Beachcombing on Strips and Islands

Evangelos Bampas[1] Jurek Czyzowicz[2]
David Ilcinkas[3] Ralf Klasing[3]

[1]Aix-Marseille Université (LIF), France
[2]Université du Québec en Outaouais, Canada
[3]CNRS & Université de Bordeaux (LaBRI), France

GT Algorithmique Distribuée
December 14, 2015

The Beachcombers' problem

Setting

- Robots have to search every point of a given domain
- They have different maximum searching speeds s_i and maximum walking speeds $w_i > s_i$

Motivation: Careful searching (or some other action) takes more time than casual traversal of the domain.

Goal

Minimize the time taken until all points of the domain have been searched.

The Beachcombers' problem

Setting

- Robots have to search every point of a given domain
- They have different maximum searching speeds s_i and maximum walking speeds $w_i > s_i$

Motivation: Careful searching (or some other action) takes more time than casual traversal of the domain.

Goal

Minimize the time taken until all points of the domain have been searched.

The Beachcombers' problem

Setting

- Robots have to search every point of a given domain
- They have different maximum searching speeds s_i and maximum walking speeds $w_i > s_i$

Motivation: Careful searching (or some other action) takes more time than casual traversal of the domain.

Goal

Minimize the time taken until all points of the domain have been searched.

The Beachcombers' problem

Setting

- Robots have to search every point of a given domain
- They have different maximum searching speeds s_i and maximum walking speeds $w_i > s_i$

Motivation: Careful searching (or some other action) takes more time than casual traversal of the domain.

Goal

Minimize the time taken until all points of the domain have been searched.

The Beachcombers' problem

Setting

- Robots have to search every point of a given domain
- They have different maximum searching speeds s_i and maximum walking speeds $w_i > s_i$

Motivation: Careful searching (or some other action) takes more time than casual traversal of the domain.

Goal

Minimize the time taken until all points of the domain have been searched.

Applications

Searching type

- Beachcombing (beach looking for things of value)
- Looking for leaks/corrosion on pipelines
- Searching the lost Teddy bear after a family excursion
- ⋮

Processing type

- Harvesting a field
- Snow removal and de-icing of roads
- Web page indexing
- ⋮

Applications

Searching type

- Beachcombing (beach looking for things of value)
- Looking for leaks/corrosion on pipelines
- Searching the lost Teddy bear after a family excursion

⋮

Processing type

- Harvesting a field
- Snow removal and de-icing of roads
- Web page indexing

⋮

Basic model

Input

- Set of robots with
  - A maximum walking speed w_i
  - A maximum searching speed $s_i < w_i$
- A domain: so far, the line segment

Additional assumptions

- No communication
- No cost for changing speed, mode, direction

Output

A schedule such that every point of the domain is searched

Basic model

Input

- Set of robots with
  - A maximum walking speed w_i
  - A maximum searching speed $s_i < w_i$
- A domain: so far, the line segment

Additional assumptions

- No communication
- No cost for changing speed, mode, direction

Output

A schedule such that every point of the domain is searched

Basic model

Input

- Set of robots with
  - A maximum walking speed w_i
  - A maximum searching speed $s_i < w_i$
- A domain: so far, the line segment

Additional assumptions

- No communication
- No cost for changing speed, mode, direction

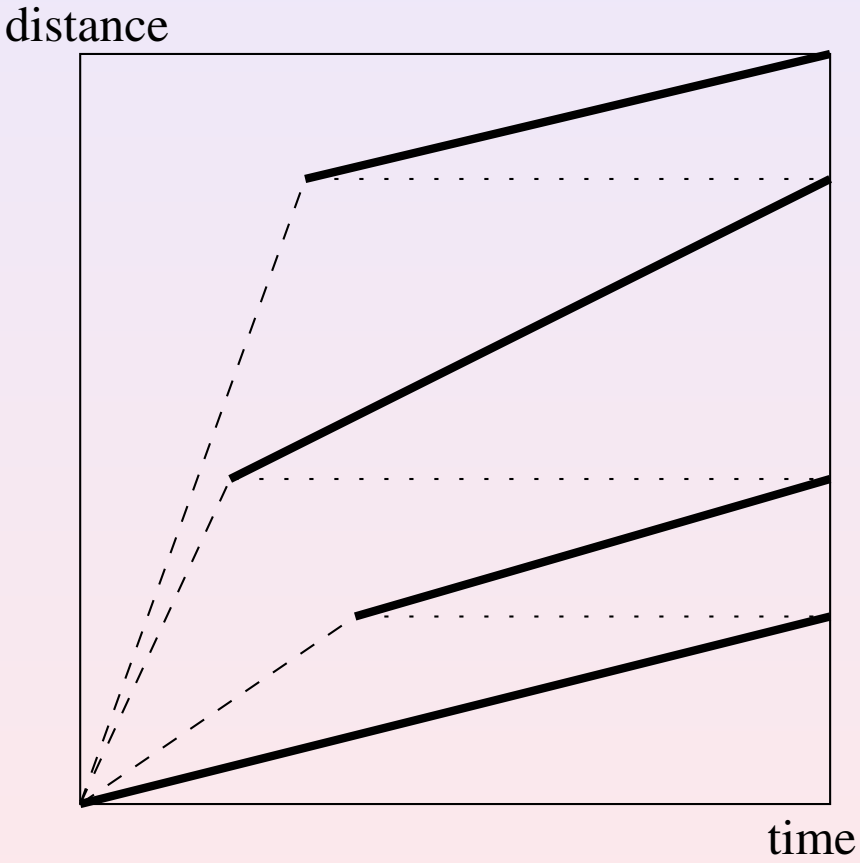
Output

A schedule such that every point of the domain is searched

A schedule

Combing a segment of known length from one endpoint

Optimal solution:
(computable in
polynomial time)



Algorithm COMB [Czyzowicz et al., SIROCCO 2014]

The online variants

Online version: the length of the segment is not known
Alternatively: the domain is a semi-infinite line

Discrete Online Beachcombers' problem

- The segment length is an integer
- Goal: maximizing $\inf_{\ell \in \mathbb{N}^*} \frac{\text{length } \ell}{\text{time of } A \text{ over segment } [0;\ell]}$

Continuous Online Beachcombers' problem

- The segment length is a real (at least 1)
- Goal: maximizing $\inf_{\ell \geq 1} \frac{\text{length } \ell}{\text{time of } A \text{ over segment } [0;\ell]}$

The online variants

Online version: the length of the segment is not known
Alternatively: the domain is a semi-infinite line

Discrete Online Beachcombers' problem

- The segment length is an integer
- Goal: maximizing $\inf_{\ell \in \mathbb{N}^*} \frac{\text{length } \ell}{\text{time of } A \text{ over segment } [0;\ell]}$

Continuous Online Beachcombers' problem

- The segment length is a real (at least 1)
- Goal: maximizing $\inf_{\ell \geq 1} \frac{\text{length } \ell}{\text{time of } A \text{ over segment } [0;\ell]}$

The online variants

Online version: the length of the segment is not known
Alternatively: the domain is a semi-infinite line

Discrete Online Beachcombers' problem

- The segment length is an integer
- Goal: maximizing $\inf_{\ell \in \mathbb{N}^*} \frac{\text{length } \ell}{\text{time of } A \text{ over segment } [0;\ell]}$

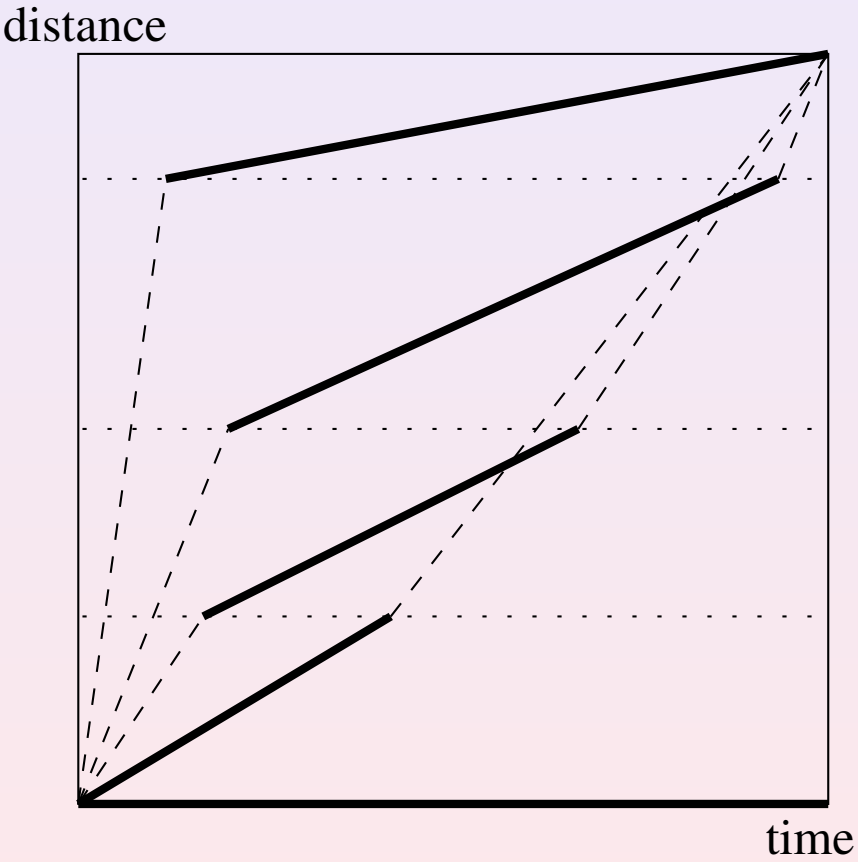
Continuous Online Beachcombers' problem

- The segment length is a real (at least 1)
- Goal: maximizing $\inf_{\ell \geq 1} \frac{\text{length } \ell}{\text{time of } A \text{ over segment } [0;\ell]}$

Another schedule

Combing a segment of unknown length from one endpoint

Algorithm LeapFrog
[Czyzowicz et al., SIROCCO 2014]



Conjectured to be optimal in the discrete online setting

The multi-source Beachcombers' problem

Introduced in [Czyzowicz et al., ALGOSENSORS 2014]

t-source Beachcombers' problem (t-SBP)

Main modifications:

- The robots cannot change direction
- Additional outputs
  - A partition of the robots in at most t groups
  - A source and a direction of move for each of the groups

Their hardness result

2-SBP is NP-hard, even with uniform walking speeds

The multi-source Beachcombers' problem

Introduced in [Czyzowicz et al., ALGOSENSORS 2014]

t-source Beachcombers' problem (t-SBP)

Main modifications:

- The robots cannot change direction
- Additional outputs
  - A partition of the robots in at most t groups
  - A source and a direction of move for each of the groups

Their hardness result

2-SBP is NP-hard, even with uniform walking speeds

Previous work (summary)

Only studied on the segment

[Czyzowicz et al., SIROCCO 2014]

- Offline COMB algorithm: proved optimal
- Online LEAPFROG algorithm: 2-competitive

[Czyzowicz et al., ALGOSENSORS 2014]

- 2-SBP is NP-hard, even with uniform walking speeds
- 0.5 and 0.56 approximation algorithms for 2-SBP
- Poly-time algo for t-SBP with uniform searching speeds
- Randomized algorithm for t-SBP with expected approximation ratio $1 - (1 - 1/t)^t$

Our results

Online Beachcombers' Problem on the **segment**

Algo LeapFrog is optimal, even for the continuous variant

Offline Beachcombers' Problem on the **cycle**

- Equivalent to 2-SBP, thus NP-hard
- Efficient algorithms (from [Czyzowicz et al., ALGOSENSORS 2014])

Offline **Multi-source** Beachcombers' Problem with zigzags

- Zigzags do not help (both in segments and cycles)
- Generalization of the results from [Czyzowicz et al., ALGOSENSORS 2014]

Our results

Online Beachcombers' Problem on the **segment**

Algo LeapFrog is optimal, even for the continuous variant

Offline Beachcombers' Problem on the **cycle**

- Equivalent to 2-SBP, thus NP-hard
- Efficient algorithms
  (from [Czyzowicz et al., ALGOSENSORS 2014])

Offline Multi-source Beachcombers' Problem with zigzags

- Zigzags do not help (both in segments and cycles)
- Generalization of the results from
  [Czyzowicz et al., ALGOSENSORS 2014]

Our results

Online Beachcombers' Problem on the **segment**

Algo LeapFrog is optimal, even for the continuous variant

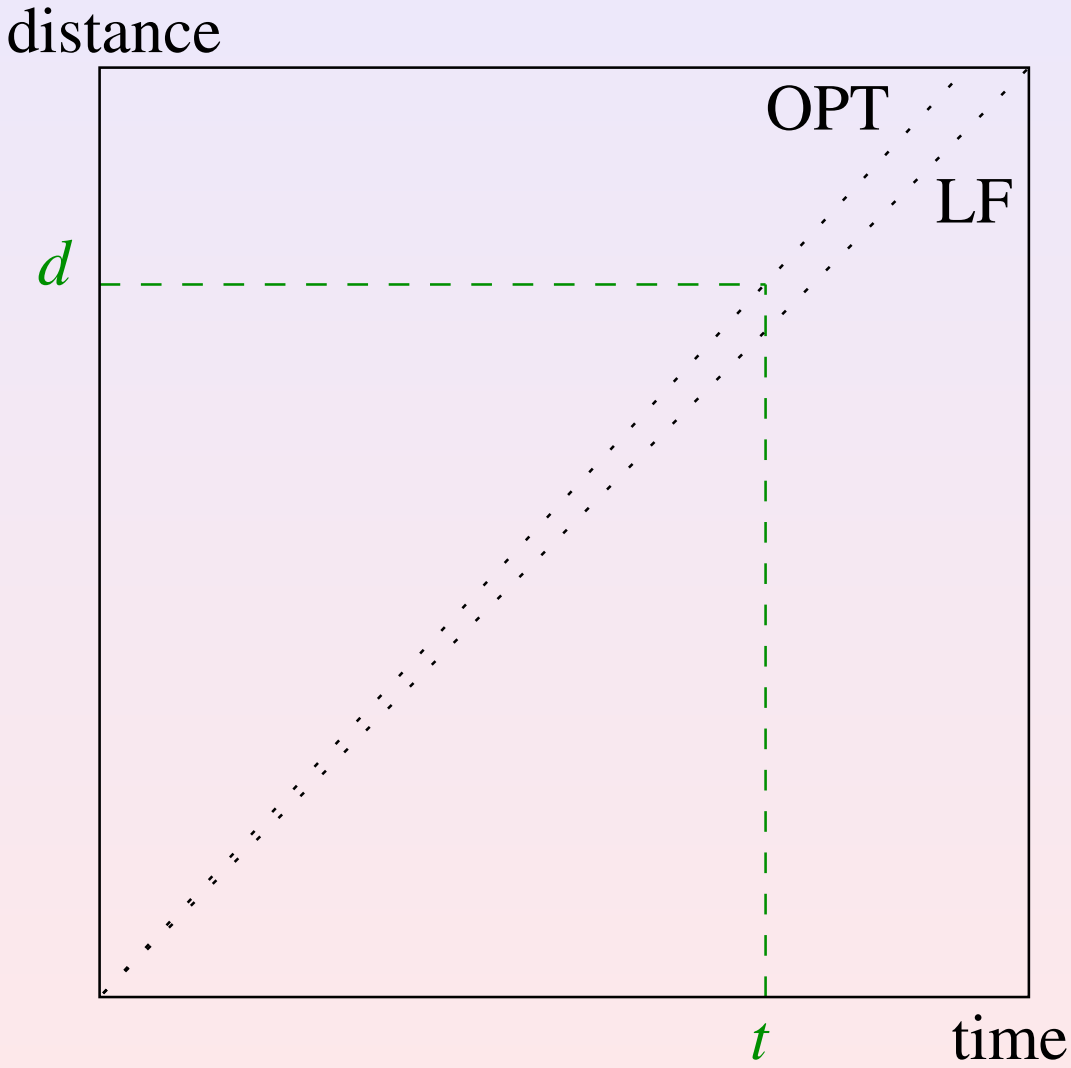
Offline Beachcombers' Problem on the **cycle**

- Equivalent to 2-SBP, thus NP-hard
- Efficient algorithms
  (from [Czyzowicz et al., ALGOSENSORS 2014])

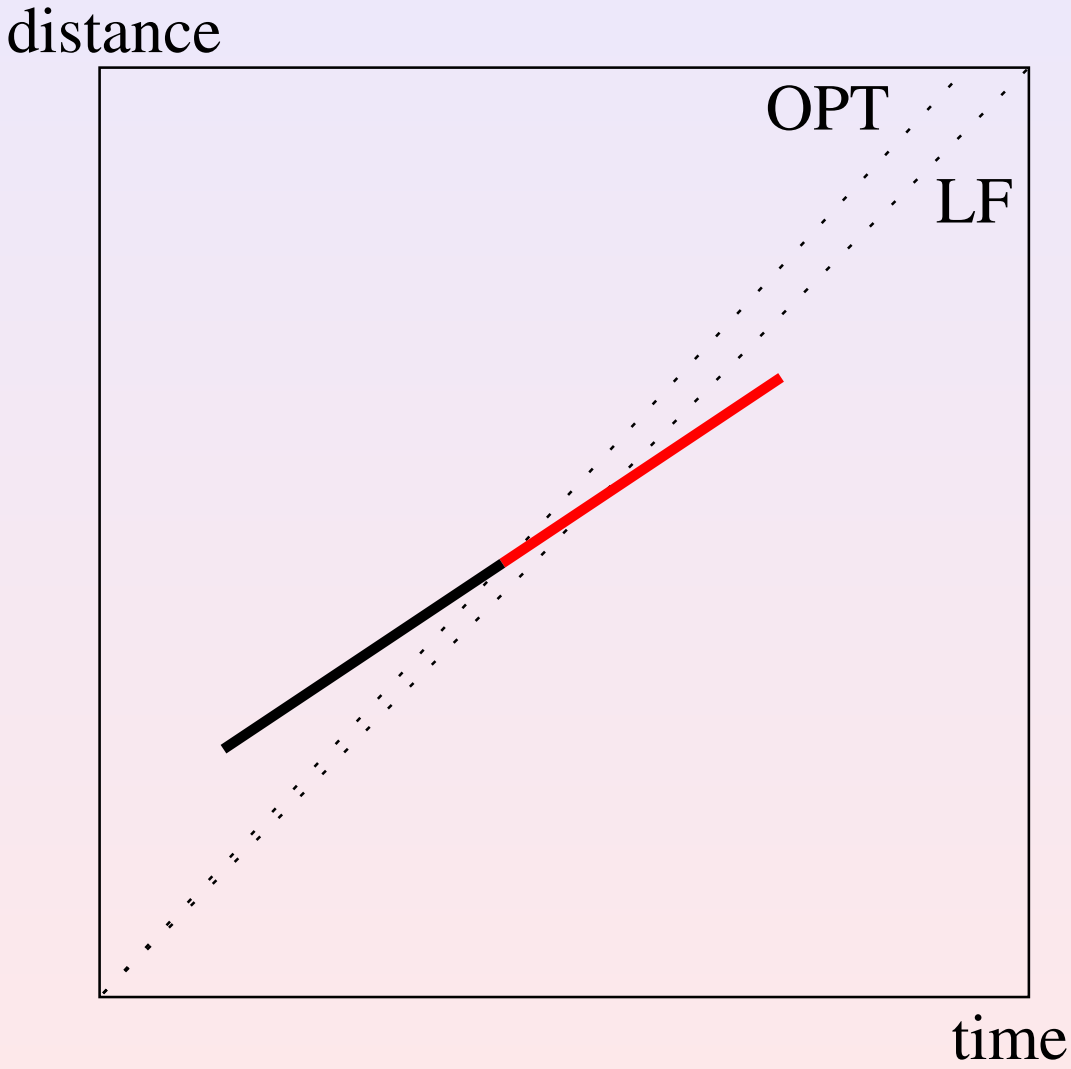
Offline **Multi-source** Beachcombers' Problem with zigzags

- Zigzags do not help (both in segments and cycles)
- Generalization of the results from
  [Czyzowicz et al., ALGOSENSORS 2014]

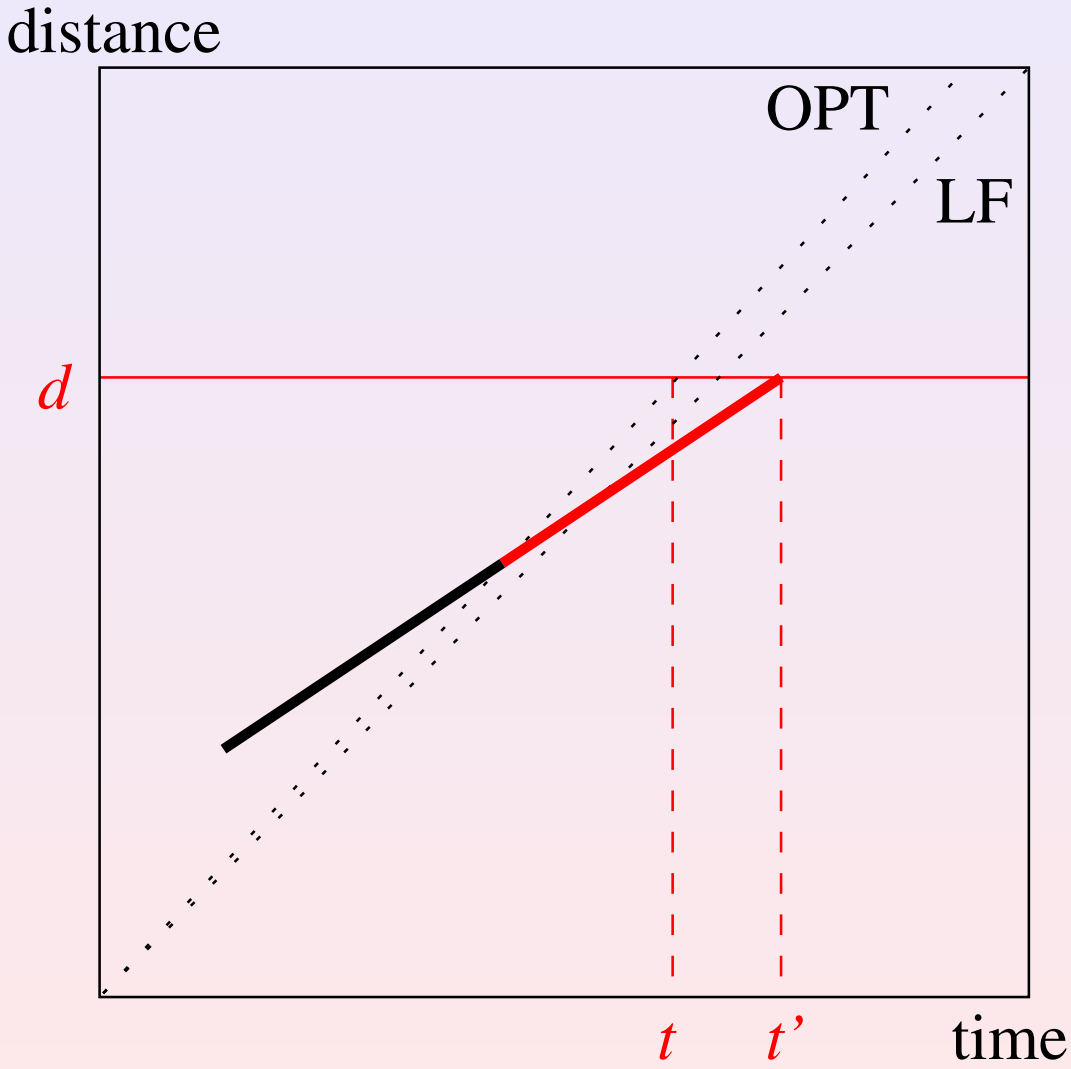
A hint about the optimality of LeapFrog



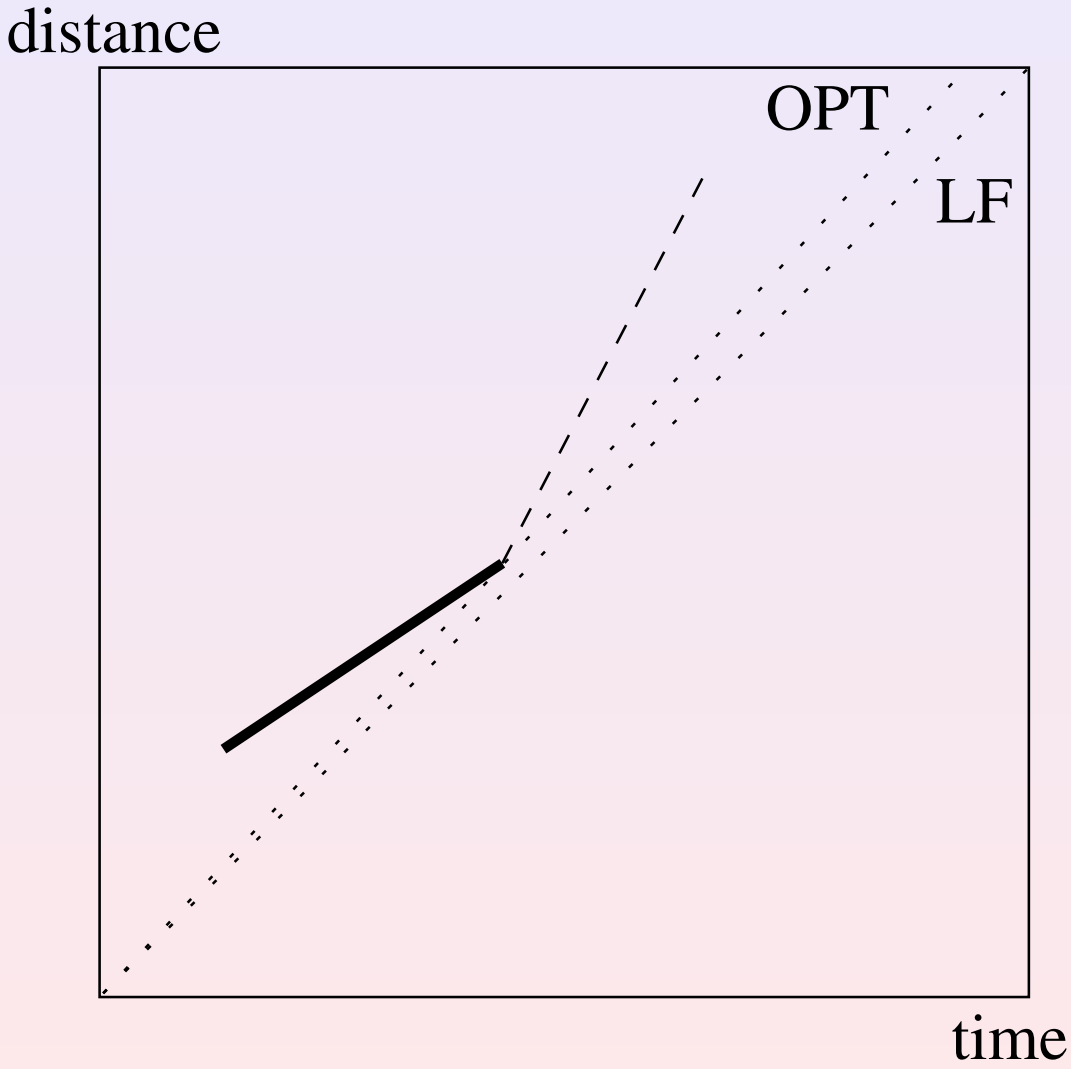
A hint about the optimality of LeapFrog



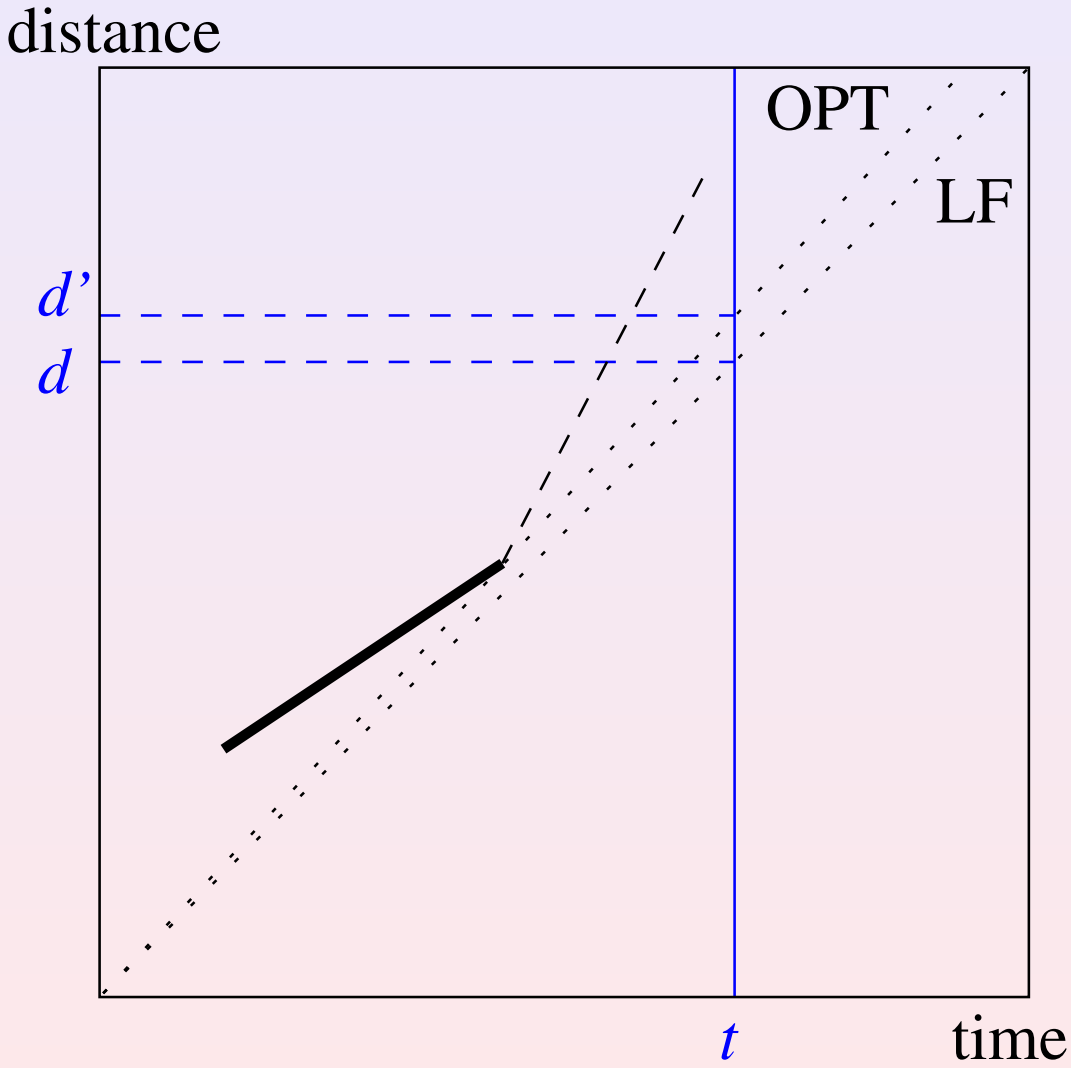
A hint about the optimality of LeapFrog



A hint about the optimality of LeapFrog



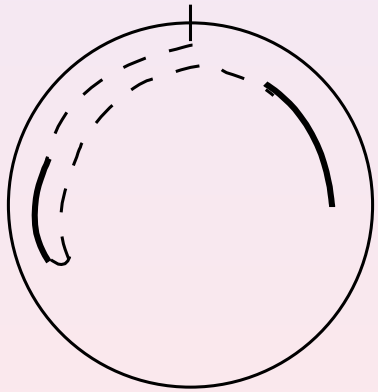
A hint about the optimality of LeapFrog



Combing a cycle of known circumference

Assumption: Initially collocated robots.

- If zig-zags were not allowed, this would be equivalent to combing a known segment from both endpoints.
- However, zig-zags are allowed:
  - either use them in order to obtain better solutions,
  - or prove that they do not help

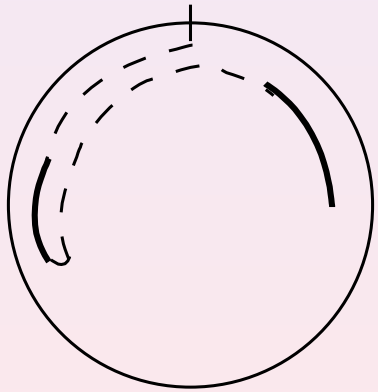


Intuitively, zig-zags shouldn't help the robots achieve optimality, but how does one prove this?

Combing a cycle of known circumference

Assumption: Initially collocated robots.

- If zig-zags were not allowed, this would be equivalent to combing a known segment from both endpoints.
- However, zig-zags are allowed:
  - either use them in order to obtain better solutions,
  - or prove that they do not help



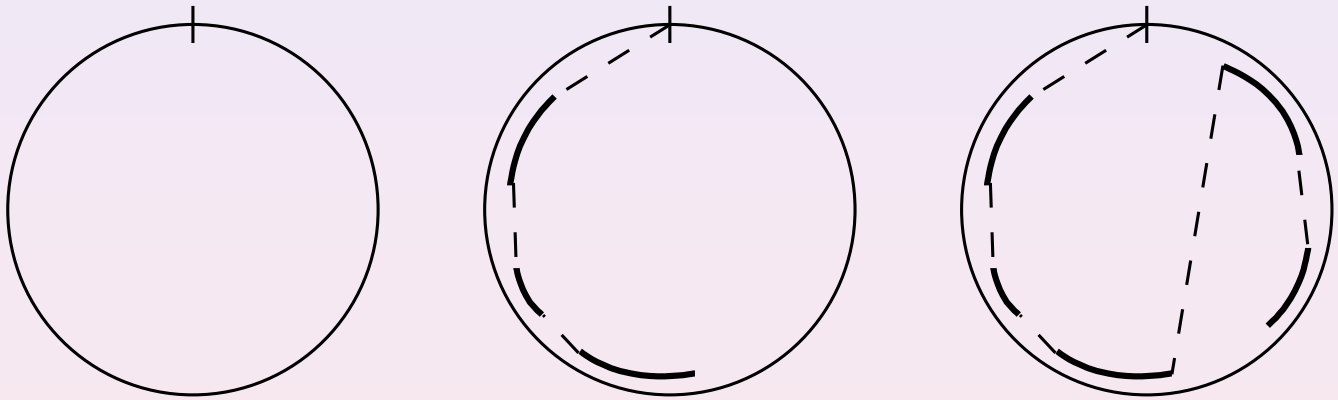
Intuitively, zig-zags shouldn't help the robots achieve optimality, but how does one prove this?

Some simple observations

- It doesn't make sense to walk at a speed smaller than w_i or to search at a speed smaller than s_i.
- It doesn't make sense to have overlapping searched arcs.
- We can assume that each trajectory looks like this:
- In fact, if we look at all robots' searched arcs:

Some simple observations

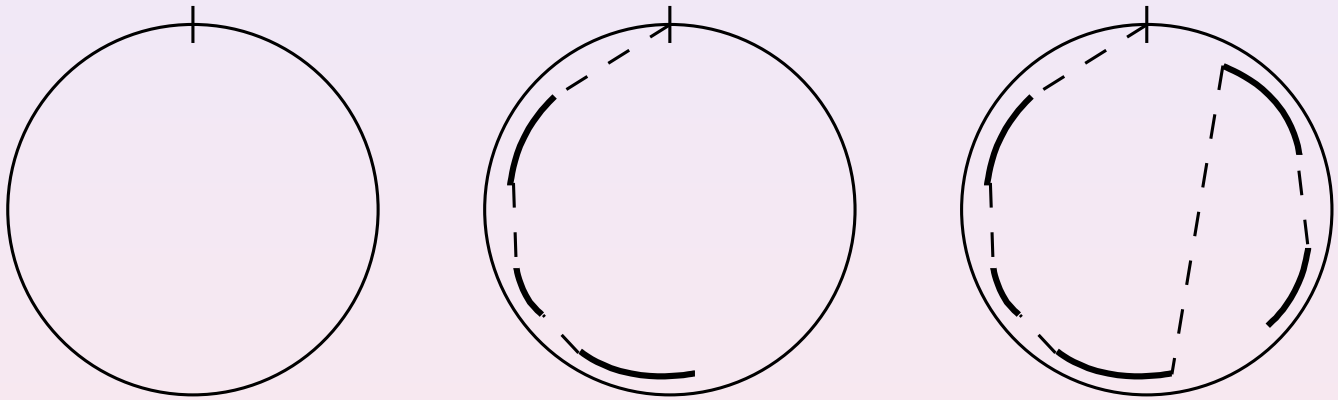
- It doesn't make sense to walk at a speed smaller than w_i or to search at a speed smaller than s_i.
- It doesn't make sense to have overlapping searched arcs.
- We can assume that each trajectory looks like this:



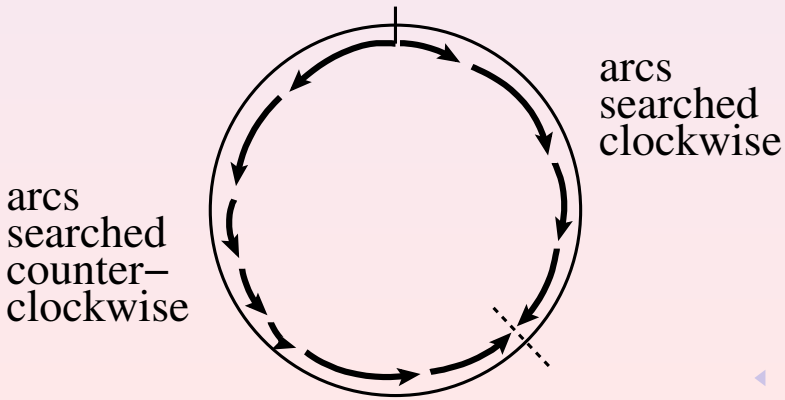
- In fact, if we look at all robots' searched arcs:

Some simple observations

- It doesn't make sense to walk at a speed smaller than w_i or to search at a speed smaller than s_i.
- It doesn't make sense to have overlapping searched arcs.
- We can assume that each trajectory looks like this:



- In fact, if we look at all robots' searched arcs:



Normal schedules

Definition

We call a schedule normal if it is as described in the previous slide.

Lemma

We can transform every non-normal schedule into a normal schedule whose completion time is not increased, and which has at least one robot whose trajectory is strictly decreased.

Lemma

In every optimal schedule, all robots terminate simultaneously.

Corollary: Every optimal schedule is normal.

Normal schedules

Definition

We call a schedule normal if it is as described in the previous slide.

Lemma

We can transform every non-normal schedule into a normal schedule whose completion time is not increased, and which has at least one robot whose trajectory is strictly decreased.

Lemma

In every optimal schedule, all robots terminate simultaneously.

Corollary: Every optimal schedule is normal.

Normal schedules

Definition

We call a schedule normal if it is as described in the previous slide.

Lemma

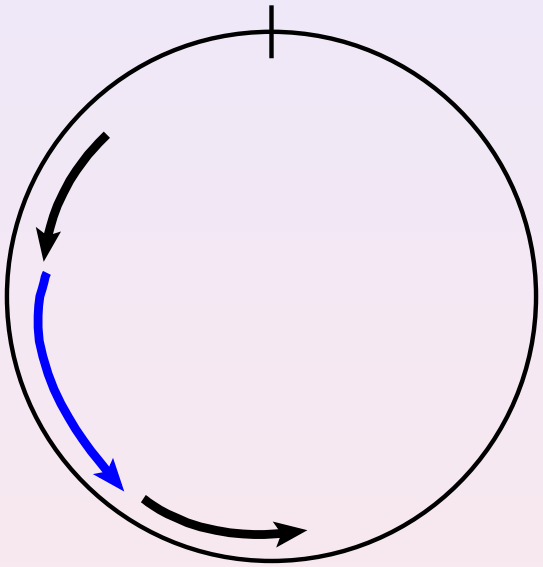
We can transform every non-normal schedule into a normal schedule whose completion time is not increased, and which has at least one robot whose trajectory is strictly decreased.

Lemma

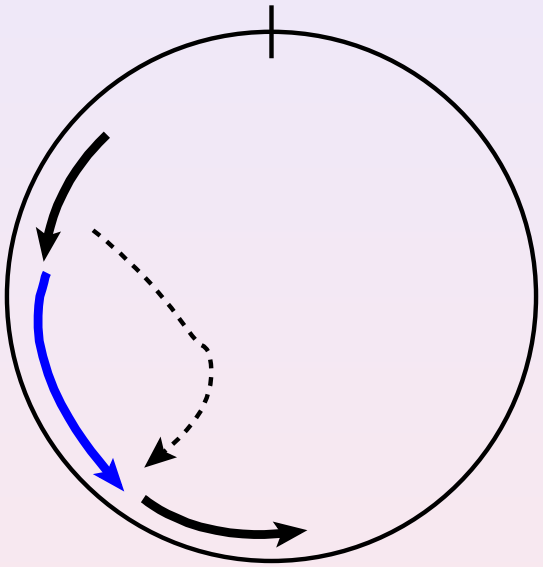
In every optimal schedule, all robots terminate simultaneously.

Corollary: Every optimal schedule is normal.

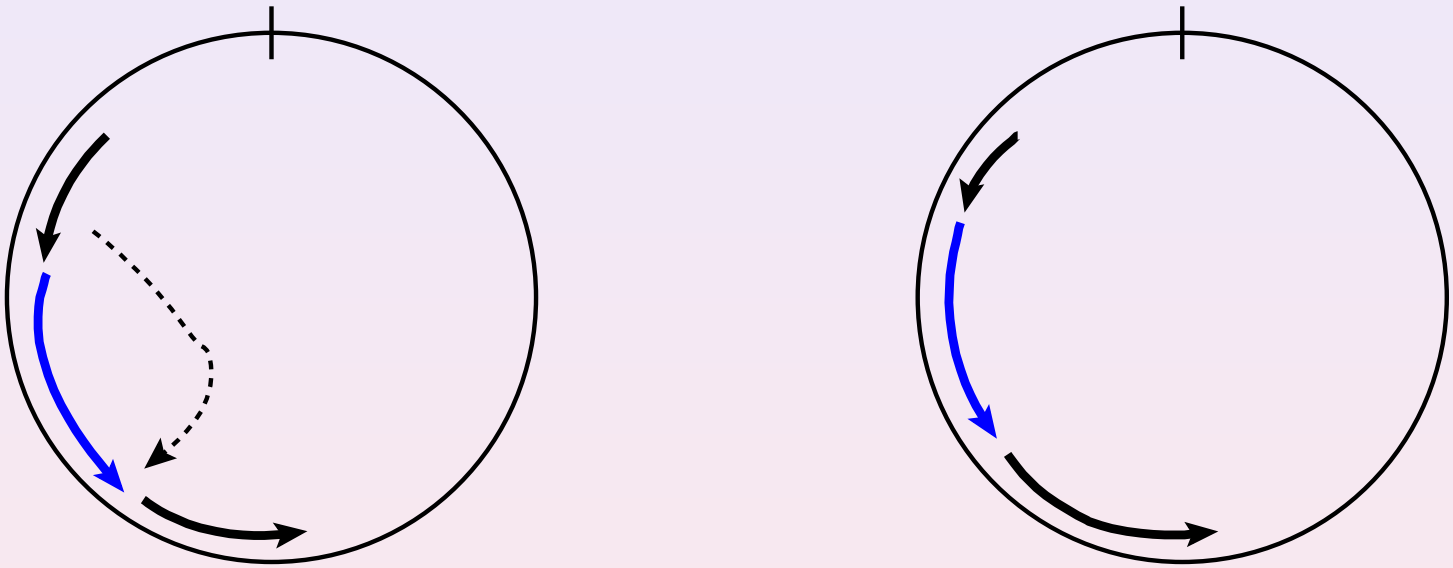
One searched arc per leg



One searched arc per leg

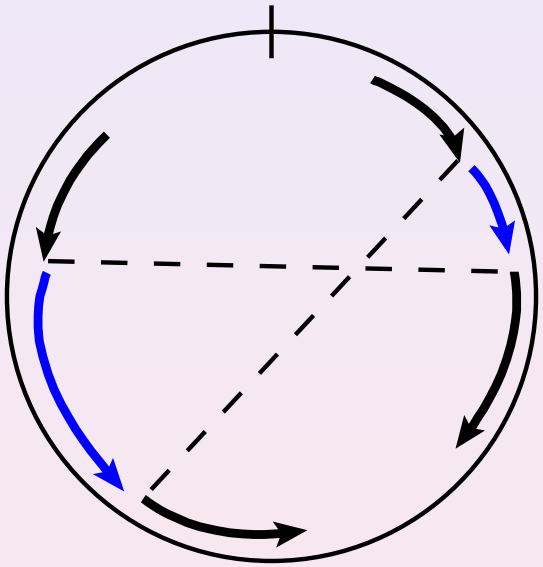


One searched arc per leg

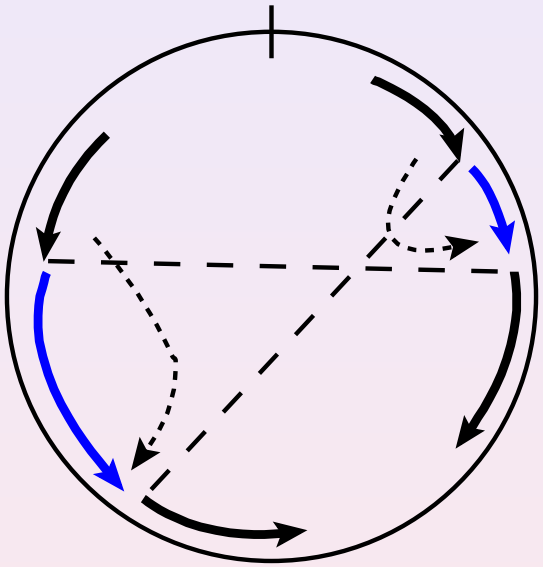


Robots searching the blue part finish their trajectories earlier.

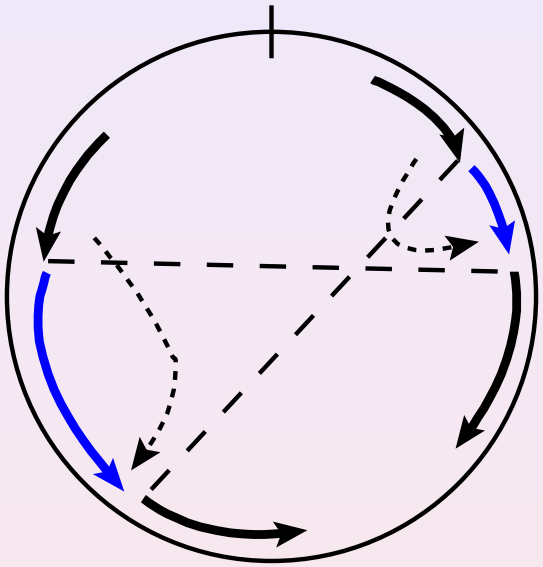
No crossing robots

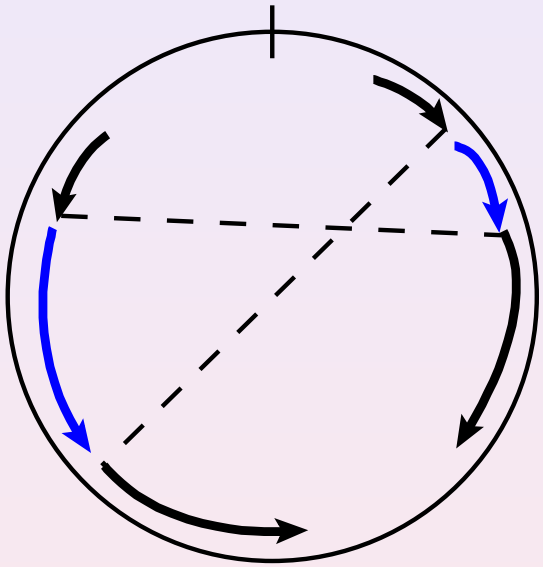


No crossing robots



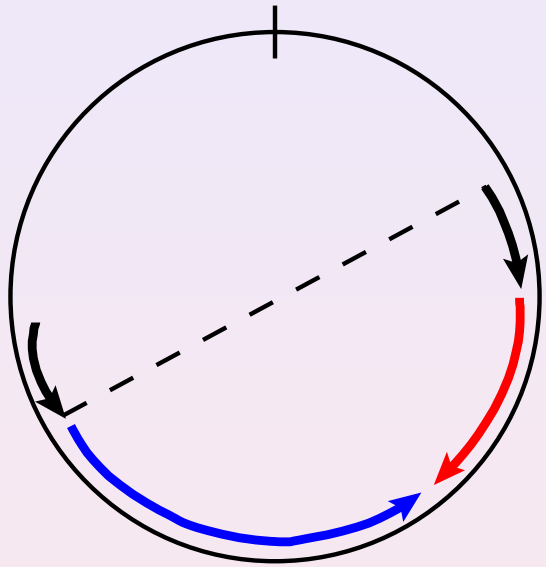
No crossing robots





All affected robots conclude their trajectories earlier.

Finally: no zig-zags



Idea: We can "nudge" the two arcs of the zig-zagging robot so that its completion time is reduced while the completion time of robots covering the blue and red arcs remain the same.

Perspectives

Many natural extensions are worth being looked at:

- Different domains
- Robots have to return to their starting position
- Online version with communication (adaptive schedule)
- Fault-tolerance
- ⋮

Thank you

for your attention