multi-criteria
- Solutions are evaluated with respect to several, possible conflicting, objectives
- A solution is better in a criterion and has worse performance in other criterion
- Results in set of *incomparable* solutions
- Such problems arise in engineering, operation research, telecommunication, finance, medicine, etc.

Example 1: Tour Planning



- Multiple criteria: distance, tolls, traffic, scenic value, etc.
- Best route?

Example 2: Scheduling

- Modern day processors can vary their processing speed
  - High speed leads to shorter execution time of a job
  - More speed means more energy consumption

- Trade-off between energy consumption and execution time

Multi-Objective Optimization

Formalizing Trade-offs

- Problems with trade-offs can be seen as multi-objective optimization problems

Formalizing Trade-offs

- Problems with trade-offs can be seen as multi-objective optimization problems
- Addressed by providing a set of incomparable solutions
  - Example: route 1 $= \{382$ kms, 4 tolls$\}$
    route 2 $= \{463$ kms, 1 toll$\}$

Mathematical Formalization

- $\mathcal{S}$ represents the solution space
  - Example: route 1, route 2 and route 3

Mathematical Formalization

- $\mathcal{S}$ represents the solution space
  - Example: route 1, route 2 and route 3
- $\mathcal{C}$ represents the cost space
  - Example: distance, tolls, scenic values

Mathematical Formalization

- $\mathcal{S}$ represents the solution space
  - Example: route 1, route 2 and route 3
- $\mathcal{C}$ represents the cost space
  - Example: distance, tolls, scenic values
- $\mathcal{F} : \mathcal{S} \rightarrow \mathcal{C}$ represents a set of d-objective functions, *i.e.* $\mathcal{F} = \{f_1, ..., f_d\}$
  - Example: $f_1(\text{route 1}) = 382$ kms
    $f_2(\text{route 1}) = 4$ tolls

Mathematical Formalization

- $\mathcal{S}$ represents the solution space
  - Example: route 1, route 2 and route 3
- $\mathcal{C}$ represents the cost space
  - Example: distance, tolls, scenic values
- $\mathcal{F} : \mathcal{S} \rightarrow \mathcal{C}$ represents a set of d-objective functions, *i.e.* $\mathcal{F} = \{f_1, ..., f_d\}$
  - Example: $f_1(\text{route 1}) = 382$ kms
  $f_2(\text{route 1}) = 4$ tolls
- A multi-objective problem can be seen as a tuple $\varphi = \{\mathcal{S}, \mathcal{C}, \mathcal{F}\}$

Partial Order

- s strongly dominates s' iff $\forall i \in \{1, ...d\} : f_i(s) \leq f_i(s')$ and for some j, $f_j(s) < f_j(s')$
  - Example: $\mathcal{F}(\text{route x}) = \{382 \text{ kms}, 1 \text{ toll}\}$
    $\mathcal{F}(\text{route y}) = \{263 \text{ kms}, 0 \text{ tolls}\}$

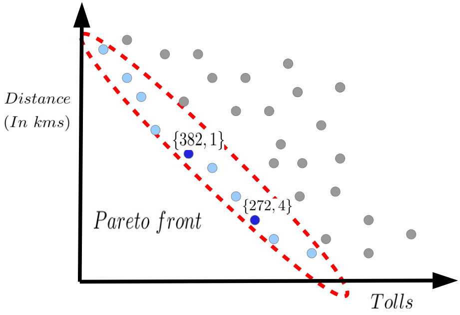
Partial Order

- s strongly dominates s' iff $\forall i \in \{1, ...d\} : f_i(s) \leq f_i(s')$ and for some j, $f_j(s) < f_j(s')$
  - Example: $\mathcal{F}(\text{route x}) = \{382 \text{ kms}, 1 \text{ toll}\}$
    $\mathcal{F}(\text{route y}) = \{263 \text{ kms}, 0 \text{ tolls}\}$

- s is incomparable with s' iff $\exists\, i, j \in \{1, ..., d\} : f_i(s) < f_i(s')$ and $f_j(s) > f_j(s')$
  - Example: $\mathcal{F}(\text{route x}) = \{382 \text{ kms}, 1 \text{ toll}\}$
    $\mathcal{F}(\text{route y}) = \{272 \text{ kms}, 4 \text{ tolls}\}$

Pareto Front

- s is a Pareto optimal solution iff $\forall s' \in \mathcal{S}$, s' does not strongly dominate s'
- Pareto front: A set with all Pareto optimal solutions



Problem and Solution

- A trade-off problem can be formulated as a multi-objective problem $\varphi = \{\mathcal{S}, \mathcal{C}, \mathcal{F}\}$
- The objective is to find the Pareto front

Finding Pareto front

- Difficulties
  - Many discrete problems are NP-complete, even in the single objective case
  - There can be a large number of solutions in the Pareto front
- Solution
  - We need an approximation of the Pareto front

Finding Pareto front

- Difficulties
  - Many discrete problems are NP-complete, even in the single objective case
  - There can be a large number of solutions in the Pareto front
- Solution
  - We need an approximation of the Pareto front

Definition

$A \subseteq \mathcal{S}$ is an approximation iff s and s' are incomparable $\forall s, s' \in A$.

Finding Pareto front

- Difficulties
  - Many discrete problems are NP-complete, even in the single objective case
  - There can be a large number of solutions in the Pareto front
- Solution
  - We need an approximation of the Pareto front

Definition

$A \subseteq \mathcal{S}$ is an approximation iff s and s' are incomparable $\forall s, s' \in A$.

- Optimality is no more guaranteed
- There may be a solution $s \in \mathcal{S}$ that strongly dominates $s' \in A$

Generating Pareto front

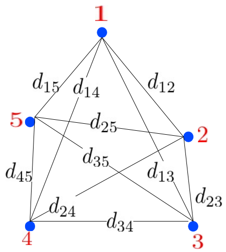
- Numerous optimizers in the literature
- Our focus is on the local search algorithms
- They are very effective in solving hard single-objective problems
  - Example: Best solutions for *travelling salesman problem (TSP)*
- Extensions to multi-objective scenario

Local Search

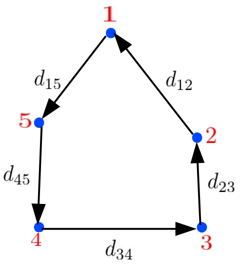
Consider a single objective version of TSP:

- Given n cities
- $\forall i, j \in 1, .., n : d_{ij}$
- Find the tour with smallest total distance

Example:



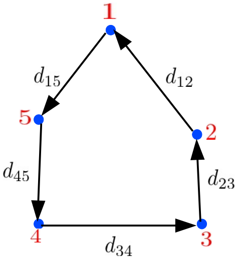
5 cities with all pair wise distances



One possible solution

Representing a Solution

Each solution $s \in \mathcal{S}$ is defined by the values assigned to a set of discrete variables
Example:

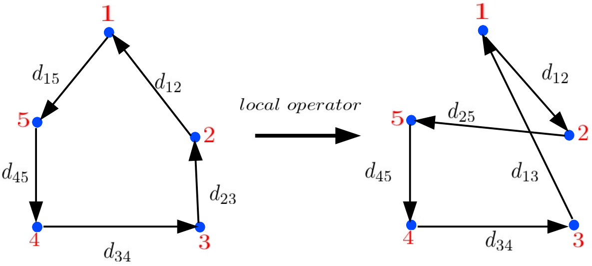


One possible solution

$$\pi = \{3, 2, 1, 5, 4\}$$

Local Operator $\mathcal{L} : \mathcal{S} \to \mathcal{S}$

Transforms a solution to another solution by making *local changes* in the representation
Example:



$\pi = \{3, 2, 1, 5, 4\}$ $\quad$ $\pi' = \{3, 1, 2, 5, 4\}$

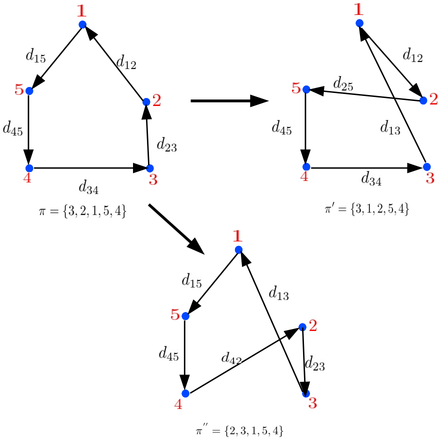
Neighborhood $N(s)$

$Dist(s, s')$: smallest number of changes required to transform s into s'

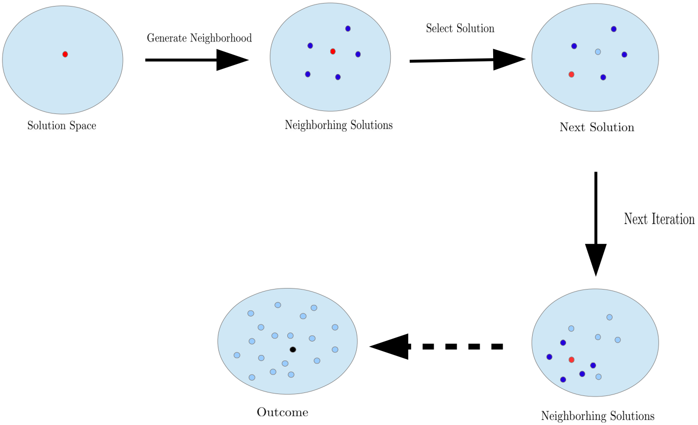
Neighborhood $N(s)$

$Dist(s, s')$: smallest number of changes required to transform s into s'

- There exist multiple solutions at any fixed distance
- A set of all such solutions is called the neighborhood of the solution



Local Search



Extensions to Multi-objective Scenario

- Problems
  - The cost space is multi-dimensional, $\mathcal{C} \subset \mathbb{R}^d$
  - $N(s)$ may contain multiple incomparable solutions
  - Outcome is also a set of incomparable solutions

Extensions to Multi-objective Scenario

- Problems
  - The cost space is multi-dimensional, $\mathcal{C} \subset \mathbb{R}^d$
  - $N(s)$ may contain multiple incomparable solutions
  - Outcome is also a set of incomparable solutions
- Solutions
  - Scalarize multiple objectives into a single objective
  - Another approach is to use the notion of dominance in the local search
  - Such algorithms are known as Pareto local search (PLS)

Pareto Local Search

- Data structure
  - PLS maintains a set P of non-dominated solutions
  - Each solution $s \in P$ is flagged either as visited or unvisited

Pareto Local Search

- Data structure
  - PLS maintains a set P of non-dominated solutions
  - Each solution $s \in P$ is flagged either as visited or unvisited
- Basic steps in each iteration
  - Select a unvisited solution $s \in P$
  - Generate neighbors $N(s)$ of s
  - Merge $N(s)$ with P using dominance criteria

Pareto Local Search

- Pros:
  - No scalarization needed
  - Outcomes are mutually incomparable
  - PLS provides fast convergence to Pareto local optimum
  - It can handle problems with large number of optimal solutions

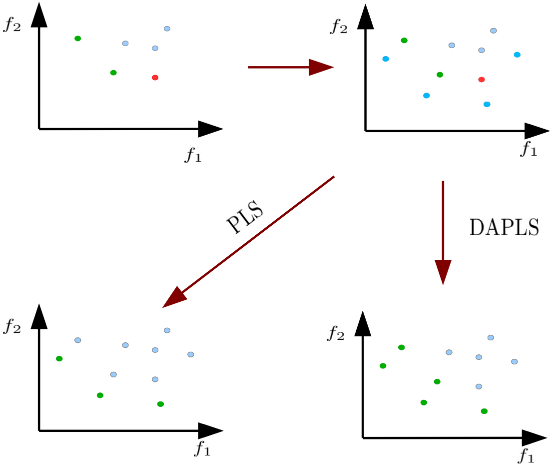
Pareto Local Search

- Pros:
  - No scalarization needed
  - Outcomes are mutually incomparable
  - PLS provides fast convergence to Pareto local optimum
  - It can handle problems with large number of optimal solutions
- Cons:
  - PLS searches only a subset of the solutions space
  - The unvisited solutions from P are remove if dominated by a new solution
  - This restricts convergence to Pareto front
  - This can also have a negative effect on the spread of solution (diversity)

Our Contribution

- We propose a new algorithm, DAPLS
- DAPLS does not prematurely remove candidate solutions
- We show that it provides better convergence to the Pareto front
- It maintains same diversity (spread of solutions) comparison to PLS

Intuition behind DAPLS



Double Archive Pareto Local Search

- Data structures
  - DAPLS maintains a set P of non-dominated solution
  - An additional set L to maintain the candidate solutions
  - The set P is presented as the final outcome

Double Archive Pareto Local Search

- Data structures
  - DAPLS maintains a set P of non-dominated solution
  - An additional set L to maintain the candidate solutions
  - The set P is presented as the final outcome
- Basic steps in each iteration
  - Select a solution $s \in L$ without replacement
  - Generate neighbors $N(s)$ of s
  - Merge $N(s)$ with P using dominance criteria
  - Merge $(N(s) \cap P)$ to L without using dominance criteria

Double Archive Pareto Local Search

- Data structures
  - DAPLS maintains a set P of non-dominated solution
  - An additional set L to maintain the candidate solutions
  - The set P is presented as the final outcome
- Basic steps in each iteration
  - Select a solution $s \in L$ without replacement
  - Generate neighbors $N(s)$ of s
  - Merge $N(s)$ with P using dominance criteria
  - Merge $(N(s) \cap P)$ to L without using dominance criteria
- $(N(s) \cap P)$ consists of new solutions added to P
- L contains solutions that may be dominated

Benchmark

- Multi-objective quadratic assignment problem

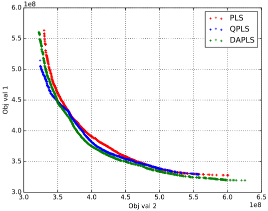
  Given: - Given n facilities and n locations
  - Distance between each pair of locations d_{ij}
  - Multi-dimensional flow between each pair of facilities f_{ab}^k

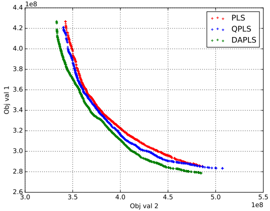
  Find: A mapping π from facilities to locations that minimizes $C^k(\pi), \forall k \in \{1, \cdots, d\}$

$$C^k(\pi) = \sum_{a=1}^{n} \sum_{b=1}^{n} F_{ab}^{i} . d_{\pi(a),\pi(b)}$$

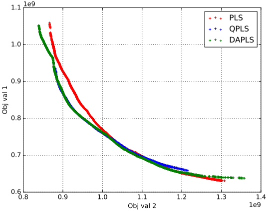
- Several instances of bi-objective and tri-objective QAP
- Instances are generated with MQAP tool, Knowles et al. 2003

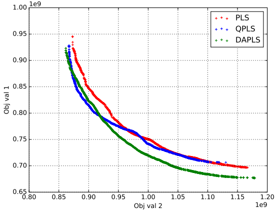
Experimental Results





Median attainment surfaces for $n = 50$ with $\rho = 0.25$ (on left) and $\rho = 0.75$ (on right)





Median attainment surfaces for $n = 75$ with $\rho = 0.25$ (on left) and $\rho = 0.75$ (on right)

Conclusion

- We treat trade-offs as a multi-objective problem
- DAPLS for solving multi-objective combinatorial problems
- Our method improves upon the previous works
  - Provides better convergence to the optimal Pareto front
  - Provides same spread of solutions as PLS and QPLS

Conclusion

- We treat trade-offs as a multi-objective problem
- DAPLS for solving multi-objective combinatorial problems
- Our method improves upon the previous works
  - Provides better convergence to the optimal Pareto front
  - Provides same spread of solutions as PLS and QPLS

- How does DAPLS perform on other kind of problems?
- How to deal with problems in higher dimension?
- Performance of DAPLS in tabu search, simulated annealing, other models?

Trade-offs in Scheduling

Theoretical Guarantees

- Heuristics are known to perform well on real-world problems
- Generally, they provide no guarantee on the quality of solutions
- Guarantees for understanding the complexity of the problem
- Instances on which particular heuristic will perform well

Approximation Ratio

- We assume the offline setting
- The entire instance beforehand
- We focus on minimization problems, e.g. Scheduling

Definition

An algorithm is ρ-approximation iff

$$\rho \geq \max_{I} \left\{ \frac{\text{Cost of the algorithm on input instance } \mathcal{I}}{\text{Optimal cost on input instance } \mathcal{I}} \right\}$$

Competitive Ratio

- We assume the online setting
- The instance is revealed as the time progresses
- Again our focus is on minimization problems

Definition

An algorithm is ρ-competitive iff

$$\rho \geq \max_{I} \left\{ \frac{\text{Cost of the algorithm on input instance } \mathcal{I}}{\text{Optimal offline cost on input instance } \mathcal{I}} \right\}$$

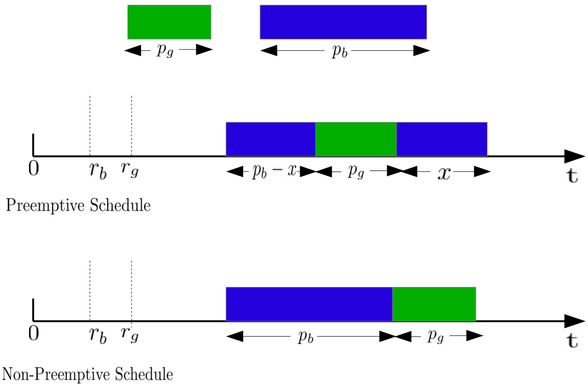
Scheduling Model

- The problem of allocation resources to a set of requests
- We consider the client-server model where
  - Resources are modelled as machines
  - Requests are modelled as jobs
- Such systems include
  - Operating systems,
  - High performance platforms,
  - Web-servers, etc.

Preliminaries

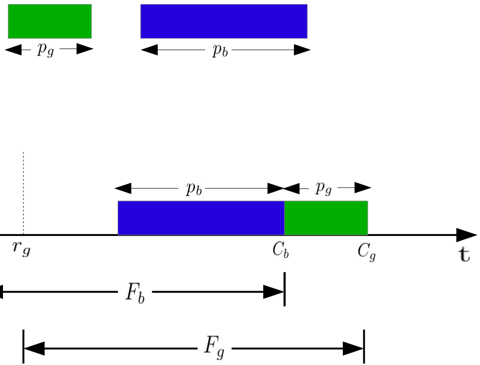
- A scheduling problem consists of
  - a set of jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$
  - a set of machines $\mathcal{M} = \{1, \ldots, m\}$
- Each job $j \in \mathcal{J}$ is characterised by
  - a processing requirement p_j
  - a release time r_j
  - a weight w_j
- Machine environment
  - Single machine $m = 1$
  - Parallel machines $m > 1$
  - Unrelated machines $m > 1$
- In parallel machines, each job has machine-independent processing time
- In unrelated machines, each job has machine-dependent processing times

Types of Schedules



Problem Definition

- We focus on non-preemptive scheduling
- Our aim is to reduce the time a job spends in a system, Flow time



Objective Functions

- The first measure is based on the average performance of the system
  Average weighted flow-time: $\sum_j w_j F_j$
- Average flow-time measure is known to have extreme outliers
- The second measure is based on minimizing these extreme outliers
  Maximum weighted flow: $\max_j w_j F_j$

Special Case

- We consider the problem of fair scheduling
- Jobs should wait proportionally to their processing requirement
- The most relevant metric is stretch
- Stretch $S_j = \frac{F_j}{p_j}$
- Specialized case of $w_j F_j$ with $w_j = 1/p_j$

Max-stretch Problem

- Each job $j \in \mathcal{J}$ is characterised by
  - a processing requirement p_j
  - a release time r_j
  - a weight $w_j = \frac{1}{p_j}$
- Machine environment : single machine $m = 1$
- Objective function: $\min \max_{j \in \mathcal{J}} w_j F_j = F_j / p_j$
- Our model considers the online problem where the jobs' processing time are known at their release time

Previous Results

- Bender et al. 1998: Non-preemptive problem cannot be approximated within factor of $\Omega(n^{1-\epsilon})$
- Interesting results can be derived in instance-dependent parameter $\Delta = \frac{p_{max}}{p_{min}}$
- Bender et al. 1998: Any online algorithm has at least $\Omega(\Delta^{1/3})$ for preemptive problem
- Saule et al. 2012: An improved lower bound of $\left(\frac{1+\Delta}{2}\right)$ was shown for the non-preemptive problem
- Legrand et al. 2008: FCFS is known to be Δ-competitive

Our Contributions

Theorem

There is no $\rho\Delta$-competitive non-preemptive algorithm for minimizing max-stretch on a single machine for any fixed $\rho < \left(\frac{\sqrt{5}-1}{2}\right) \approx 0.618$.

Our Contributions

Theorem

There is no $\rho\Delta$-competitive non-preemptive algorithm for minimizing max-stretch on a single machine for any fixed $\rho < \left(\frac{\sqrt{5}-1}{2}\right) \approx 0.618$.

Theorem

There exists an algorithm that achieves $(1 + \alpha\Delta)$-competitive for the problem of minimizing max-stretch non-preemptively, where $\alpha = \left(\frac{\sqrt{5}-1}{2}\right)$.

Our idea is based on the waiting-time strategy

Accurate Models

- Previous result was instance dependent (Δ)
- The aim is to provide theoretical guarantees independent of the instance
- Flow time problems have strong lower bound
- However, many heuristics perform well in practice

Trade-offs in Scheduling

- The widely accepted norm is to use resource augmentation

Trade-offs in Scheduling

- The widely accepted norm is to use resource augmentation
- Kalyanasundaram et al. 2000 proposed the idea of speed augmentation
  - The online algorithm is equipped more speed in comparison to the optimal algorithm
- Phillips et al. 2002 proposed the idea of machine augmentation
  - The online algorithm is equipped more number of machines in comparison to the optimal algorithm
- Choudhury et al. 2015 proposed the idea of rejection model
  - The online algorithm has slightly smaller instance in comparison to the optimal algorithm

Re-defining Competitive Ratio

Speed augmentation: A job with processing requirement p will take $\frac{p}{s}$ time units in the algorithm while the optimal takes p time units

$$\rho \geq \max_{I} \left\{ \frac{\text{Cost of the algorithm with speed } s \text{ on input instance } \mathcal{I}}{\text{Optimal cost with speed 1 on input instance } \mathcal{I}} \right\}$$

Re-defining Competitive Ratio

Speed augmentation: A job with processing requirement p will take $\frac{p}{s}$ time units in the algorithm while the optimal takes p time units

$$\rho \geq \max_{I} \left\{ \frac{\text{Cost of the algorithm with speed } s \text{ on input instance } \mathcal{I}}{\text{Optimal cost with speed 1 on input instance } \mathcal{I}} \right\}$$

Rejection Model: the algorithm's performance is computed on a slightly smaller instance than the optimal algorithm

$$\rho \geq \max_{I} \left\{ \frac{\text{Cost of the algorithm on input instance } \mathcal{I}'}{\text{Optimal cost on input instance } \mathcal{I}} \right\}$$

where $I' \subseteq I$

Problem Definition

- We have a set $\mathcal{M}$ of m unrelated machines
- Each job j has
  - a machine-dependent processing time, i.e. $p_{ij}, \forall i \in \mathcal{M}$
  - a release time r_j
  - a weights $w_j = 1$
- Our goal is to design a non-preemptive schedule that min $\sum_j F_j$
- Jobs arrive online
- (p_{ij}, w_j) are known at r_j

Related Works

- Offline settings:
  - Kellerer et al. 1999: A strong lower bound of $O(\sqrt{n})$ exists in the classical model
  - Bansal et al. 2007: There exists a 12-speed 2-approximation algorithm
  - Im et al. 2015: A quasi-polynomial $(1+\epsilon)$-speed $(1+\epsilon)$-approximation algorithm

Related Works

- Offline settings:
  - Kellerer et al. 1999: A strong lower bound of $O(\sqrt{n})$ exists in the classical model
  - Bansal et al. 2007: There exists a 12-speed 2-approximation algorithm
  - Im et al. 2015: A quasi-polynomial $(1+\epsilon)$-speed $(1+\epsilon)$-approximation algorithm
- Online settings:
  - Chekuri et al. 2001: Lower bound of $\Omega(n)$ for unweighted flow on a single machine
  - Bunde et al. 2004: SPT is $\Delta/2$-competitive for total flow on a single machine
  - Tao et al. 2013: WSPT is $O(\Delta)$-competitive for parallel machines

Our Approach

- We formulate our problem as a linear program
- We use the concept of duality in optimization
- Weak duality: the cost of dual problem is at most the cost of the primal problem

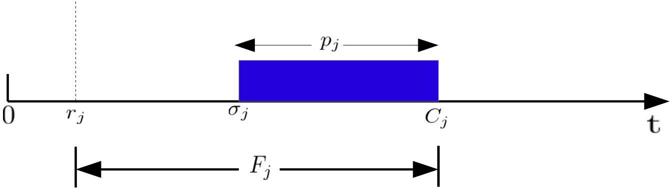
- Competitive ratio can be defined as:

$$\frac{\text{Objective value of Primal LP}}{\text{Objective value of Dual LP}}$$

Decision Variables

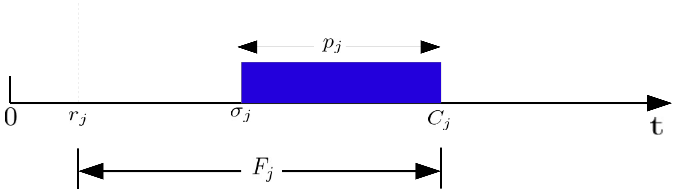
- Each job $j \in \mathcal{J}$ has a set of variables $x_j(t), \forall t$
- Constraint on $x_j(t) \in \{0, 1\}$
  - $x_j(t) = 1$ iff job is running at t
  - $x_j(t) = 0$, otherwise
- Job j can run only after its release time r_j
  - $x_j(t) = 0, \forall t < r_j$
- Job as processing requirement of at most p_j
  - $\int\limits_0^\infty x_j(t)dt = p_j \implies \int\limits_{r_j}^\infty x_j(t)dt = p_j$
- At each time, at most one job can run
  - $\sum\limits_j x_j(t) \leq 1, \forall t$

Objective Function



$$F_j = C_j - r_j = \sigma_j - r_j + p_j \geq \int_{\sigma_j}^{C_j} \left(\frac{t - r_j}{p_j} + 1 \right) x_j(t) dt$$

Objective Function



$$F_j = C_j - r_j = \sigma_j - r_j + p_j \geq \int_{\sigma_j}^{C_j} \left(\frac{t - r_j}{p_j} + 1 \right) x_j(t)dt$$

Objective function: $\min \sum_j \int_{r_j}^{\infty} \left(\frac{t-r_j}{p_j} + 1 \right) x_j(t)dt$

Linear Programming Relaxation

Single Machine

$$\min \sum_{j \in \mathcal{J}} \int_{r_j}^{\infty} \left(\frac{t - r_j + p_j}{p_j} \right) x_j(t) dt$$

$$\int_{r_j}^{\infty} \frac{x_j(t)}{p_j} dt \geq 1 \qquad \forall j \in \mathcal{J}$$

$$\sum_{j \in \mathcal{J}} x_j(t) \leq 1 \qquad \forall t \geq 0$$

$$x_j(t) \geq 0 \qquad \forall j \in \mathcal{J}, t \geq 0$$

Linear Programming Relaxation

Single Machine

$$\min \sum_{j \in \mathcal{J}} \int_{r_j}^{\infty} \left(\frac{t - r_j + p_j}{p_j} \right) x_j(t) dt$$

$$\begin{aligned} \int_{r_j}^{\infty} \frac{x_j(t)}{p_j} dt &\geq 1 && \forall j \in \mathcal{J} \\ \sum_{j \in \mathcal{J}} x_j(t) &\leq 1 && \forall t \geq 0 \\ x_j(t) &\geq 0 && \forall j \in \mathcal{J}, t \geq 0 \end{aligned}$$

Unrelated Machines

$$\min \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}} \int_{r_j}^{\infty} \frac{t - r_j + p_{ij}}{p_{ij}} x_{ij}(t) dt$$

$$\begin{aligned} \sum_{i \in \mathcal{M}} \int_{r_j}^{\infty} \frac{x_{ij}(t)}{p_{ij}} dt &\geq 1 && \forall j \in \mathcal{J} \\ \sum_{j \in \mathcal{J}} x_{ij}(t) &\leq 1 && \forall i \in \mathcal{M}, t \geq 0 \\ x_{ij}(t) &\geq 0 && \forall i \in \mathcal{M}, j \in \mathcal{J}, t \geq 0 \end{aligned}$$

Primal-Dual

Primal LP

$$\min \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\int_{r_j}^{\infty}\frac{t-r_j+p_{ij}}{p_{ij}}x_{ij}(t)dt$$

$$\sum_{i\in\mathcal{M}}\int_{r_j}^{\infty}\frac{x_{ij}(t)}{p_{ij}}dt \geq 1 \qquad \forall j\in\mathcal{J}$$

$$\sum_{j\in\mathcal{J}}x_{ij}(t) \leq 1 \qquad \forall i\in\mathcal{M},\ t\geq 0$$

$$x_{ij}(t)\geq 0 \qquad \forall i\in\mathcal{M},\ j\in\mathcal{J},\ t\geq 0$$

Primal-Dual

Primal LP

$$\min \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}} \int_{r_j}^{\infty} \frac{t - r_j + p_{ij}}{p_{ij}} x_{ij}(t)dt$$

$$\sum_{i\in\mathcal{M}} \int_{r_j}^{\infty} \frac{x_{ij}(t)}{p_{ij}} dt \geq 1 \qquad \forall j \in \mathcal{J}$$

$$\sum_{j\in\mathcal{J}} x_{ij}(t) \leq 1 \qquad \forall i \in \mathcal{M},\ t \geq 0$$

$$x_{ij}(t) \geq 0 \qquad \forall i \in \mathcal{M},\ j \in \mathcal{J},\ t \geq 0$$

Dual LP

$$\max \sum_{j\in\mathcal{J}} \lambda_j - \sum_{i\in\mathcal{M}} \int_0^{\infty} \gamma_i(t)dt$$

$$\frac{\lambda_j}{p_{ij}} - \gamma_i(t) \leq \frac{t - r_j + p_{ij}}{p_{ij}} \qquad \forall i \in \mathcal{M},\ j \in \mathcal{J},\ t \geq r_j$$

$$\lambda_j, \gamma_i(t) \geq 0 \qquad \forall i \in \mathcal{M},\ j \in \mathcal{J},\ t \geq 0$$

Competitive Ratio

Competitive Ratio ρ can be defined as:

$$\rho = \frac{\min \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}} \int_{r_j}^{\infty} \frac{t - r_j + p_{ij}}{p_{ij}} x_{ij}(t) dt}{\max \sum_{j \in \mathcal{J}} \lambda_j - \sum_{i \in \mathcal{M}} \int_0^{\infty} \gamma_i(t) dt}$$

Dual LP plays the role of the optimal algorithm

Speed Augmentation

Primal LP (Online algorithm)

$$\min \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{J}} \int_{r_j}^{\infty} \frac{t - r_j + p_{ij}}{p_{ij}} x_{ij}(t) dt$$

$$\sum_{i \in \mathcal{M}} \int_{r_j}^{\infty} \frac{x_{ij}(t)}{p_{ij}} dt \geq 1 \qquad \forall j \in \mathcal{J}$$

$$\sum_{j \in \mathcal{J}} x_{ij}(t) \leq (1 + \epsilon_s) \qquad \forall i \in \mathcal{M},\ t \geq 0$$

$$x_{ij}(t) \geq 0 \qquad \forall i \in \mathcal{M},\ j \in \mathcal{J},\ t \geq 0$$

Dual LP (Optimal algorithm)

$$\max \sum_{j \in \mathcal{J}} \lambda_j - \sum_{i \in \mathcal{M}} \int_{0}^{\infty} \gamma_i(t) dt$$

$$\frac{\lambda_j}{p_{ij}} - \gamma_i(t) \leq \frac{t - r_j + p_{ij}}{p_{ij}} \qquad \forall i \in \mathcal{M},\ j \in \mathcal{J},\ t \geq r_j$$

$$\lambda_j, \gamma_i(t) \geq 0 \qquad \forall i \in \mathcal{M},\ j \in \mathcal{J},\ t \geq 0$$

Rejection Model

Primal LP (Online algorithm)

$$\min \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}\setminus\mathcal{R}} \int_{r_j}^{\infty} \frac{t-r_j+p_{ij}}{p_{ij}} x_{ij}(t)dt$$

$$\sum_{i\in\mathcal{M}} \int_{r_j}^{\infty} \frac{x_{ij}(t)}{p_{ij}} dt \geq 1 \qquad \forall j \in \mathcal{J}\setminus\mathcal{R}$$

$$\sum_{j\in\mathcal{J}\setminus\mathcal{R}} x_{ij}(t) \leq 1 \qquad \forall i \in \mathcal{M},\ t\geq 0$$

$$x_{ij}(t) \geq 0 \qquad \forall i \in \mathcal{M},\ j\in\mathcal{J}\setminus\mathcal{R},\ t\geq 0$$

Dual LP (Optimal algorithm)

$$\max \sum_{j\in\mathcal{J}} \lambda_j - \sum_{i\in\mathcal{M}} \int_{0}^{\infty} \gamma_i(t)dt$$

$$\frac{\lambda_j}{p_{ij}} - \gamma_i(t) \leq \frac{t-r_j+p_{ij}}{p_{ij}} \qquad \forall i \in \mathcal{M},\ j\in\mathcal{J},\ t\geq r_j$$

$$\lambda_j, \gamma_i(t) \geq 0 \qquad \forall i \in \mathcal{M},\ j\in\mathcal{J},\ t\geq 0$$

Speed Augmentation + Rejection Model

Primal LP (Online algorithm)

$$\min \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}\setminus\mathcal{R}} \int_{r_j}^{\infty} \frac{t-r_j+p_{ij}}{p_{ij}} x_{ij}(t)dt$$

$$\sum_{i\in\mathcal{M}} \int_{r_j}^{\infty} \frac{x_{ij}(t)}{p_{ij}} dt \geq 1 \qquad \forall j \in \mathcal{J}\setminus\mathcal{R}$$

$$\sum_{j\in\mathcal{J}\setminus\mathcal{R}} x_{ij}(t) \leq (1+\epsilon_s) \qquad \forall i \in \mathcal{M},\ t \geq 0$$

$$x_{ij}(t) \geq 0 \qquad \forall i \in \mathcal{M},\ j \in \mathcal{J}\setminus\mathcal{R},\ t \geq 0$$

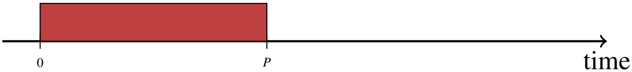
Dual LP (Optimal algorithm)

$$\max \sum_{j\in\mathcal{J}} \lambda_j - \sum_{i\in\mathcal{M}} \int_0^{\infty} \gamma_i(t)dt$$

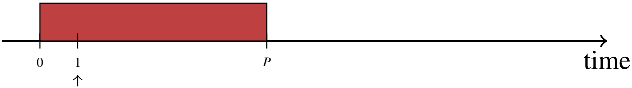
$$\frac{\lambda_j}{p_{ij}} - \gamma_i(t) \leq \frac{t-r_j+p_{ij}}{p_{ij}} \qquad \forall i \in \mathcal{M},\ j \in \mathcal{J},\ t \geq r_j$$

$$\lambda_j, \gamma_i(t) \geq 0 \qquad \forall i \in \mathcal{M},\ j \in \mathcal{J},\ t \geq 0$$

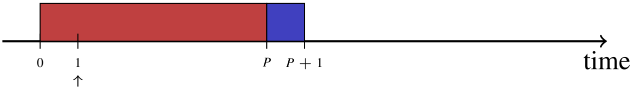
Intuition behind Rejection



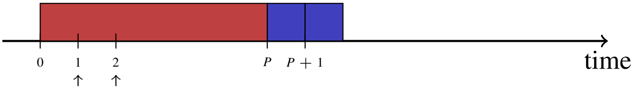
Intuition behind Rejection



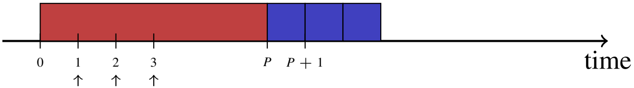
Intuition behind Rejection



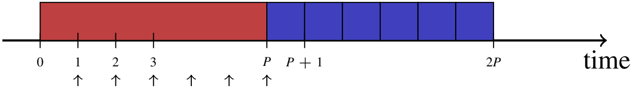
Intuition behind Rejection



Intuition behind Rejection

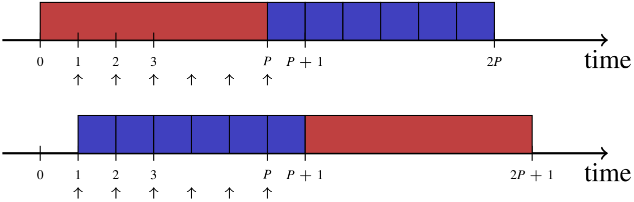


Intuition behind Rejection



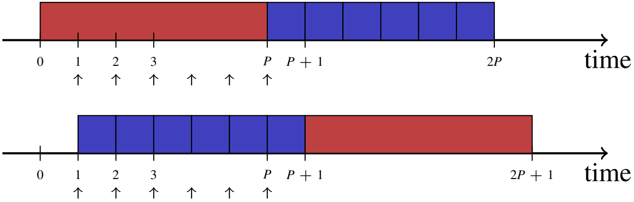
- P small jobs
- each small job has flow time P

Intuition behind Rejection



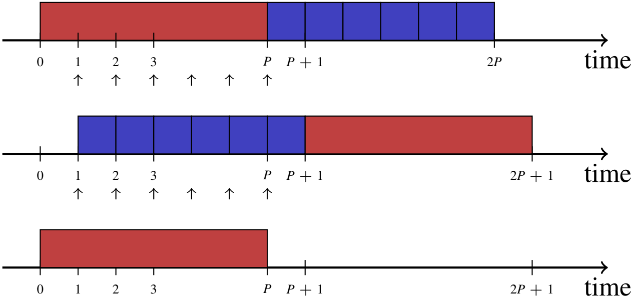
- P small jobs
- each small job has flow time P
- while in the optimal it has flow time 1

Intuition behind Rejection



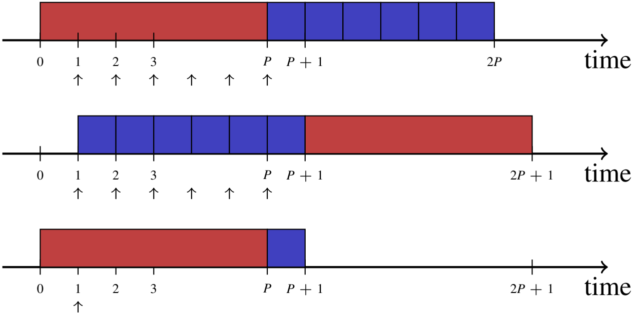
- P small jobs
- each small job has flow time P
- while in the optimal it has flow time 1
- but we can reject

Intuition behind Rejection



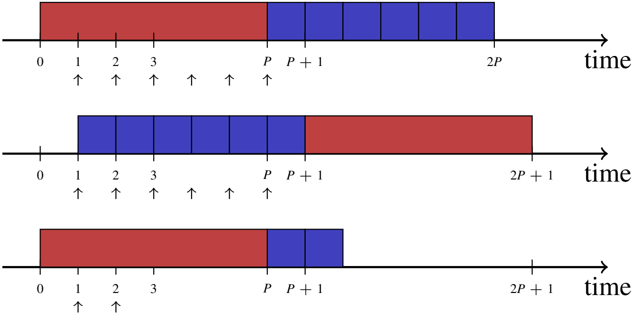
- P small jobs
- each small job has flow time P
- while in the optimal it has flow time 1
- but we can reject

Intuition behind Rejection



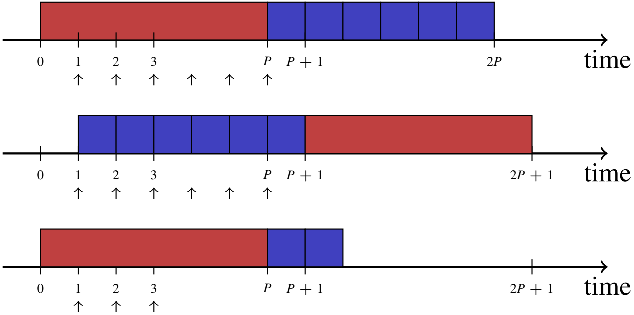
- P small jobs
- each small job has flow time P
- while in the optimal it has flow time 1
- but we can reject

Intuition behind Rejection



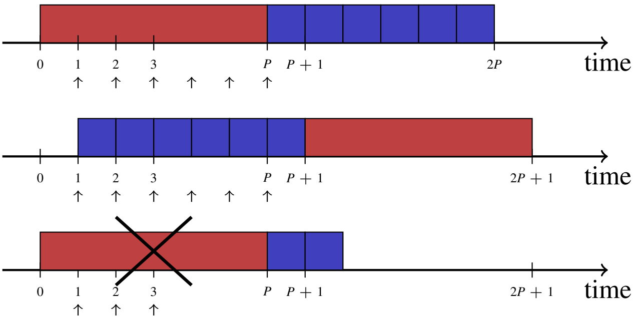
- P small jobs
- each small job has flow time P
- while in the optimal it has flow time 1
- but we can reject

Intuition behind Rejection



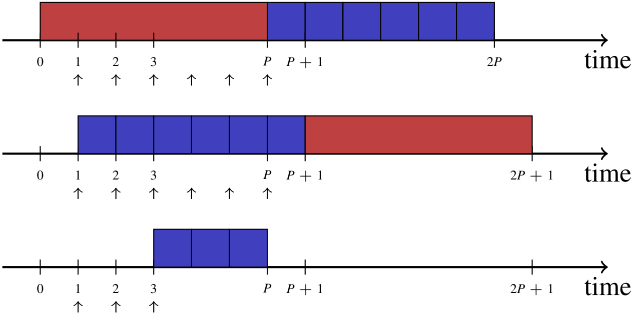
- P small jobs
- each small job has flow time P
- while in the optimal it has flow time 1
- but we can reject

Intuition behind Rejection



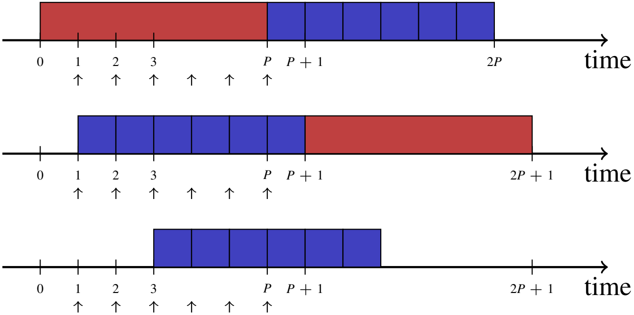
- P small jobs
- each small job has flow time P
- while in the optimal it has flow time 1
- but we can reject

Intuition behind Rejection



- P small jobs
- each small job has flow time P
- while in the optimal it has flow time 1
- but we can reject

Intuition behind Rejection



- P small jobs
- each small job has flow time P
- while in the optimal it has flow time 1
- but we can reject

Rejection Policy

- $\epsilon_r \in (0, 1)$: the rejection constant

Rejection Policy

- $\epsilon_r \in (0,1)$: the rejection constant

1. At the beginning of the execution of job k on machine i
   $\Rightarrow$ introduce a counter $v_k = 0$
2. Each time a job j, with $p_{ij} < p_{ik}$, arrives during the execution of k and j is dispatched to machine i
   $v_k \leftarrow v_k + 1$
3. Interrupt and reject k the first time where $v_k \geq \frac{1}{\epsilon_r}$

Rejection Policy

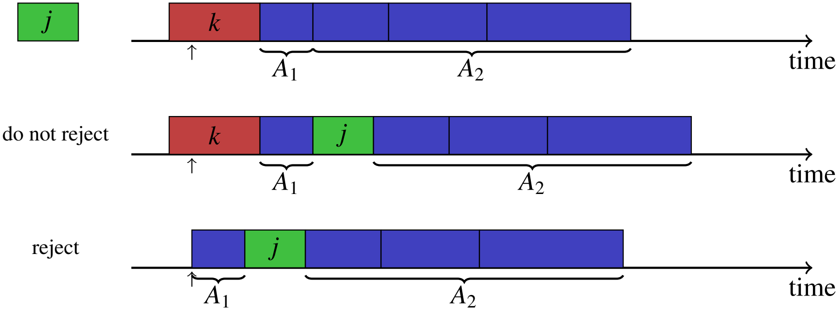
- $\epsilon_r \in (0, 1)$: the rejection constant

1. At the beginning of the execution of job k on machine i
   $\Rightarrow$ introduce a counter $v_k = 0$
2. Each time a job j, with $p_{ij} < p_{ik}$, arrives during the execution of k and j is dispatched to machine i
   $v_k \leftarrow v_k + 1$
3. Interrupt and reject k the first time where $v_k \geq \frac{1}{\epsilon_r}$

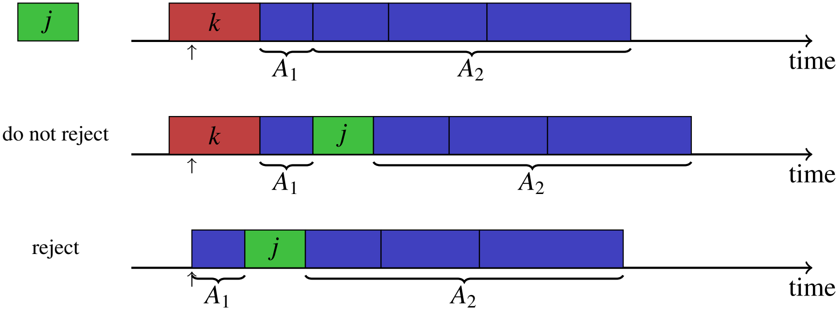
Lemma

We reject at most an ϵ_r-fraction of the jobs

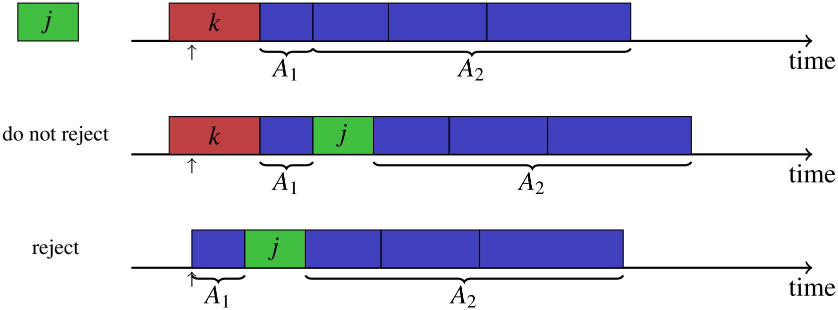
Scheduling Policy



Scheduling Policy



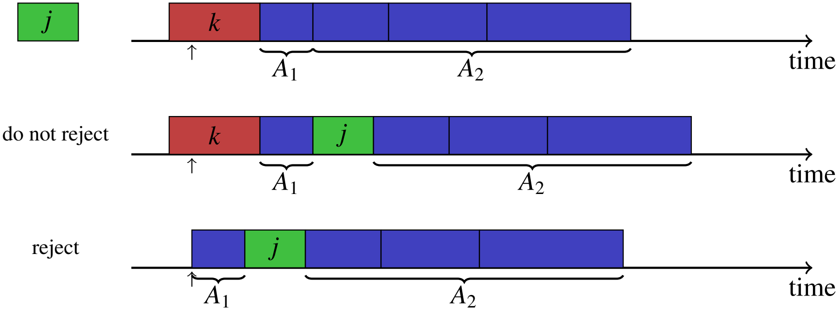
Scheduling Policy



- For each machine i

  schedule the jobs dispatched on i in Shortest Processing Time order

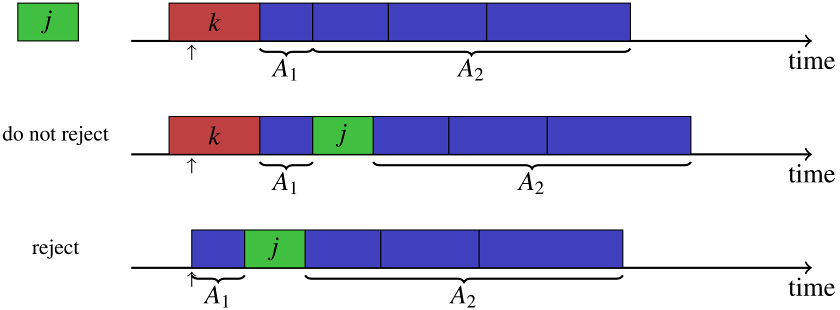
Scheduling Policy



Marginal increase

- A_1: set of jobs with shorter processing time than j
  contribute to the flow time of the new job j
- A_2: set of jobs with longer processing time than j
  the new job j delays them by p_{ij}

Scheduling Policy



Marginal increase

$$\Delta_{ij} = \begin{cases} \left(p_{ik}(r_j) + \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} \right) + |A_2| \cdot p_{ij} & \text{if } k \text{ is not rejected} \\ \sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + \left(|A_2| \cdot p_{ij} - |A_1 \cup A_2| \cdot p_{ik}(r_j) \right) & \text{otherwise} \end{cases}$$

Charging Marginal Increase

Marginal increase

$$\Delta_{ij} \leq \begin{cases} p_{ik}(r_j) + \left(\sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{if } k \text{ is not rejected} \\ \left(\sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{otherwise} \end{cases}$$

Recall rejection: increase the counter of k only if j has smaller processing time

Define:

$$\lambda_{ij} = \begin{cases} \frac{1}{\epsilon_r} p_{ij} + \left(\sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{if } p_{ij} < p_{ik} \\ \frac{1}{\epsilon_r} p_{ij} + p_{ik}(r_j) + \left(\sum_{\ell \in A_1 \cup \{j\}} p_{i\ell} + |A_2| \cdot p_{ij} \right) & \text{otherwise} \end{cases}$$

Dispatching Policy

- Immediate dispatch at arrival and never change this decision
- Dispatch j to the machine i of minimum λ_{ij}

Dual Variables

- $\lambda_j = \min_i \lambda_{ij}$
- $(1 + \epsilon_s).\gamma_i(t) =$number of pending jobs on machine i

Dual Variables

- $\lambda_j = \min_i \lambda_{ij}$
- $(1 + \epsilon_s).\gamma_i(t)$ =number of pending jobs on machine i

Recall dual objective

$$\sum_{j \in \mathcal{J}} \lambda_j - \sum_{i \in \mathcal{M}} \int_0^\infty \gamma_i(t) dt$$

Dual Variables

- $\lambda_j = \min_i \lambda_{ij}$
- $(1+\epsilon_s).\gamma_i(t)$ =number of pending jobs on machine i

Recall dual objective

$$\boxed{\sum_{j \in \mathcal{J}} \lambda_j} - \sum_{i \in \mathcal{M}} \int_0^\infty \gamma_i(t)dt$$

$\geq$ total marginal increase

= total flow time

Dual Variables

- $\lambda_j = \min_i \lambda_{ij}$
- $(1 + \epsilon_s).\gamma_i(t)$ =number of pending jobs on machine i

Recall dual objective

$$\boxed{\sum_{j \in \mathcal{J}} \lambda_j} - \sum_{i \in \mathcal{M}} \boxed{\int_0^{\infty} \gamma_i(t)dt}$$

$\geq$ total marginal increase = total flow time

= $\left(\frac{1}{1+\epsilon_s}\right)$. total flow time

Putting All Together

- rejection: update the counter of executed job when a new job arrives
  $\Rightarrow$ reject if the counter exceeds a threshold based on ϵ_r

- immediate dispatch: based on minimum λ_{ij}

- schedule: select the pending job of smallest processing time

Putting All Together

- rejection: update the counter of executed job when a new job arrives
  $\Rightarrow$ reject if the counter exceeds a threshold based on ϵ_r
- immediate dispatch: based on minimum λ_{ij}
- schedule: select the pending job of smallest processing time

Theorem

There exists an $(1+\epsilon_s)$-speed ϵ_r-rejection $O\left(\frac{1}{\epsilon_r \epsilon_s}\right)$-competitive algorithm for minimizing total flow on a set of unrelated machines that rejects at most ϵ_r-fraction of total number of jobs.

Our Results

Theorem

There exists an $(1+\epsilon_s)$-speed ϵ_r-rejection $O\left(\frac{1}{\epsilon_r \epsilon_s}\right)$-competitive algorithm for minimizing $\sum w_j F_j$ on a set of unrelated machines that rejects at most ϵ_r-fraction of total weights of jobs.

We also extend our analysis to the general problem of minimizing $\left(\sum w_j {F_j}^k\right)^{1/k}$ on a set of unrelated machines

Theorem

There exists an $(1+\epsilon_s)$-speed ϵ_r-rejection $O\left(\frac{k^{(k+2)/k}}{\epsilon_r^{1/k} \epsilon_s^{(k+2)/k}}\right)$-competitive algorithm that rejects at most ϵ_r-fraction of total weights of jobs.

Conclusion and Future Works

- Rejection is a powerful tool for analysing online scheduling algorithms
- We presented $O(1)$-competitive algorithms for minimizing flow time problems
- No online algorithm with performance guarantee was known

Conclusion and Future Works

- Rejection is a powerful tool for analysing online scheduling algorithms
- We presented $O(1)$-competitive algorithms for minimizing flow time problems
- No online algorithm with performance guarantee was known

- Is speed really necessary ?
- How rejections can be extended to other scheduling problems?
- Can rejections be a powerful tool for other online combinatorial algorithms?

Publications

1. Double Archive Pareto Local Search
   Oded Maler and Abhinav Srivastav
   In Proc. of IEEE Symposium on Computational Intelligence, 2016
2. Online Non-preemptive Scheduling to Optimize Max-stretch on a Single Machine
   Pierre.F Dutot, Erik Saule, Abhinav Srivastav and Denis Trystram
   In Proc. of International Computing and Combinatorics Conference, 2016
3. From Preemptive to Non-preemptive using Rejections
   Giorgio Lucarelli, Abhinav Srivastav and Denis Trystram
   In Proc. of International Computing and Combinatorics Conference, 2016
4. Online Non-preemptive Scheduling in a Resource Augmentation Model based on Duality
   Giorgio Lucarelli, Nguyen.K Thang, Abhinav Srivastav and Denis Trystram
   In Proc. of European Symposium on Algorithms, 2016

Thank You!