No-Wait Flowshop Scheduling is as Hard as Asymmetric Traveling Salesman Problem

Marcin Mucha (University of Warsaw)
Maxim Sviridenko (University of Warwick, DIMAP)

FND 2013, Fields Institute, Toronto
1 August 2013

European Research Council
Established by the European Commission

Problem (No-Wait Flowshop)

- *We are given n jobs* $J_1, \ldots, J_n$ *on a sequence of m machines* $M_1, \ldots, M_m$.

No-Wait Flowshop

Problem (No-Wait Flowshop)

- *We are given n jobs* $J_1, \ldots, J_n$ *on a sequence of m machines* $M_1, \ldots, M_m$.
- *Each job is a sequence of operations* $O_{j1}, \ldots, O_{jm}$ *with processing times* $t_{j1}, \ldots, t_{jm}$.

No-Wait Flowshop

Problem (No-Wait Flowshop)

- *We are given n jobs $J_1, \ldots, J_n$ on a sequence of m machines $M_1, \ldots, M_m$.*
- *Each job is a sequence of operations $O_{j1}, \ldots, O_{jm}$ with processing times $t_{j1}, \ldots, t_{jm}$.*
- *Jobs can not be paused once started (no-wait scheduling).*

No-Wait Flowshop

Problem (No-Wait Flowshop)

- *We are given n jobs* $J_1, \ldots, J_n$ *on a sequence of m machines* $M_1, \ldots, M_m$.
- *Each job is a sequence of operations* $O_{j1}, \ldots, O_{jm}$ *with processing times* $t_{j1}, \ldots, t_{jm}$.
- *Jobs can not be paused once started (no-wait scheduling).*
- *Every machine runs the jobs in the same order – no overtaking (permutation scheduling).*

No-Wait Flowshop

Problem (No-Wait Flowshop)

- *We are given n jobs $J_1, \ldots, J_n$ on a sequence of m machines $M_1, \ldots, M_m$.*
- *Each job is a sequence of operations $O_{j1}, \ldots, O_{jm}$ with processing times $t_{j1}, \ldots, t_{jm}$.*
- *Jobs can not be paused once started (no-wait scheduling).*
- *Every machine runs the jobs in the same order – no overtaking (permutation scheduling).*
- *Find job permutation which minimizes makespan.*

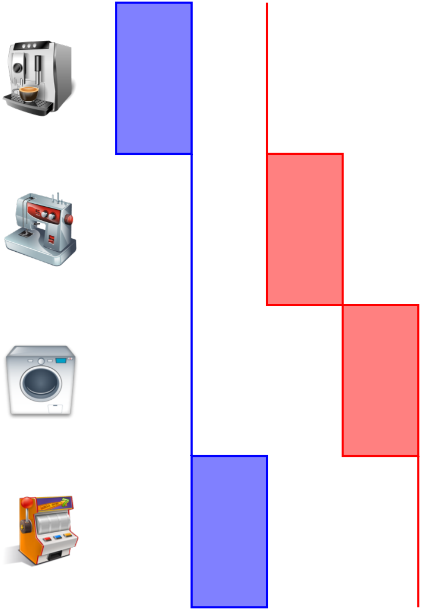
No-Wait Flowshop

Problem (No-Wait Flowshop)

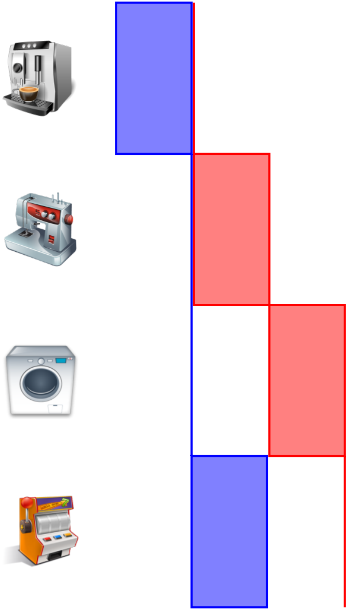
- *We are given n jobs $J_1, \dots, J_n$ on a sequence of m machines $M_1, \dots, M_m$.*
- *Each job is a sequence of operations $O_{j1}, \dots, O_{jm}$ with processing times $t_{j1}, \dots, t_{jm}$.*
- *Jobs can not be paused once started (no-wait scheduling).*
- *Every machine runs the jobs in the same order – no overtaking (permutation scheduling).*
- *Find job permutation which minimizes makespan.*

Think: production line for steel manufacturing, or production units with no intermediate storage capacity.

Example

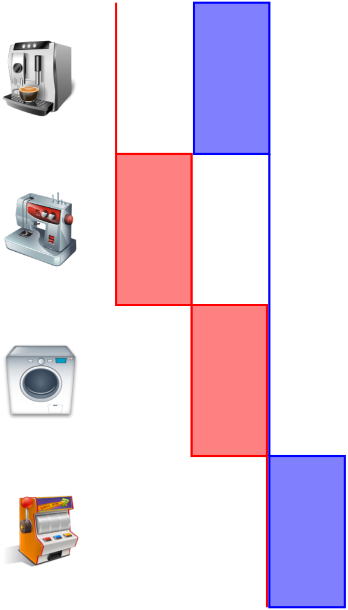


Example – OK



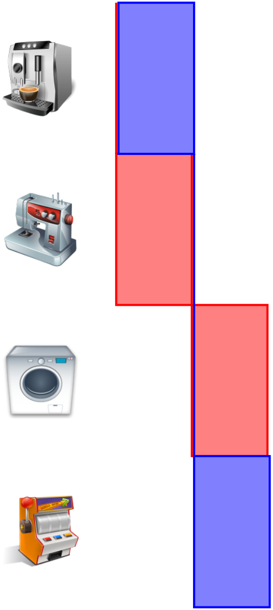
OK

Example – OK



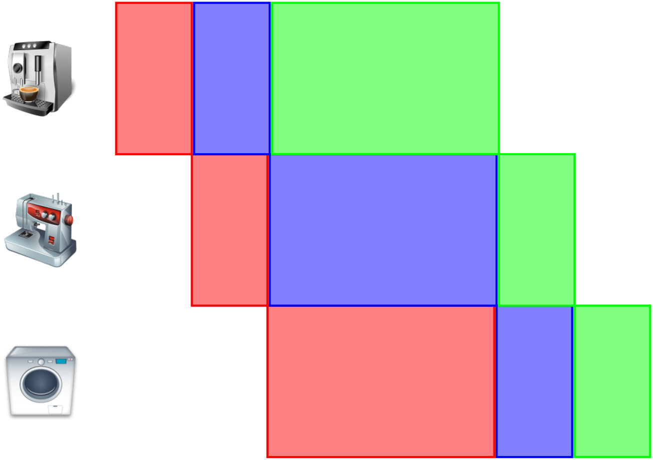
OK

Example – Not OK!

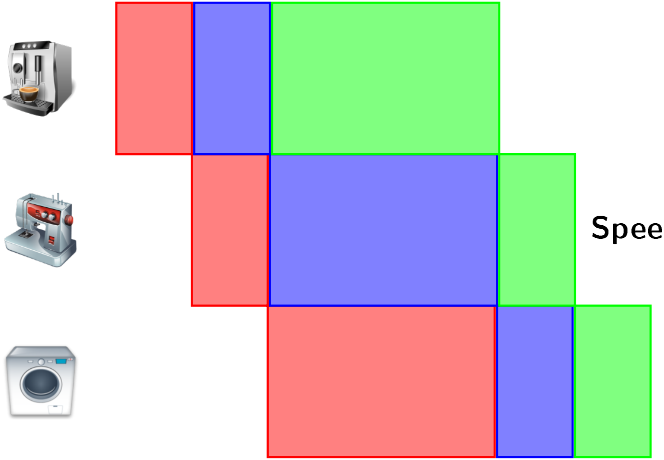


Not OK!

Curiouser and curiouser

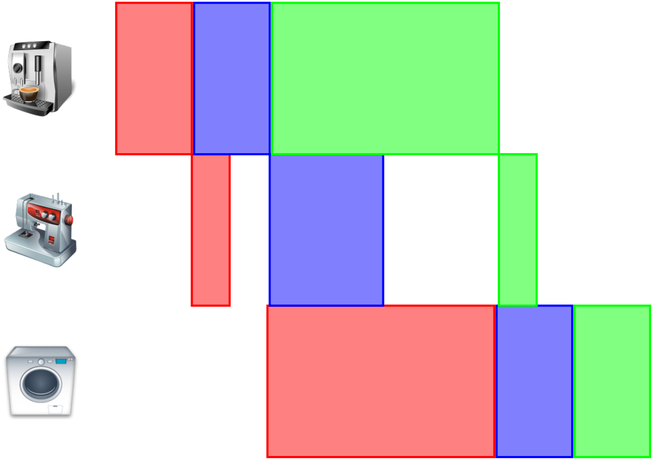


Curiouser and curiouser

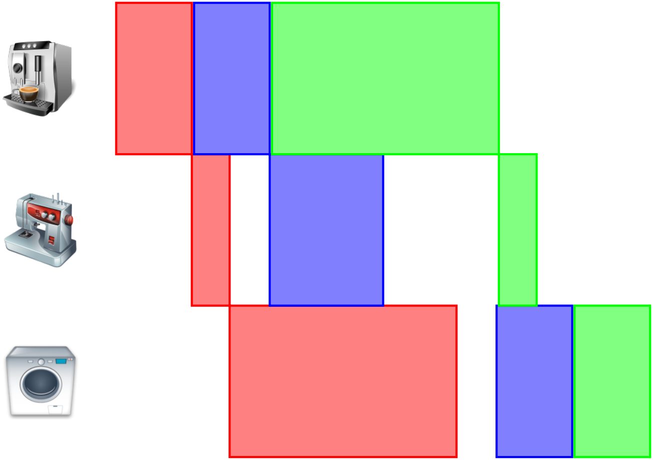


Speed up x2

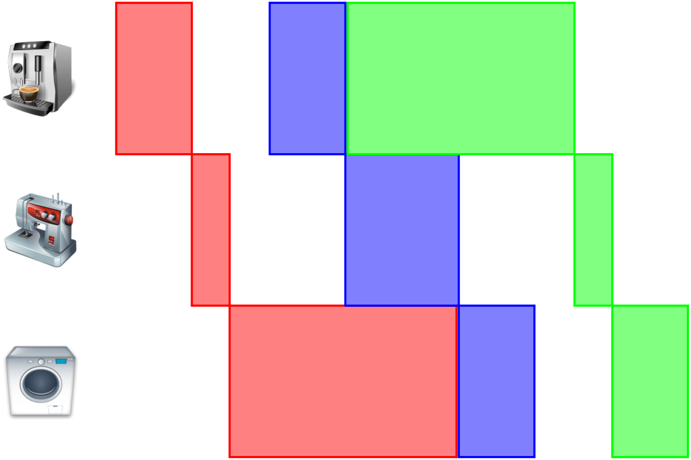
Curiouser and curiouser



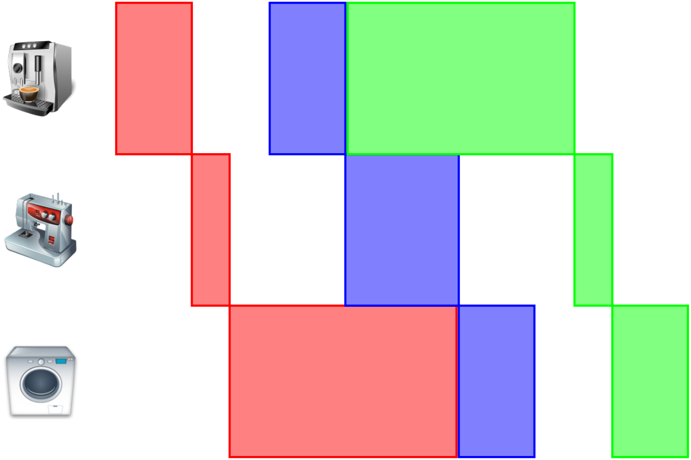
Curiouser and curiouser



Curiouser and curiouser



Curiouser and curiouser



Example due to Spieksma and Woeginger (2005).

Known results

Gilmore,Gomory (1964) $O(n \log n)$ algorithm for $m = 2$.

Known results

Gilmore, Gomory (1964) $O(n \log n)$ algorithm for $m = 2$.

Papadimitriou, Kanellakis (1980) Strongly NP-hard for $m \geq 4$. $\star$

Known results

Gilmore,Gomory (1964) $O(n \log n)$ algorithm for $m = 2$.

Papadimitriou,Kanellakis (1980) Strongly NP-hard for $m \geq 4$. $\star$

Röck,Schmidt (1983) $\lceil \frac{m}{2} \rceil$-approximation. $\star$

Known results

Gilmore, Gomory (1964) $O(n \log n)$ algorithm for $m = 2$.

Papadimitriou, Kanellakis (1980) Strongly NP-hard for $m \geq 4$. $\star$

Röck, Schmidt (1983) $\lceil \frac{m}{2} \rceil$-approximation. $\star$

Röck (1984) Strongly NP-hard for $m \geq 3$. $\star$

Known results

Gilmore, Gomory (1964) $O(n \log n)$ algorithm for $m = 2$.

Papadimitriou, Kanellakis (1980) Strongly NP-hard for $m \geq 4$. $\star$

Röck, Schmidt (1983) $\lceil \frac{m}{2} \rceil$-approximation. $\star$

Röck (1984) Strongly NP-hard for $m \geq 3$. $\star$

Sviridenko (2003) PTAS for $m = O(1)$.

Known results

Gilmore,Gomory (1964) $O(n \log n)$ algorithm for $m = 2$.

Papadimitriou,Kanellakis (1980) Strongly NP-hard for $m \geq 4$. $\star$

Röck,Schmidt (1983) $\lceil \frac{m}{2} \rceil$-approximation. $\star$

Röck (1984) Strongly NP-hard for $m \geq 3$. $\star$

Sviridenko (2003) PTAS for $m = O(1)$.

No-Wait Flowshop reduces to ATSP, so:

Known results

Gilmore,Gomory (1964) $O(n \log n)$ algorithm for $m = 2$.

Papadimitriou,Kanellakis (1980) Strongly NP-hard for $m \geq 4$. $\star$

Röck,Schmidt (1983) $\lceil \frac{m}{2} \rceil$-approximation. $\star$

Röck (1984) Strongly NP-hard for $m \geq 3$. $\star$

Sviridenko (2003) PTAS for $m = O(1)$.

No-Wait Flowshop reduces to ATSP, so:

Frieze, Galbiati, Maffioli (1982) $O(\log n)$-approximation. $\star$

Asadpour et al.(2010) $O(\log n / \log \log n)$-approximation.

Known results

Gilmore,Gomory (1964) $O(n \log n)$ algorithm for $m = 2$.

Papadimitriou,Kanellakis (1980) Strongly NP-hard for $m \geq 4$. $\star$

Röck,Schmidt (1983) $\lceil \frac{m}{2} \rceil$-approximation. $\star$

Röck (1984) Strongly NP-hard for $m \geq 3$. $\star$

Sviridenko (2003) PTAS for $m = O(1)$.

No-Wait Flowshop reduces to ATSP, so:

Frieze, Galbiati, Maffioli (1982) $O(\log n)$-approximation. $\star$

Asadpour et al.(2010) $O(\log n / \log \log n)$-approximation.

Is No-Wait Flowshop an easy case of ATSP?

Our results

Theorem (Main Result 1)

No-Wait Flowshop is as hard as ATSP, in particular APX-hard.

Our results

Theorem (Main Result 1)

No-Wait Flowshop is as hard as ATSP, in particular APX-hard.

Theorem (Main Result 2)

There is an $O(\log m)$-approximation algorithm for No-Wait Flowshop.

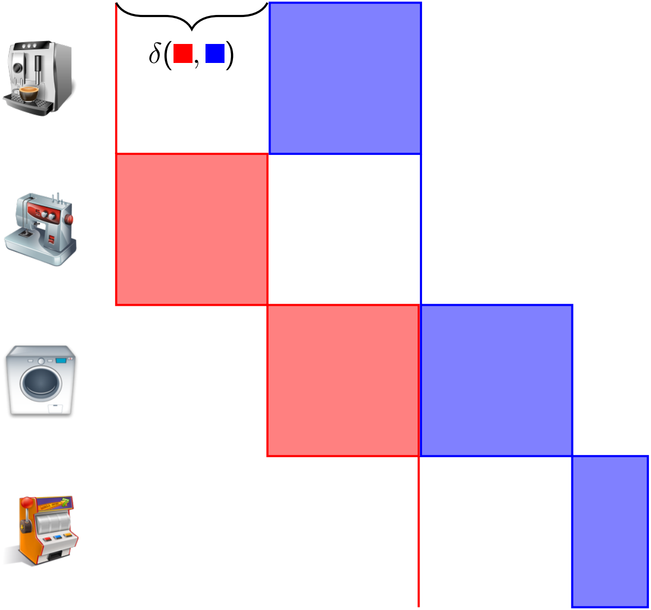
Overview

1. Reduction to ATSP
2. Encoding semi-metrics in No-Wait Flowshop
3. $O(\log m)$-approximation

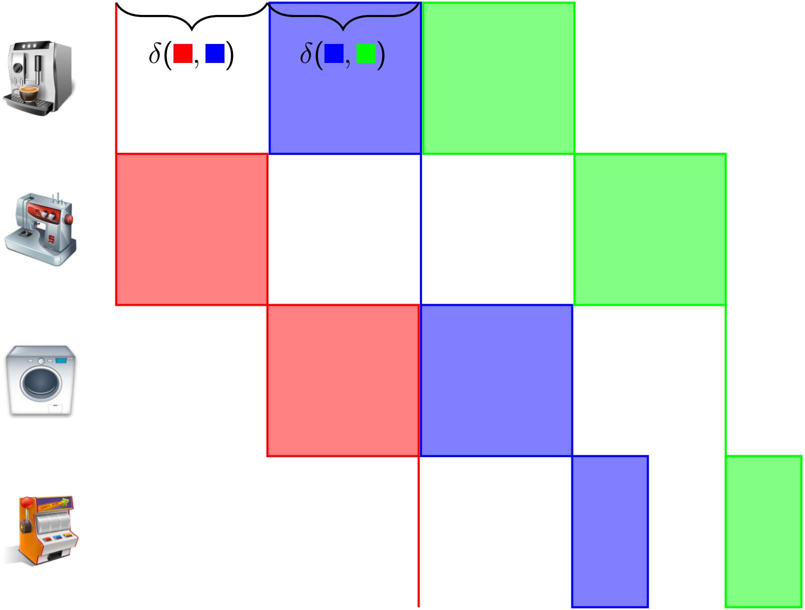
Overview

1. Reduction to ATSP
2. Encoding semi-metrics in No-Wait Flowshop
3. $O(\log m)$-approximation

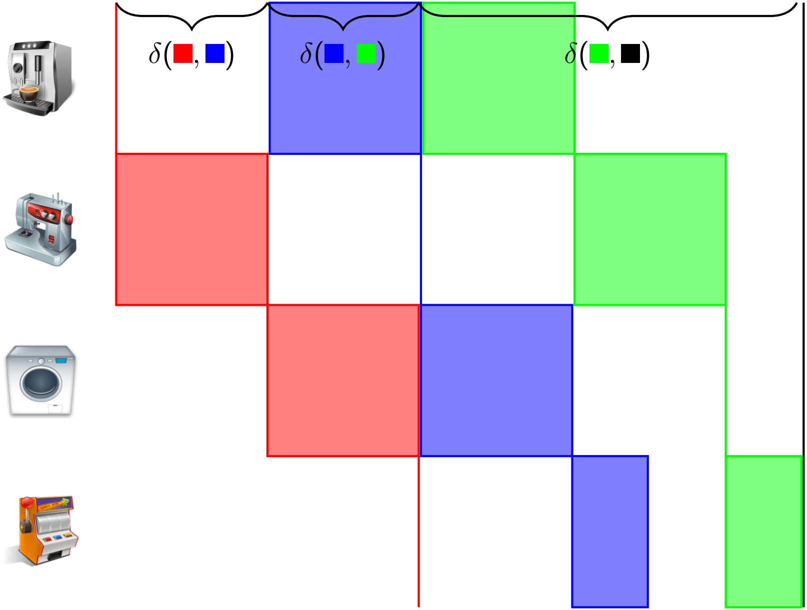
Job distance



Job distance



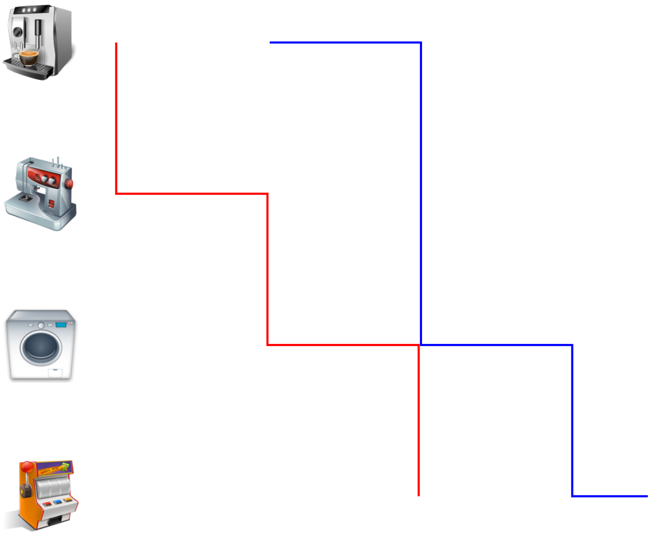
No-Wait Flowshop as ATSP



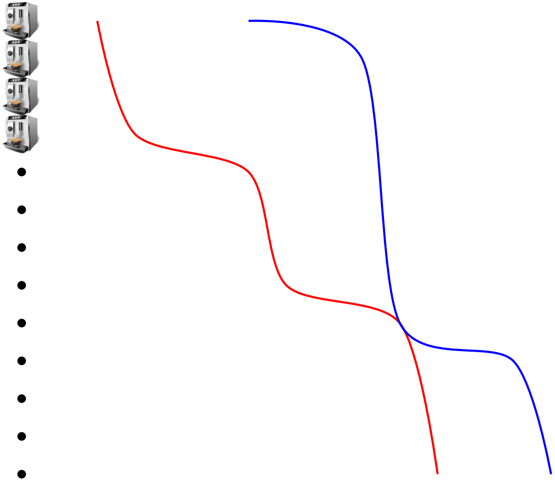
Overview

1. Reduction to ATSP
2. Encoding semi-metrics in No-Wait Flowshop
3. $O(\log m)$-approximation

Four machines



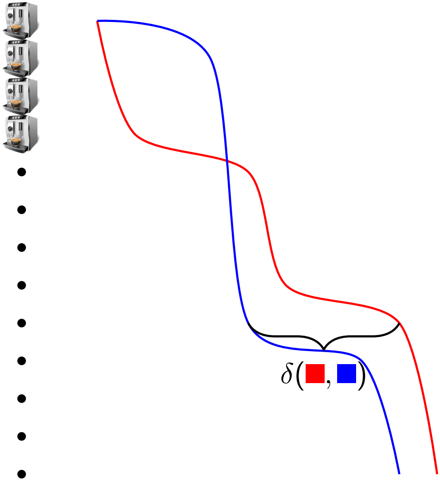
Many machines



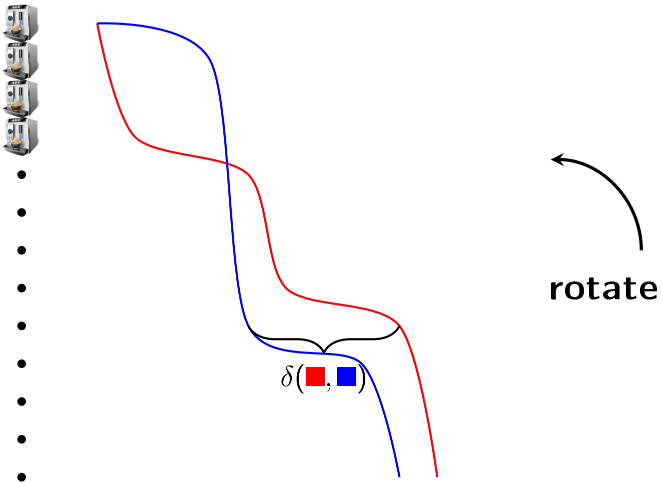
many machines

small operations

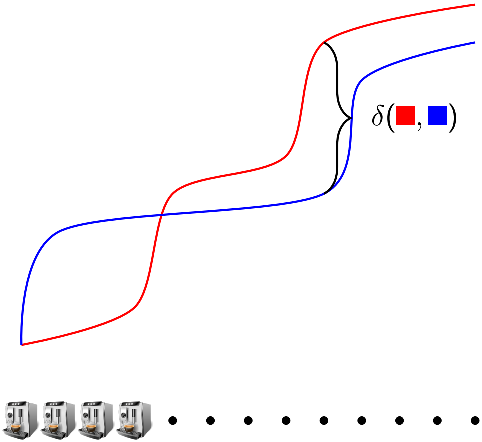
Many machines



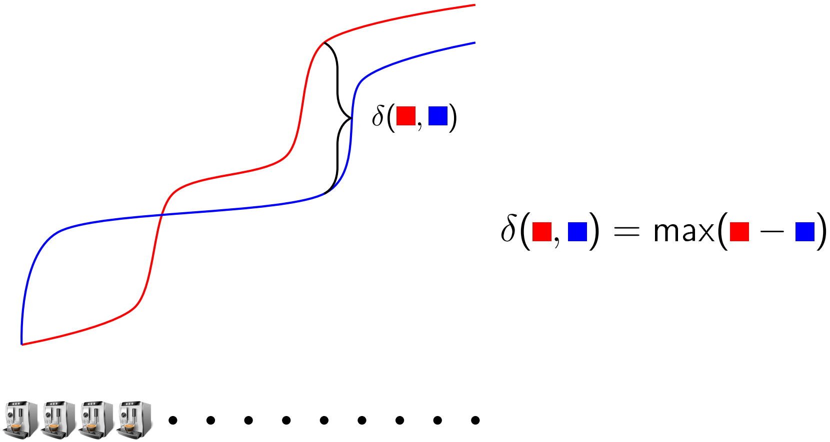
Many machines



Many machines



Many machines



No-Wait Flowshop distance is hard!

Fact

Any n-point semi-metric (V, d) embeds isometrically into the semi-metric $(\mathbb{R}^n, \delta)$ with

$$\delta(x, y) = \max(x - y).$$

No-Wait Flowshop distance is hard!

Fact

Any n-point semi-metric (V, d) embeds isometrically into the semi-metric $(\mathbb{R}^n, \delta)$ with

$$\delta(x, y) = \max(x - y).$$

Proof.

$$F(v) = (d(v, v_1), \ldots, d(v, v_n)).$$

□

No-Wait Flowshop distance is hard!

Fact

Any n-point semi-metric (V, d) embeds isometrically into the semi-metric $(\mathbb{R}^n, \delta)$ with

$$\delta(x, y) = \max(x - y).$$

Proof.

$$F(v) = (d(v, v_1), \ldots, d(v, v_n)).$$

□

Message: Wo-Wait Flowshop distance functions are as hard as general semi-metrics.

No-Wait Flowshop distance is hard!

Fact

Any n-point semi-metric (V, d) embeds isometrically into the semi-metric $(\mathbb{R}^n, \delta)$ with

$$\delta(x, y) = \max(x - y).$$

Proof.

$$F(v) = (d(v, v_1), \ldots, d(v, v_n)).$$

□

Message: Wo-Wait Flowshop distance functions are as hard as general semi-metrics.

Theorem (Main Result 1)

No-Wait Flowshop is as hard as ATSP, in particular APX-hard.

Why is it not over

- Our functions are monotone.

Why is it not over

- Our functions are monotone.
- Our functions are ε-Lipschitz (ε – max. operation size)

Why is it not over

- Our functions are monotone.
- Our functions are ε-Lipschitz (ε – max. operation size)

Why is it not over

- Our functions are monotone.
- Our functions are ε-Lipschitz (ε – max. operation size), so...
- The number of machines depends on max. ATSP distance W.

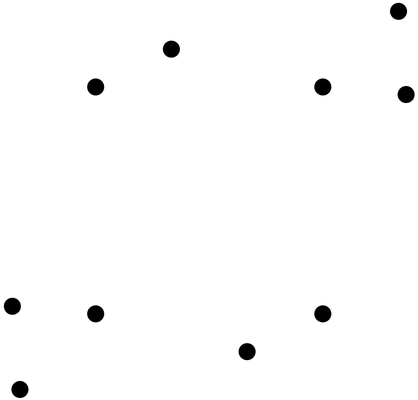
Why is it not over

- Our functions are monotone.
- Our functions are ε-Lipschitz (ε – max. operation size), so...
- The number of machines depends on max. ATSP distance W.
- The total length of each job is $\Omega(nW) >> OPT$.

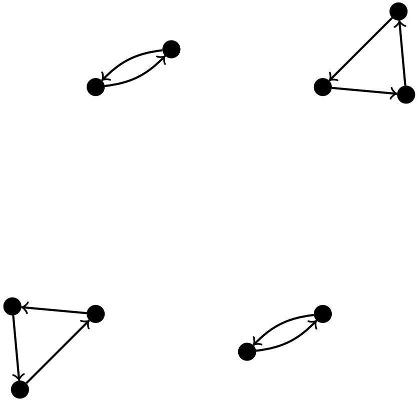
Overview

1. Reduction to ATSP
2. Encoding semi-metrics in No-Wait Flowshop
3. $O(\log m)$-approximation

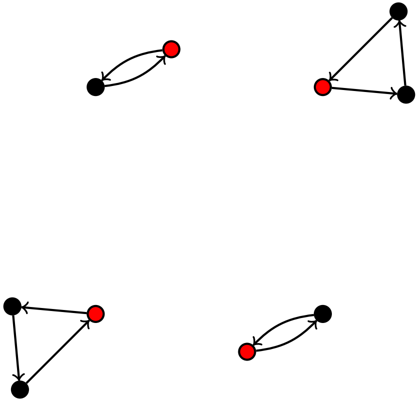
Frieze, Galbiati, Maffioli



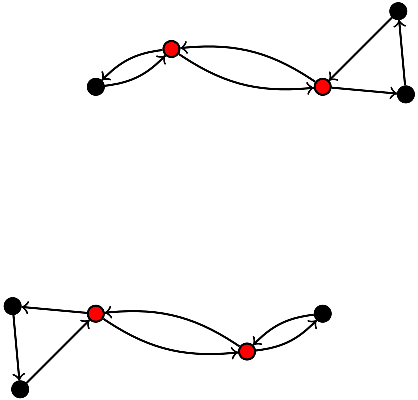
Frieze, Galbiati, Maffioli



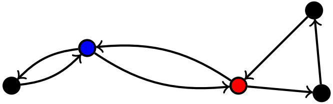
Frieze, Galbiati, Maffioli

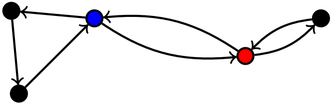


Frieze, Galbiati, Maffioli

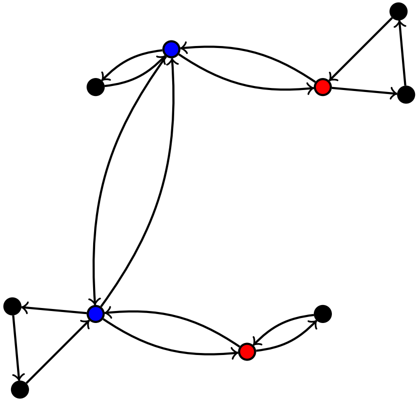


Frieze, Galbiati, Maffioli





Frieze, Galbiati, Maffioli



$O(\log m)$-approximation

- Always choose the shortest representative.
- Stop after $\log m$ rounds and add an arbitrary Hamiltonian cycle.

$O(\log m)$-approximation

- Always choose the shortest representative.
- Stop after $\log m$ rounds and add an arbitrary Hamiltonian cycle.

Theorem

This algorithm is a $O(\log m)$-approximation.

$O(\log m)$-approximation

Proof.

Let $r_1, \ldots, r_k$ be the representative vertices after $\log m$ rounds and $L(r_1), \ldots, L(r_k)$ their lengths.

$O(\log m)$-approximation

Proof.

Let $r_1, \ldots, r_k$ be the representative vertices after $\log m$ rounds and $L(r_1), \ldots, L(r_k)$ their lengths.

The total length of jobs in r_i's component is at least $mL(r_i)$, so on m machines they require at least $L(r_i)$ time to schedule.

$O(\log m)$-approximation

Proof.

Let $r_1, \ldots, r_k$ be the representative vertices after $\log m$ rounds and $L(r_1), \ldots, L(r_k)$ their lengths.

The total length of jobs in r_i's component is at least $mL(r_i)$, so on m machines they require at least $L(r_i)$ time to schedule.

This gives $OPT \geq \sum_i L(r_i)$.

$O(\log m)$-approximation

Proof.

Let $r_1, \ldots, r_k$ be the representative vertices after $\log m$ rounds and $L(r_1), \ldots, L(r_k)$ their lengths.

The total length of jobs in r_i's component is at least $mL(r_i)$, so on m machines they require at least $L(r_i)$ time to schedule.

This gives $OPT \geq \sum_i L(r_i)$.

The Hamiltonian cycle we add has at most this length. $\square$

Summary and Final Remarks

Gilmore, Gomory (1964) $O(n \log n)$ algorithm for $m = 2$.

Papadimitriou, Kanellakis (1980) Strongly NP-hard for $m \geq 4$. $\star$

Röck, Schmidt (1983) $\lceil \frac{m}{2} \rceil$-approximation. $\star$

Röck (1984) Strongly NP-hard for $m \geq 3$. $\star$

Sviridenko (2003) PTAS for $m = O(1)$.

Asadpour et al.(2010) $O(\log n / \log \log n)$ approximation via ATSP.

New result ATSP-hardness (for $m = polyn(n)$).

New result $O(\log m)$-approximation.

Open problems

Problem (1)

Bridge the gap betweeen a PTAS and $O(\log m)$.
$O(1)$*-approximation for some range of* m*?*

Open problems

Problem (1)

Bridge the gap betweeen a PTAS and $O(\log m)$.
$O(1)$-approximation for some range of m?

Problem (2)

Can you get $O(\log m / \log\log m)$-approximation?

Open problems

Problem (1)

Bridge the gap betweeen a PTAS and $O(\log m)$*.*
$O(1)$*-approximation for some range of* m*?*

Problem (2)

Can you get $O(\log m / \log \log m)$*-approximation?*

Problem (3)

Can you get $O(\log n / \log \log n)$*-approximation for ATSP à la Frieze, Galbiati, Maffioli?*

Thanks!