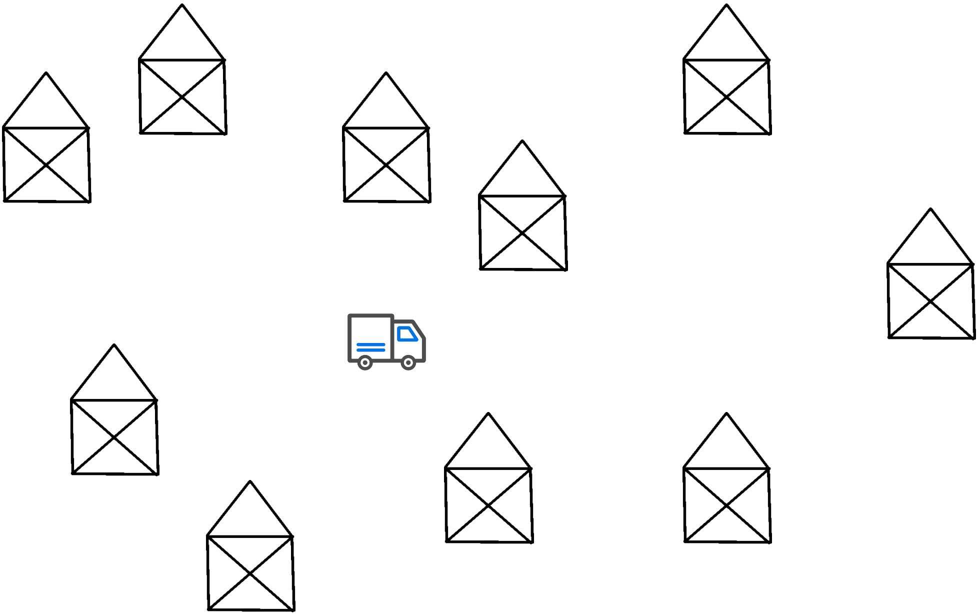
Online TSP with Known Locations

Evripidis Bampis Bruno Escoffier Niklas Hahn **Michalis Xefteris**

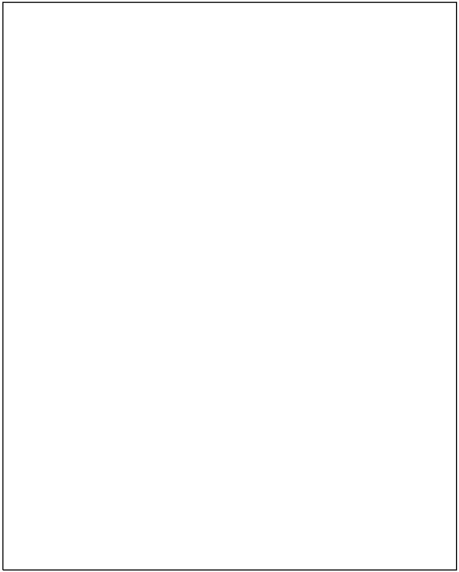
WADS 2023

31/07/2023

Traveling Salesperson Problem

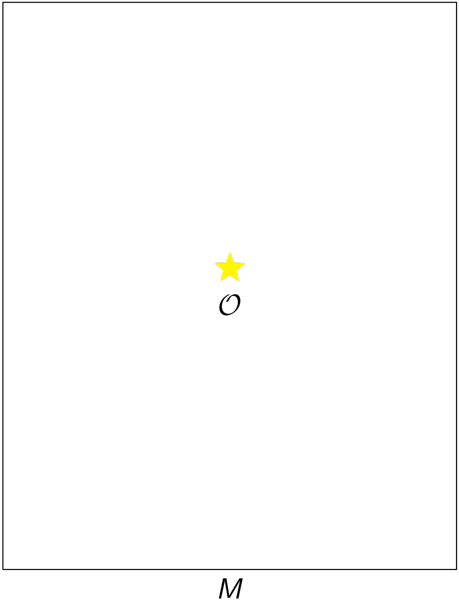


Model

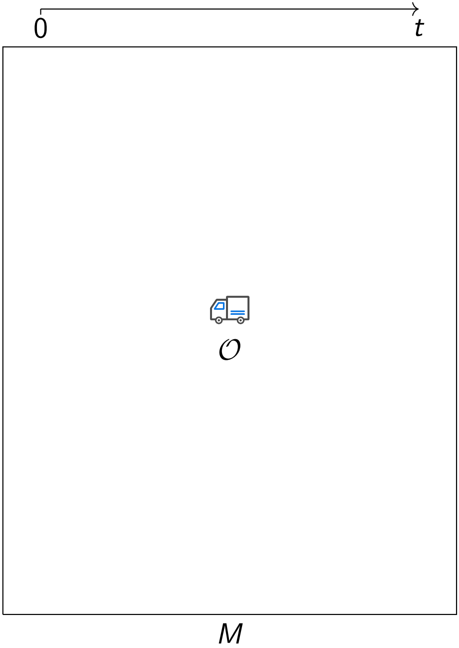


M

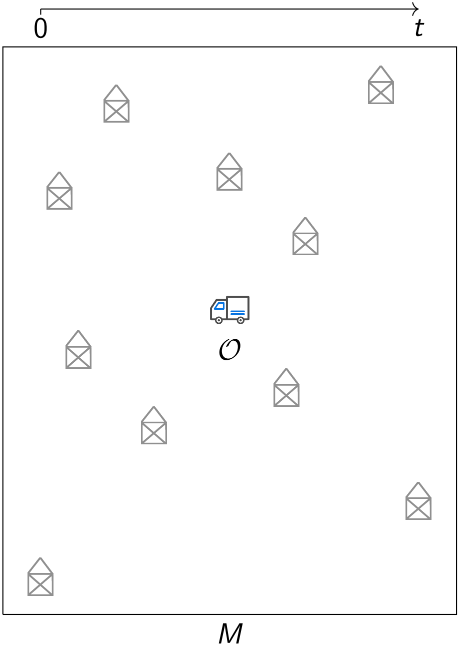
- Metric Space $M = (X, d)$



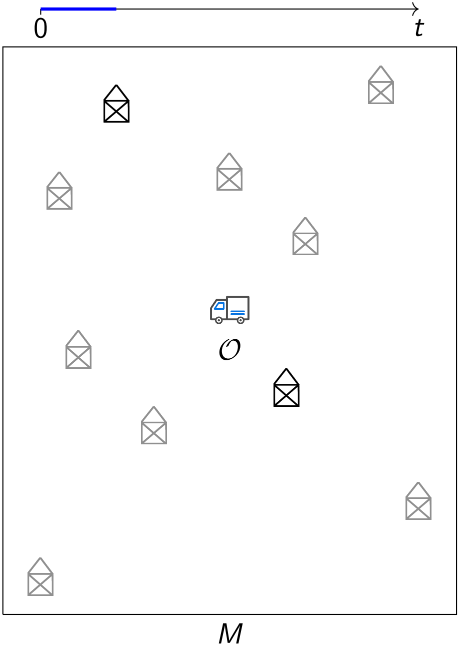
- Metric Space $M = (X, d)$
- Point $\mathcal{O} \in X$ ("Origin")



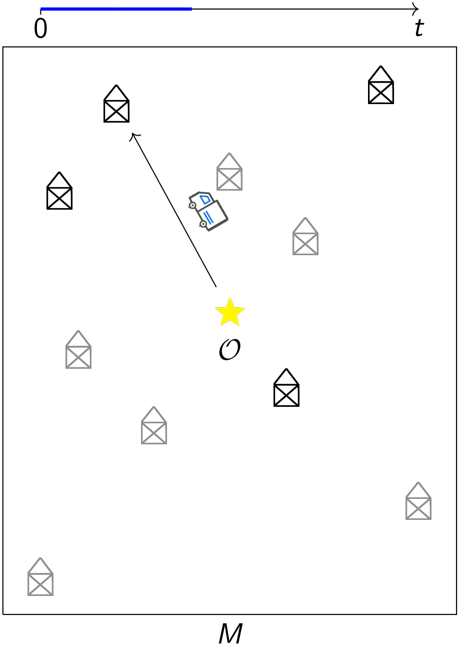
- Metric Space $M = (X, d)$
- Point $\mathcal{O} \in X$ ("Origin")
- The server $\mathcal{S}$ starts at $t = 0$ at $\mathcal{O}$ and moves with (up to) unit speed



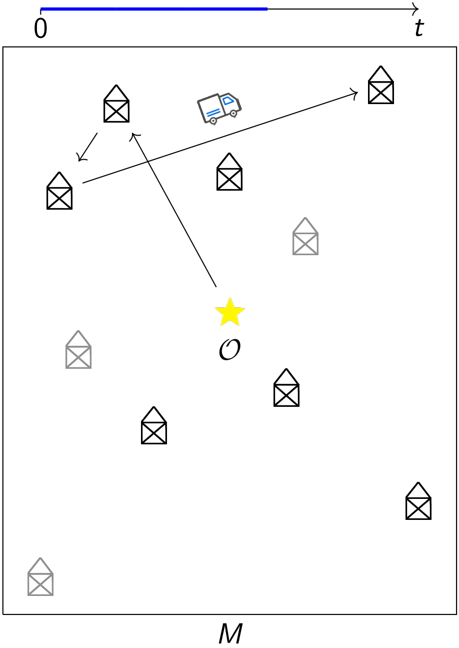
- Metric Space $M = (X, d)$
- Point $\mathcal{O} \in X$ ("Origin")
- The server $\mathcal{S}$ starts at $t = 0$ at $\mathcal{O}$ and moves with (up to) unit speed
- n requests $q_1, \ldots, q_n$
  Each q_i has position $p_i \in X$ (known) and release time $t_i \in \mathbb{R}_{\geq 0}$ (unknown)



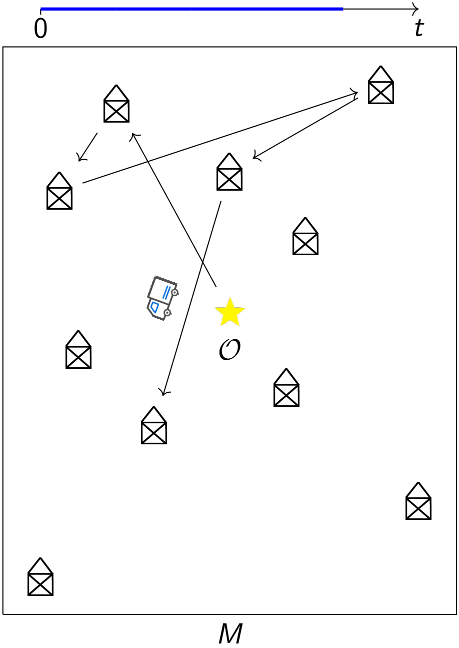
- Metric Space $M = (X, d)$
- Point $\mathcal{O} \in X$ ("Origin")
- The server $\mathcal{S}$ starts at $t = 0$ at $\mathcal{O}$ and moves with (up to) unit speed
- n requests $q_1, \ldots, q_n$
  Each q_i has position $p_i \in X$ (known) and release time $t_i \in \mathbb{R}_{\geq 0}$ (unknown)



- Metric Space $M = (X, d)$
- Point $\mathcal{O} \in X$ ("Origin")
- The server $\mathcal{S}$ starts at $t = 0$ at $\mathcal{O}$ and moves with (up to) unit speed
- n requests $q_1, \ldots, q_n$
  Each q_i has position $p_i \in X$ (known) and release time $t_i \in \mathbb{R}_{\geq 0}$ (unknown)
- $\mathcal{S}$ has to serve each q_i after t_i



- Metric Space $M = (X, d)$
- Point $\mathcal{O} \in X$ ("Origin")
- The server $\mathcal{S}$ starts at $t = 0$ at $\mathcal{O}$ and moves with (up to) unit speed
- n requests $q_1, \ldots, q_n$
  Each q_i has position $p_i \in X$ (known) and release time $t_i \in \mathbb{R}_{\geq 0}$ (unknown)
- $\mathcal{S}$ has to serve each q_i after t_i
- Objective: Minimize the completion time of the tour



- Metric Space $M = (X, d)$
- Point $\mathcal{O} \in X$ ("Origin")
- The server $\mathcal{S}$ starts at $t = 0$ at $\mathcal{O}$ and moves with (up to) unit speed
- n requests $q_1, \ldots, q_n$
  Each q_i has position $p_i \in X$ (known) and release time $t_i \in \mathbb{R}_{\geq 0}$ (unknown)
- $\mathcal{S}$ has to serve each q_i after t_i
- Objective: Minimize the completion time of the tour
- Open variant: $\mathcal{S}$ finishes when reaching the final request
  Closed variant: $\mathcal{S}$ has to return to $\mathcal{O}$

Competitive Ratio

An **online** algorithm ALG has competitive ratio ϱ if $|\text{ALG}(\sigma)| \leq \varrho \cdot |\text{OPT}(\sigma)|$ for any input instance σ, where OPT is an optimal **offline** algorithm.

Competitive Ratio

An **online** algorithm ALG has competitive ratio ϱ if $|\mathsf{ALG}(\sigma)| \leq \varrho \cdot |\mathsf{OPT}(\sigma)|$ for any input instance σ, where OPT is an optimal **offline** algorithm.

Simple 2-competitive algorithm

1. Wait at $\mathcal{O}$ until all requests are released
2. Follow an optimal tour

Competitive Ratio

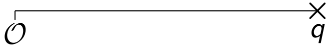
An **online** algorithm ALG has competitive ratio ϱ if $|\text{ALG}(\sigma)| \leq \varrho \cdot |\text{OPT}(\sigma)|$ for any input instance σ, where OPT is an optimal **offline** algorithm.

Simple 2-competitive algorithm

1. Wait at $\mathcal{O}$ until all requests are released
2. Follow an optimal tour

Example

Open variant, single request q with distance 1 and release time 1



Optimal offline algorithm finishes at $t = 1$, $|\text{ALG}| = 2$

Competitive Ratio

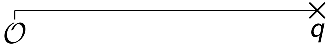
An **online** algorithm ALG has competitive ratio ϱ if $|\text{ALG}(\sigma)| \leq \varrho \cdot |\text{OPT}(\sigma)|$ for any input instance σ, where OPT is an optimal **offline** algorithm.

Simple 2-competitive algorithm

1. Wait at $\mathcal{O}$ until all requests are released
2. Follow an optimal tour

Example

Open variant, single request q with distance 1 and release time 1



Optimal offline algorithm finishes at $t = 1$, $|\text{ALG}| = 2$

Classic Online TSP

Tight bound of 2 for the general case in the closed variant [Ausiello et al. '01]

Tight bound of 2.04 for the open variant on the line, 2.5 general upper bound [Bjelde et al. '20, Ausiello et al. '01]

Observation

W.l.o.g OPT follows a tour/path waiting only at requests' positions and moving from a request A to a request B in time $d(A, B)$.

Observation

W.l.o.g OPT follows a tour/path waiting only at requests' positions and moving from a request A to a request B in time $d(A, B)$.

Main idea:

- Wait in $\mathcal{O}$ until the time that OPT may have traversed exactly half of its tour

Observation

W.l.o.g OPT follows a tour/path waiting only at requests' positions and moving from a request A to a request B in time $d(A, B)$.

Main idea:

- Wait in $\mathcal{O}$ until the time that OPT may have traversed exactly half of its tour
- Follow a "good" tour without deviating and wait at any unreleased request

Algorithm for General Metrics (1)

Observation

W.l.o.g OPT follows a tour/path waiting only at requests' positions and moving from a request A to a request B in time $d(A, B)$.

Main idea:

- Wait in $\mathcal{O}$ until the time that OPT may have traversed exactly half of its tour
- Follow a "good" tour without deviating and wait at any unreleased request

Theorem

The algorithm for general metrics achieves a competitive ratio of $3/2$ in both the open and the closed variant of online TSP with known locations.

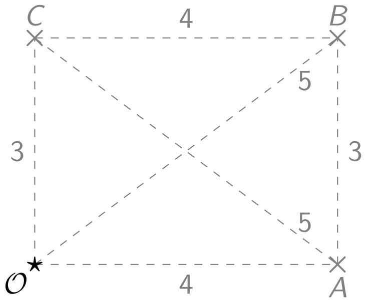
Main idea:

- Wait in $\mathcal{O}$ until time t s.t. two conditions are satisfied for a single tour σ:
  1. The first half of σ (w.r.t. its length ℓ_σ) can be traversed without encountering an unreleased request

Main idea:

- Wait in $\mathcal{O}$ until time t s.t. two conditions are satisfied for a single tour σ:
  1. The first half of σ (w.r.t. its length ℓ_σ) can be traversed without encountering an unreleased request

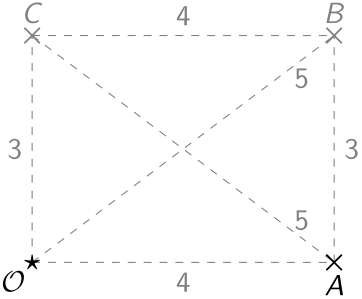
Example (closed variant):



Main idea:

- Wait in $\mathcal{O}$ until time t s.t. two conditions are satisfied for a single tour σ:
  1. The first half of σ (w.r.t. its length ℓ_σ) can be traversed without encountering an unreleased request

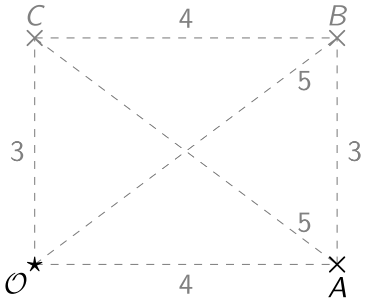
Example (closed variant):

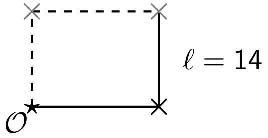


Main idea:

- Wait in $\mathcal{O}$ until time t s.t. two conditions are satisfied for a single tour σ:
  1. The first half of σ (w.r.t. its length ℓ_σ) can be traversed without encountering an unreleased request

Example (closed variant):

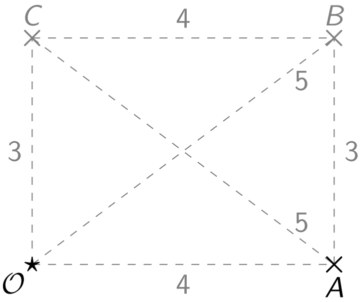


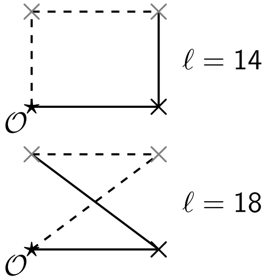


Main idea:

- Wait in $\mathcal{O}$ until time t s.t. two conditions are satisfied for a single tour σ:
  1. The first half of σ (w.r.t. its length ℓ_σ) can be traversed without encountering an unreleased request

Example (closed variant):

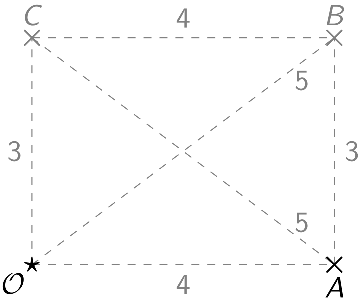


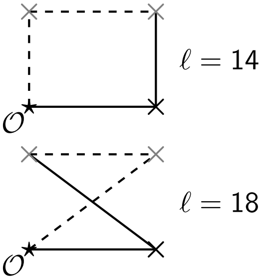


Main idea:

- Wait in $\mathcal{O}$ until time t s.t. two conditions are satisfied for a single tour σ:
  1. The first half of σ (w.r.t. its length ℓ_σ) can be traversed without encountering an unreleased request
  2. The tour has length at most $\ell_\sigma \leq 2t$

Example (closed variant):

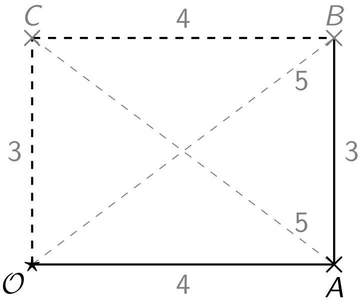


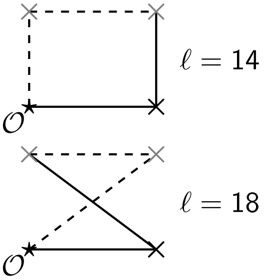


Main idea:

- Wait in $\mathcal{O}$ until time t s.t. two conditions are satisfied for a single tour σ:
  1. The first half of σ (w.r.t. its length ℓ_σ) can be traversed without encountering an unreleased request
  2. The tour has length at most $\ell_\sigma \leq 2t$
- Choose a tour σ' that minimizes $\max\{\ell_{\sigma'}/2, \text{unreleased length}(\sigma')\}$ and follow it without deviating, waiting at any unreleased request

Example (closed variant):



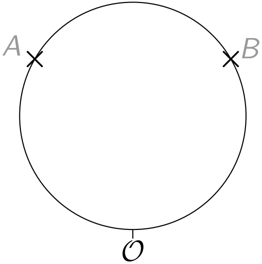


Open Variant

Competitive ratio at least $3/2$:

Open Variant

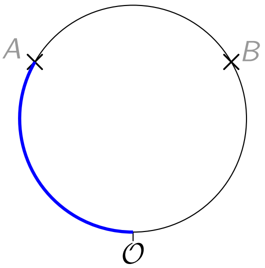
Competitive ratio at least 3/2:



- Circumference 3, 2 requests, all distances 1

Open Variant

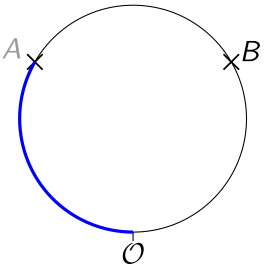
Competitive ratio at least 3/2:



- Circumference 3, 2 requests, all distances 1
- At $t = 1$, w.l.o.g. online $\mathcal{S}$ is between A and $\mathcal{O}$ (blue segment)

Open Variant

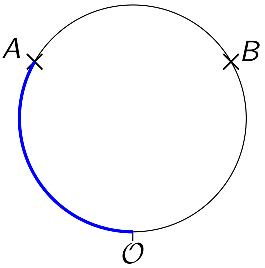
Competitive ratio at least 3/2:



- Circumference 3, 2 requests, all distances 1
- At $t = 1$, w.l.o.g. online $\mathcal{S}$ is between A and $\mathcal{O}$ (blue segment)
- B is released at $t = 1$

Open Variant

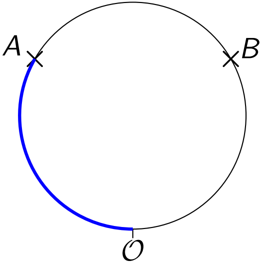
Competitive ratio at least 3/2:



- Circumference 3, 2 requests, all distances 1
- At $t = 1$, w.l.o.g. online $\mathcal{S}$ is between A and $\mathcal{O}$ (blue segment)
- B is released at $t = 1$
- A is released at $t = 2$

Open Variant

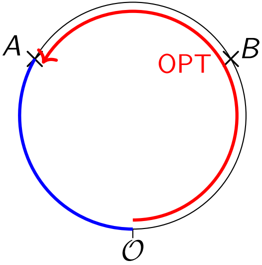
Competitive ratio at least 3/2:



- Circumference 3, 2 requests, all distances 1
- At $t = 1$, w.l.o.g. online $\mathcal{S}$ is between A and $\mathcal{O}$ (blue segment)
- B is released at $t = 1$
- A is released at $t = 2$
- $\mathcal{S}$ cannot serve the first request before $t = 2$

Open Variant

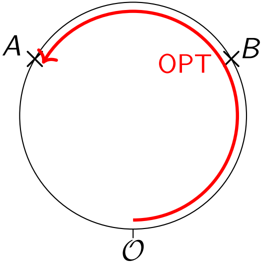
Competitive ratio at least $3/2$:



- Circumference 3, 2 requests, all distances 1
- At $t = 1$, w.l.o.g. online $\mathcal{S}$ is between A and $\mathcal{O}$ (blue segment)
- B is released at $t = 1$
- A is released at $t = 2$
- $\mathcal{S}$ cannot serve the first request before $t = 2$
- OPT finishes the tour by $t = 2$, no online algorithm can finish before $t = 3$

Open Variant

Competitive ratio at least $3/2$:



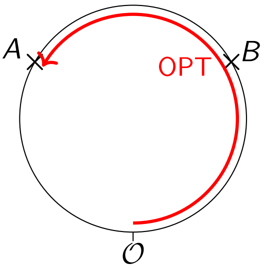
- Circumference 3, 2 requests, all distances 1
- At $t = 1$, w.l.o.g. online $\mathcal{S}$ is between A and $\mathcal{O}$ (blue segment)
- B is released at $t = 1$
- A is released at $t = 2$
- $\mathcal{S}$ cannot serve the first request before $t = 2$
- OPT finishes the tour by $t = 2$, no online algorithm can finish before $t = 3$

Closed Variant

An example on the line suffices for a lower bound of $3/2$ [Gouleakis et al. '23]

Open Variant

Competitive ratio at least $3/2$:



- Circumference 3, 2 requests, all distances 1
- At $t = 1$, w.l.o.g. online $\mathcal{S}$ is between A and $\mathcal{O}$ (blue segment)
- B is released at $t = 1$
- A is released at $t = 2$
- $\mathcal{S}$ cannot serve the first request before $t = 2$
- OPT finishes the tour by $t = 2$, no online algorithm can finish before $t = 3$

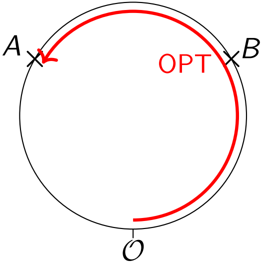
Closed Variant

An example on the line suffices for a lower bound of $3/2$ [Gouleakis et al. '23]

$\rightsquigarrow$ The general algorithm is optimal

Open Variant

Competitive ratio at least 3/2:



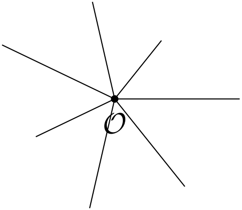
- Circumference 3, 2 requests, all distances 1
- At $t = 1$, w.l.o.g. online $\mathcal{S}$ is between A and $\mathcal{O}$ (blue segment)
- B is released at $t = 1$
- A is released at $t = 2$
- $\mathcal{S}$ cannot serve the first request before $t = 2$
- OPT finishes the tour by $t = 2$, no online algorithm can finish before $t = 3$

Closed Variant

An example on the line suffices for a lower bound of 3/2 [Gouleakis et al. '23]

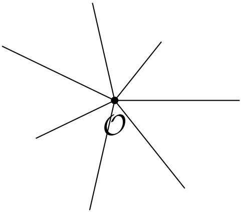
$\rightsquigarrow$ The general algorithm is optimal, **but** it is not poly-time

Stars:



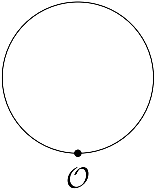
Poly-time algorithm which is $(7/4 + \varepsilon)$ competitive (closed).

Stars:



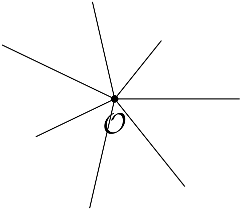
Poly-time algorithm which is $(7/4 + \varepsilon)$ competitive (closed).

Rings:



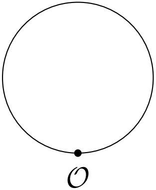
Poly-time algorithm which is $5/3$ competitive (closed).

Stars:



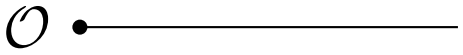
Poly-time algorithm which is $(7/4 + \varepsilon)$ competitive (closed).

Rings:



Poly-time algorithm which is $5/3$ competitive (closed).

Semi-line:



Poly-time algorithms which are 1 (closed) and $13/9$ competitive (open).

Main idea:

Main idea:

1. **If** a long ray (i.e., ray of length at least $1/4$ of the overall length of all rays) exists, first serve this ray completely and return to $\mathcal{O}$.

Main idea:

1. **If** a long ray (i.e., ray of length at least $1/4$ of the overall length of all rays) exists, first serve this ray completely and return to $\mathcal{O}$.

   Otherwise (all rays are short), wait in $\mathcal{O}$ until time t, which is exactly the combined length of all rays. Then,

Main idea:

1. **If** a long ray (i.e., ray of length at least $1/4$ of the overall length of all rays) exists, first serve this ray completely and return to $\mathcal{O}$.

   Otherwise (all rays are short), wait in $\mathcal{O}$ until time t, which is exactly the combined length of all rays. Then,

   - Identify a set R of rays maximizing the released segments in R under the constraint that the set R can be traversed completely, including going back to $\mathcal{O}$, in time t.

Main idea:

1. **If** a long ray (i.e., ray of length at least $1/4$ of the overall length of all rays) exists, first serve this ray completely and return to $\mathcal{O}$.

   Otherwise (all rays are short), wait in $\mathcal{O}$ until time t, which is exactly the combined length of all rays. Then,

   - Identify a set R of rays maximizing the released segments in R under the constraint that the set R can be traversed completely, including going back to $\mathcal{O}$, in time t.
   - Only contiguous segments starting at the outer extremities of the rays are counted.

Main idea:

1. **If** a long ray (i.e., ray of length at least $1/4$ of the overall length of all rays) exists, first serve this ray completely and return to $\mathcal{O}$.

   Otherwise (all rays are short), wait in $\mathcal{O}$ until time t, which is exactly the combined length of all rays. Then,

   - Identify a set R of rays maximizing the released segments in R under the constraint that the set R can be traversed completely, including going back to $\mathcal{O}$, in time t.
   - Only contiguous segments starting at the outer extremities of the rays are counted.
   - Then, serve the requests in R and return to $\mathcal{O}$.

Main idea:

1. **If** a long ray (i.e., ray of length at least $1/4$ of the overall length of all rays) exists, first serve this ray completely and return to $\mathcal{O}$.

   Otherwise (all rays are short), wait in $\mathcal{O}$ until time t, which is exactly the combined length of all rays. Then,

   - Identify a set R of rays maximizing the released segments in R under the constraint that the set R can be traversed completely, including going back to $\mathcal{O}$, in time t.
   - Only contiguous segments starting at the outer extremities of the rays are counted.
   - Then, serve the requests in R and return to $\mathcal{O}$.

2. Wait in $\mathcal{O}$ until all requests are released. Afterwards, serve the unserved requests in an optimal manner.

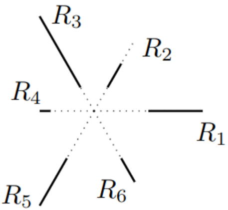
- The algorithm achieves a competitive ratio of $7/4$ with an optimal set R

Stars (2)

- The algorithm achieves a competitive ratio of $7/4$ with an optimal set R
- Finding an optimal set R constitutes solving a knapsack problem
  $\rightsquigarrow$ FPTAS to find R gives us
  for every $\varepsilon > 0$, a $(7/4 + \varepsilon)$ competitive poly-time algorithm

- The algorithm achieves a competitive ratio of $7/4$ with an optimal set R
- Finding an optimal set R constitutes solving a knapsack problem
  $\rightsquigarrow$ FPTAS to find R gives us
  for every $\varepsilon > 0$, a $(7/4 + \varepsilon)$ competitive poly-time algorithm

Example

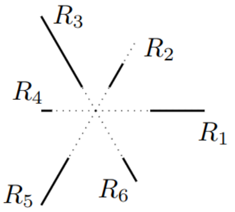


Ray	R_1	R_2	R_3	R_4	R_5	R_6
Length	0.2	0.15	0.2	0.1	0.2	0.15
Released	0.1	0	0.15	0.02	0.1	0.05

Stars (2)

- The algorithm achieves a competitive ratio of $7/4$ with an optimal set R
- Finding an optimal set R constitutes solving a knapsack problem
  $\rightsquigarrow$ FPTAS to find R gives us
  for every $\varepsilon > 0$, a $(7/4 + \varepsilon)$ competitive poly-time algorithm

Example



Ray	R_1	R_2	R_3	R_4	R_5	R_6
Length	0.2	0.15	0.2	0.1	0.2	0.15
Released	0.1	0	0.15	0.02	0.1	0.05

$\rightsquigarrow$ R : rays R_1, R_3 and R_4 (or R_3, R_4, R_5) with a combined length of $1/2$ and released length of $\ell = 0.27$

	Open OLTSP-L		Closed OLTSP-L		
	Lower Bound	Upper Bound	Lower Bound	Upper Bound	
Semi-line	4/3	13/9*	**1**	**1***	
Star	13/9	3/2	**3/2**	**3/2**	$(7/4+\varepsilon)$*
Ring	**3/2**	**3/2**	**3/2**	**3/2**	5/3*
General	**3/2**	**3/2**	**3/2**	**3/2**	

Poly-time algorithms denoted by *

Outlook

	Open OLTSP-L		Closed OLTSP-L	
	Lower Bound	Upper Bound	Lower Bound	Upper Bound
Semi-line	4/3	13/9*	**1**	**1***
Star	13/9	3/2	**3/2**	**3/2** (7/4+ε)*
Ring	**3/2**	**3/2**	**3/2**	**3/2** 5/3*
General	**3/2**	**3/2**	**3/2**	**3/2**

Poly-time algorithms denoted by *

Open Questions

- Improving running time (general)
- Improving the Bounds
- Predictions on locations

	Open OLTSP-L		Closed OLTSP-L	
	Lower Bound	Upper Bound	Lower Bound	Upper Bound
Semi-line	4/3	13/9*	**1**	**1***
Star	13/9	3/2	**3/2**	**3/2** (7/4+ε)*
Ring	**3/2**	**3/2**	**3/2**	**3/2** 5/3*
General	**3/2**	**3/2**	**3/2**	**3/2**

Poly-time algorithms denoted by *

Open Questions

- Improving running time (general)
- Improving the Bounds
- Predictions on locations

⇝ Learning-Augmented Online TSP on Rings, Trees, Flowers and (almost) Everywhere Else

Outlook

	Open OLTSP-L		Closed OLTSP-L	
	Lower Bound	Upper Bound	Lower Bound	Upper Bound
Semi-line	4/3	13/9*	**1**	**1***
Star	13/9	3/2	**3/2**	**3/2** $(7/4+\varepsilon)$*
Ring	**3/2**	**3/2**	**3/2**	**3/2** 5/3*
General	**3/2**	**3/2**	**3/2**	**3/2**

Poly-time algorithms denoted by *

Open Questions

- Improving running time (general)
- Improving the Bounds
- Predictions on locations

$\rightsquigarrow$ Learning-Augmented Online TSP on Rings, Trees, Flowers and (almost) Everywhere Else

Thank you!