Multirobot Coverage of Linear Modular Environments

Honours Programme
Scientific Research in Information Technology

December 13, 2019

Mirko Salaris: mirko.salaris@mail.polimi.it

POLITECNICO
MILANO 1863

HONOURS PROGRAMME HP-SR
in Information Technology

What is Coverage?

- a known environment
- a set of points of interest
- a mobile robot, with a ‘covering tool’ of finite size

Goal:

- optimal tour
- covers all the points

[Y. Huang, 1986]

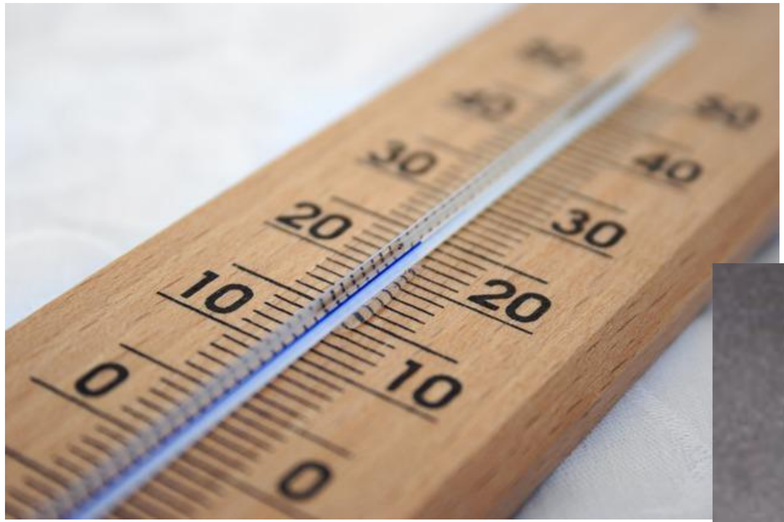
Applications

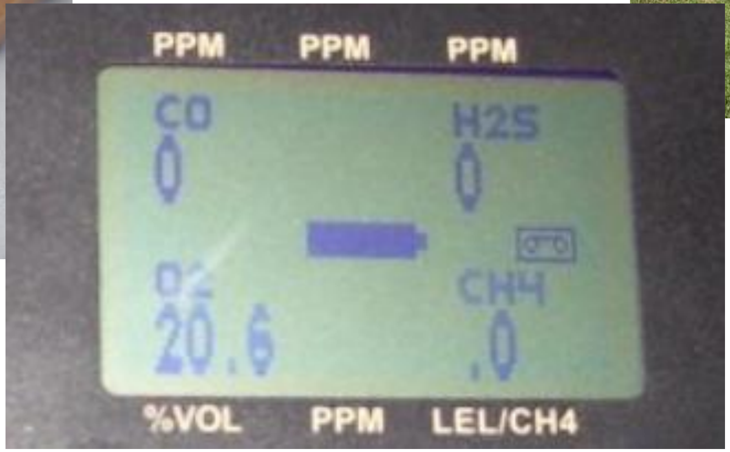
- to physically pass over a specified set of points

[E Galceran, M Carreras, 2013]

Applications

- to physically pass over a specified set of points
- to gather data about the environment





[E Galceran, M Carreras, 2013]

Applications

- to physically pass over a specified set of points
- to gather data about the environment
- for search and rescue applications

[E Galceran, M Carreras, 2013]

Multirobot Coverage - Motivation

Advantages:

- it provides robustness (i.e., supporting the loss of a robot)
- it increases efficiency

Drawbacks:

- coordination issues
- increased algorithmic complexity

[N. Karapetyan et al., 2017]

Multirobot Coverage - Definition

- a known environment
- a set of points of interest
- **multiple** mobile robots, with a ‘covering tool’ of finite size

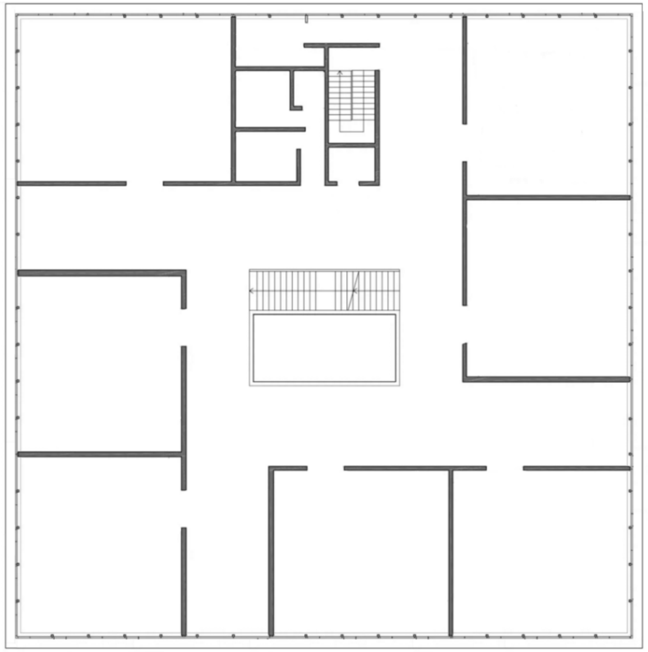
Goal:

- **optimal set** of tours
- coverage of all the points

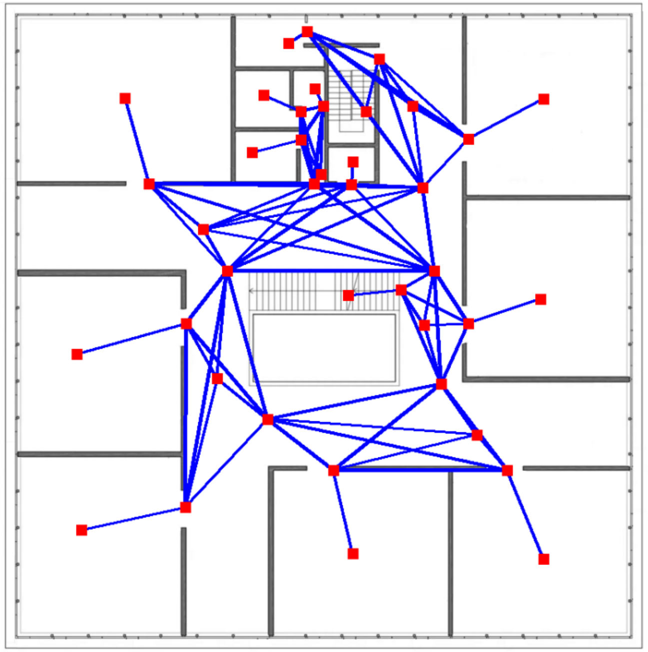
Common metrics:
MINSUM, MINMAX

Preliminaries - Environment representation

- points of interest of the environment → vertices
- connections between points → edges



→



Preliminaries - TSP

Traveling Salesperson Problem (TSP):

"Given a set of cities and the distances between each pair of them, what is the shortest possible route that visits each city and returns to the origin city?"

Cities → vertices

Distances → edges with associated cost

Origin city → depot

[D. Applegate et al., 2006]

Multirobot coverage as mTSP

Multiple Traveling Salesperson Problem (mTSP):

“Given a set of vertices and a cost metric defined in terms of distance or time, let there be *m* robots located at a single initial vertex, called depot. The remaining vertices are called ‘intermediate vertices’. The mTSP consists of finding tours for all the *m* robots, which all start and end at the depot, such that each intermediate vertex is visited exactly once and the total cost of visiting all the vertices is minimized”

[Bektas, T., 2006]

Multirobot coverage as mTSP

Multiple Traveling Salesperson Problem (mTSP):

“Given **a set of vertices** and **a cost metric** defined in terms of distance or time, let there be ***m* robots** located at a single initial vertex, called **depot**. The remaining vertices are called ‘intermediate vertices’. The mTSP consists of **finding tours** for all the *m* robots, which all start and end at the depot, such that **each** intermediate **vertex is visited exactly once** and **the total cost** of visiting all the vertices **is minimized**”

[Bektas, T., 2006]

Common metrics:

MINSUM, MINMAX

Solving the mTSP

NP-Hard!

↓

Approximation algorithms

Solving the mTSP

Intertwined issues of the mTSP:

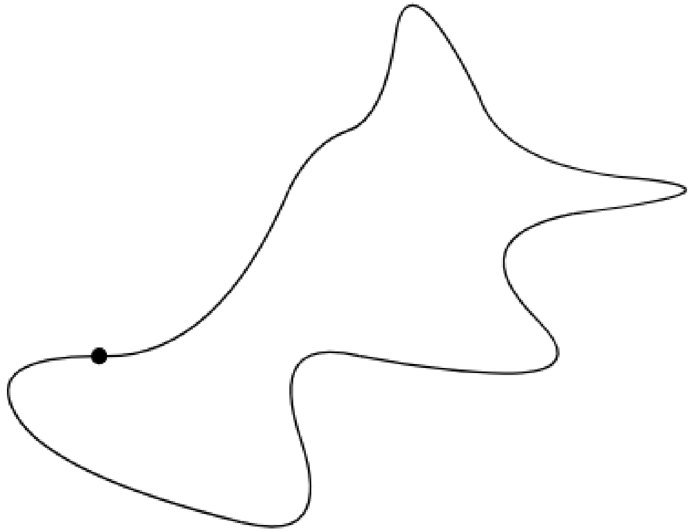
- partitioning of the vertices
- computation of the tours

Splitting the TSP

Always possible:

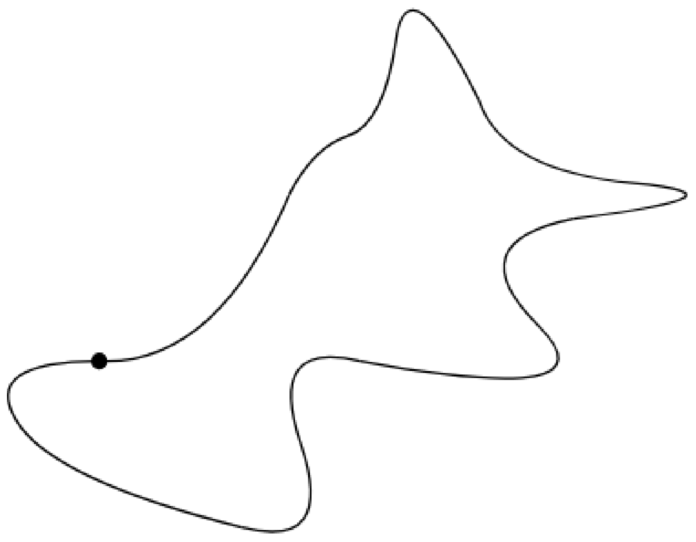
- create *m* copies of the original depot
- solve the TSP on this new graph
- split the obtained solution in the copies of the depot

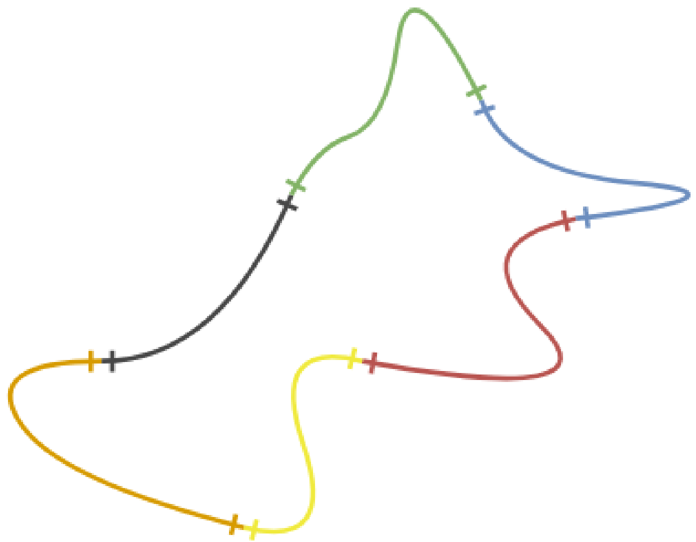
Splitting the TSP



6 robots

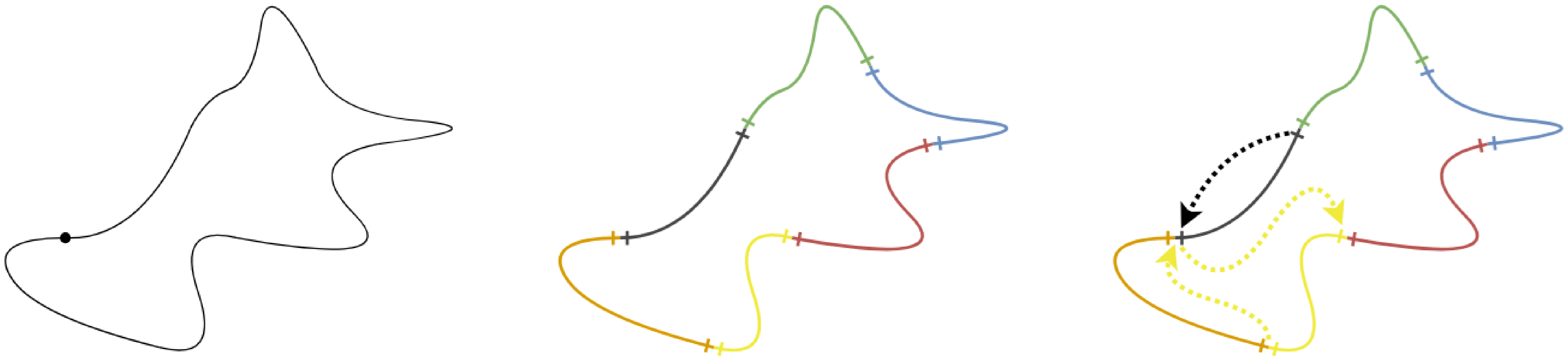
Splitting the TSP





6 robots

Splitting the TSP



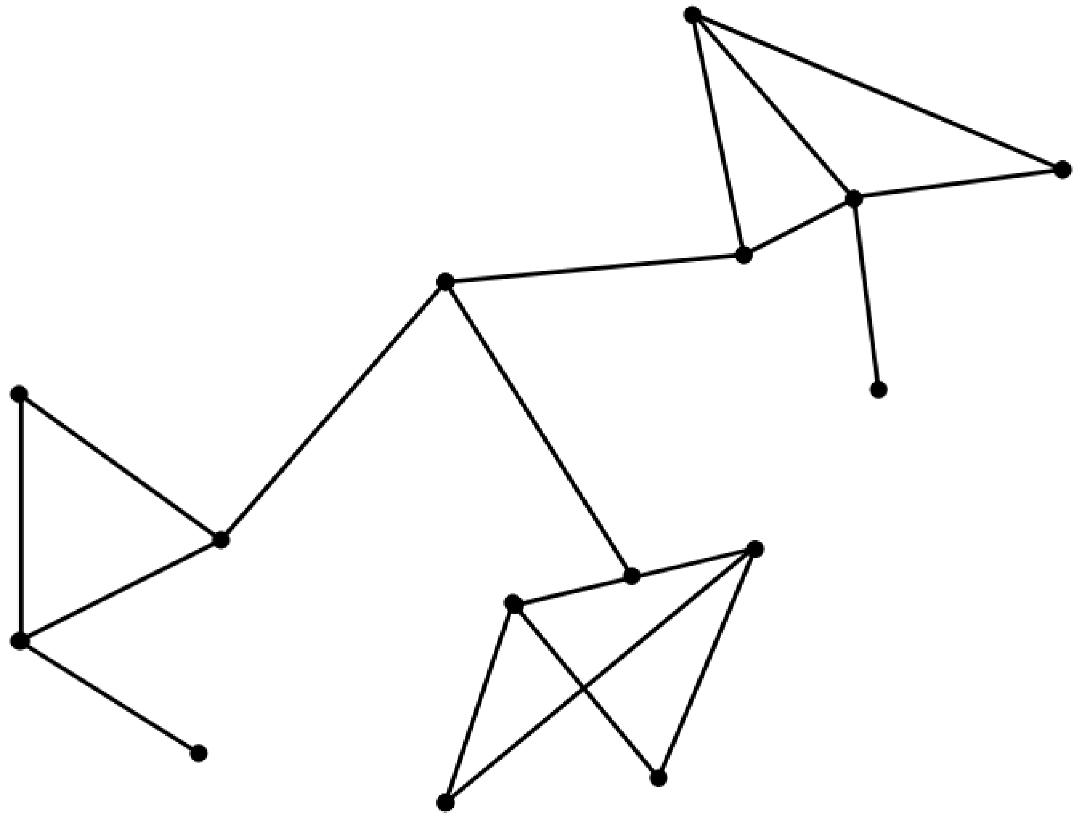
6 robots

Splitting the TSP

Frederickson et al. (1976): tour-splitting heuristic

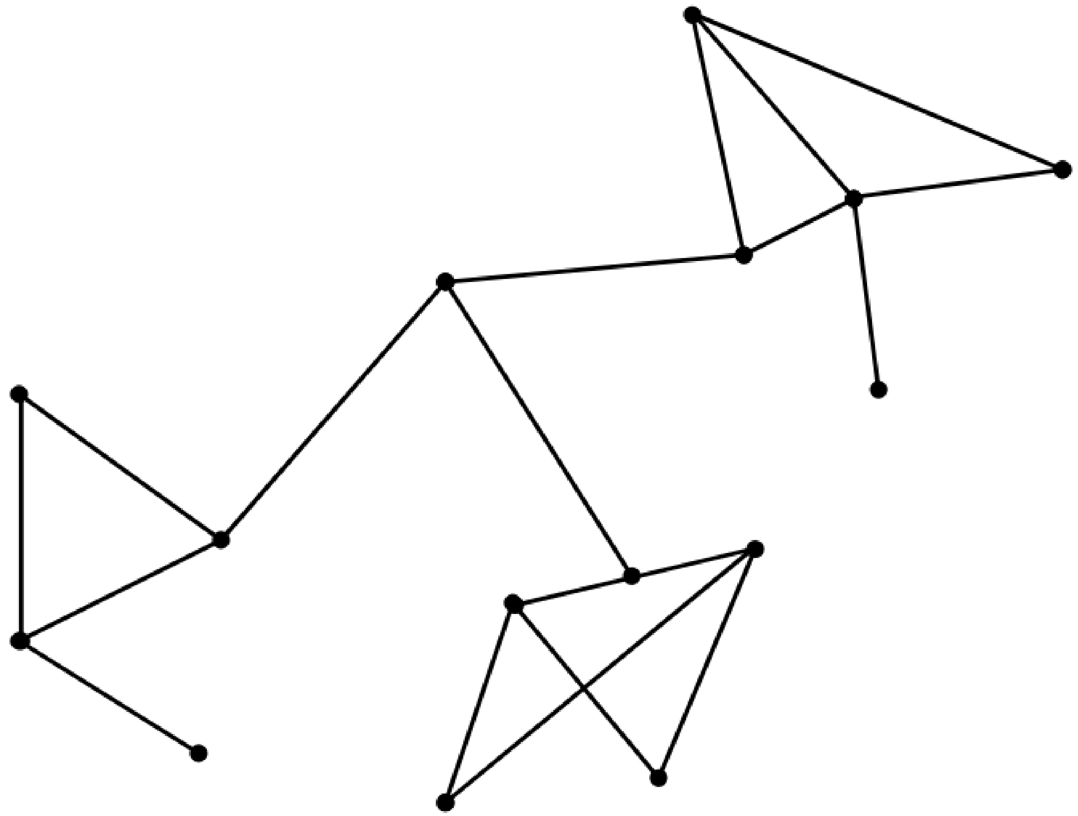
Approximation factor: $\frac{5}{2} - \frac{1}{m}$

Partitioning vertices - Clustering

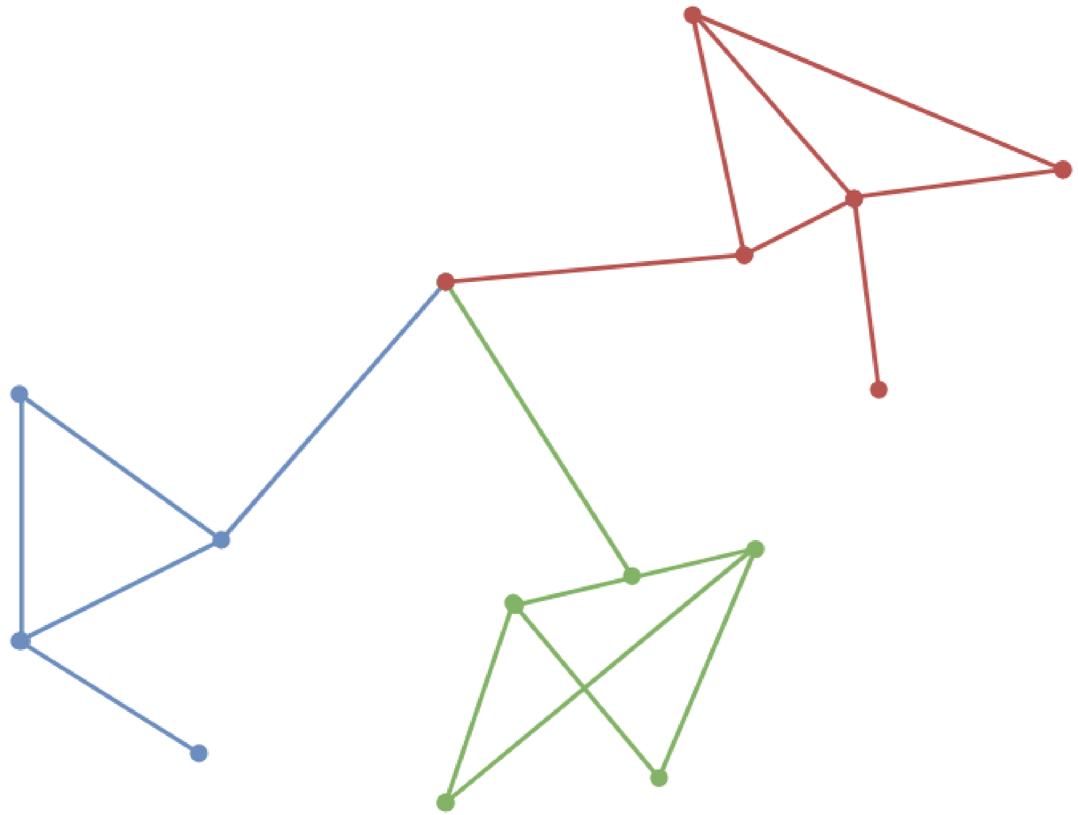


3 robots

Partitioning vertices - Clustering



3 robots



mTSP approximation bounds

Frederickson, 1976: $(\frac{5}{2} - \frac{1}{m})$ → Best theoretical guarantee in the state-of-the-art

Tighter approximation bounds

Tailored algorithms for **constrained environments**

Example

mTSP on trees with multiple depots: $(2 - \frac{2}{m+1})$ approximate algorithm

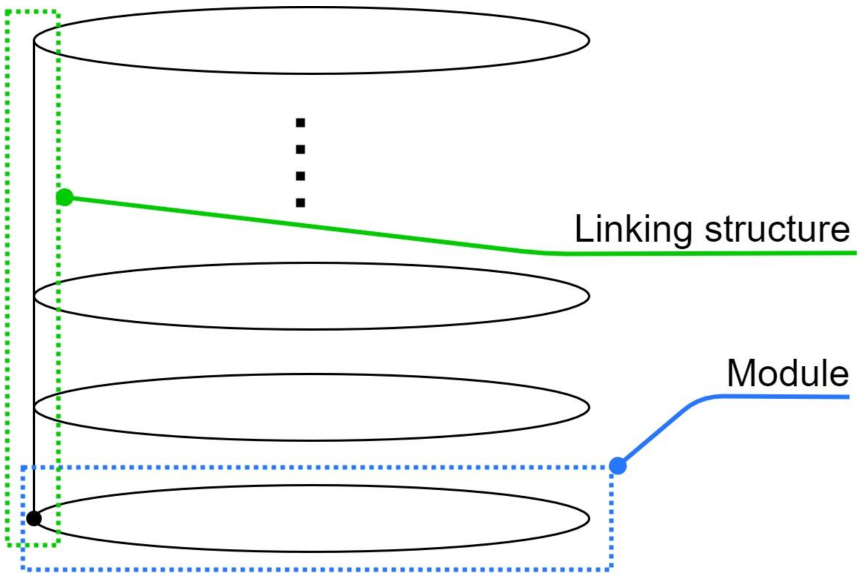
[Averbakh and Berman, 1997]

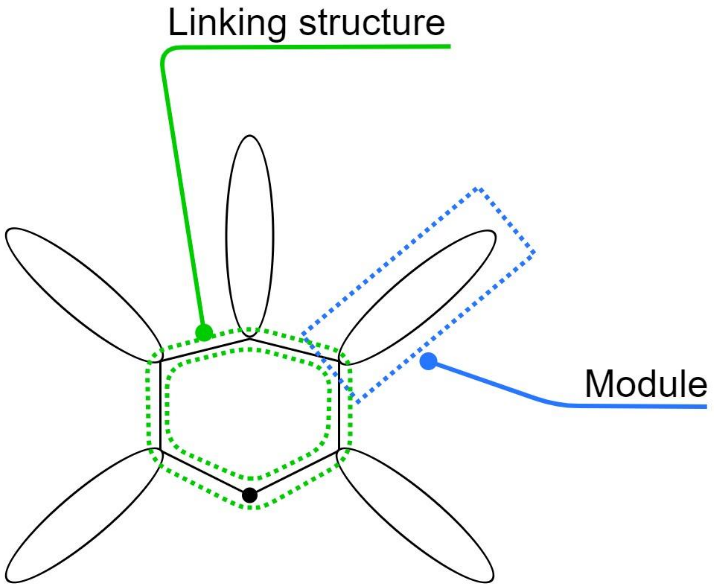
Purpose of this research

- new class of environments: *modular environments*
- analysis of *integer solutions* on *linear* modular environments
- analysis of the obtained approximation bound
- definition of an algorithm for integer solutions
- experimental comparison with two state-of-the-art algorithms

Modular environment

An environment:
- constituted by sub-parts, the modules
- connected through a linking structure





Real-world modular environments

Residential buildings

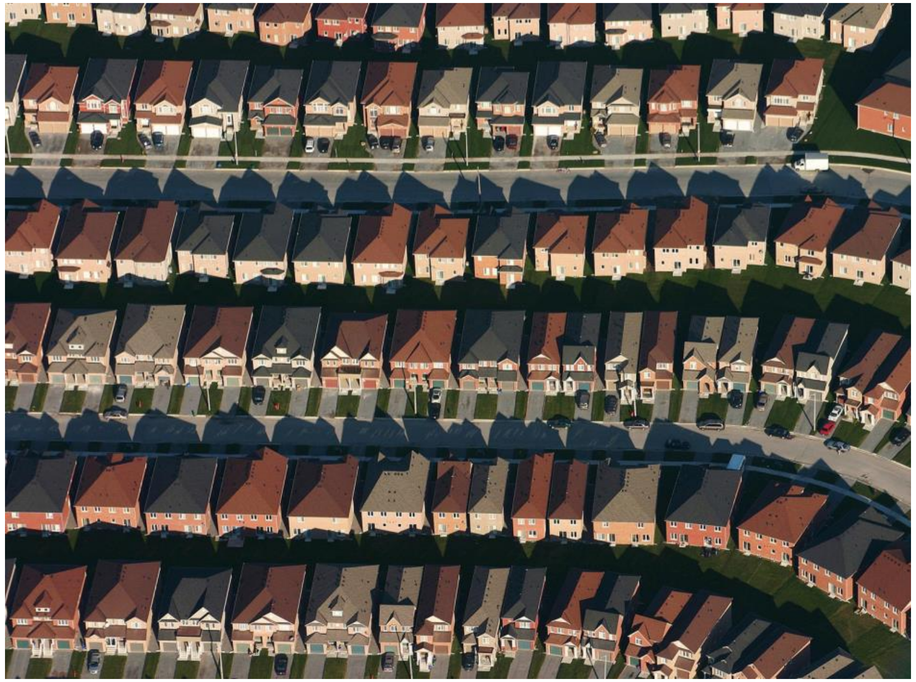
Real-world modular environments

Residential buildings

Real-world modular environments

Residential buildings

Real-world modular environments

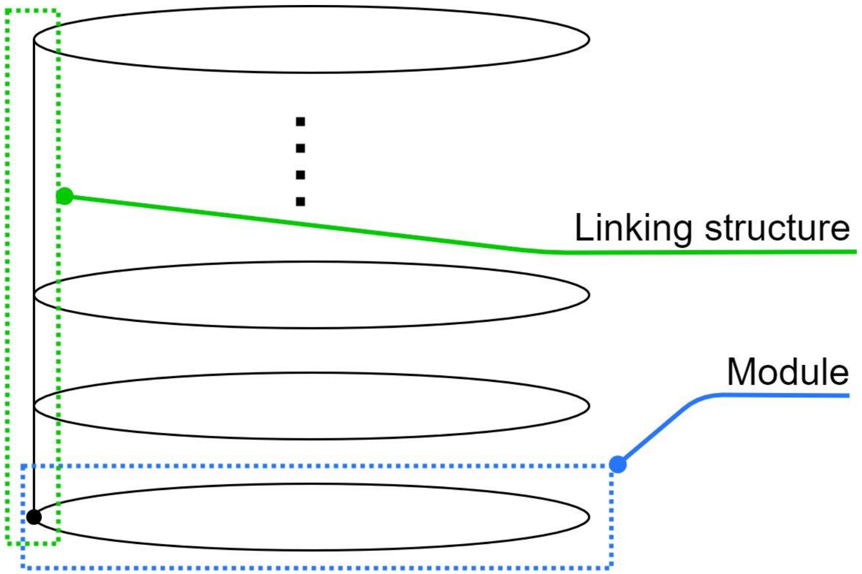


Tract housing

Linear Modular Environments

A modular environment in which:

- modules are orderly aligned
- along a linear linking structure
- connecting each module to the next one

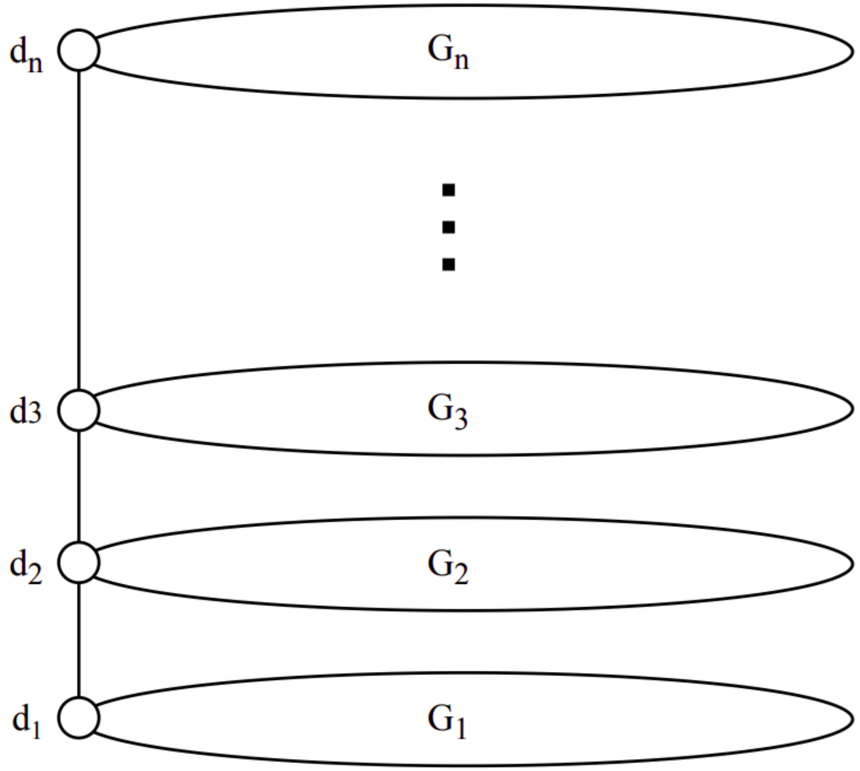


Real-world linear modular environments:

- multi-floor buildings with a single staircase
- floors of large hotels or hospitals with a single corridor
- tract houses accessible by a single street

Problem formalization

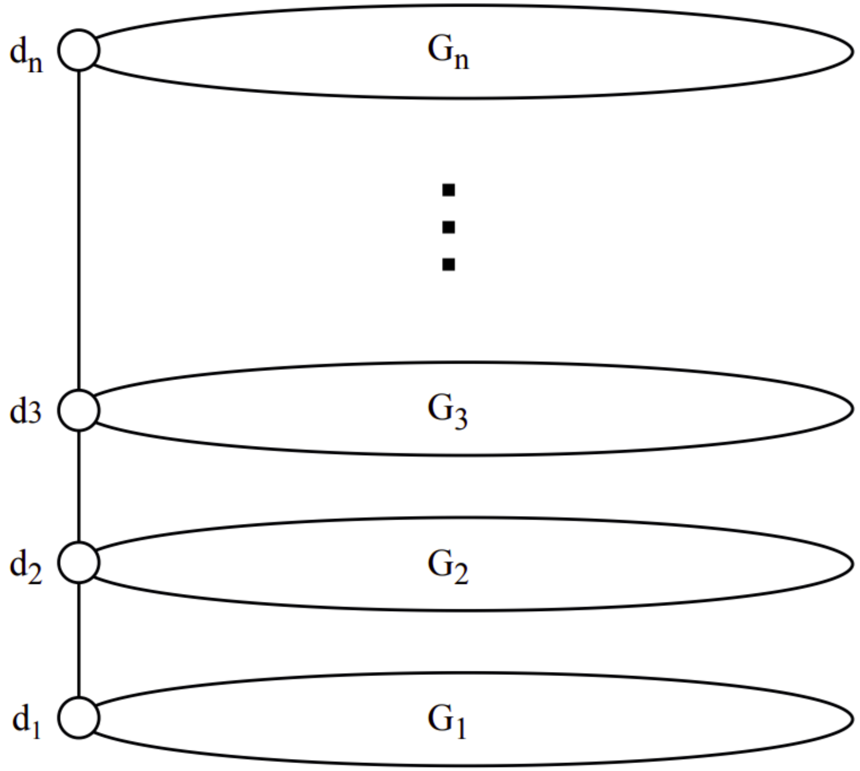
- n disjoint subgraphs $G_i = (V_i, E_i)$ → the modules
- for each module G_i a doorway $d_i \in V_i$
- d_1 is the depot
- edges (d_i, d_{i+1}) → linking structure
- a metric t defined on $V_i \times V_i$ and any pair d_i, d_{i+1}
- m homogeneous robots
- For each module G_i, we compute its *TSP* solution and therefore $t_{tsp}(i)$



Modular mTSP

Given a linear modular environment:

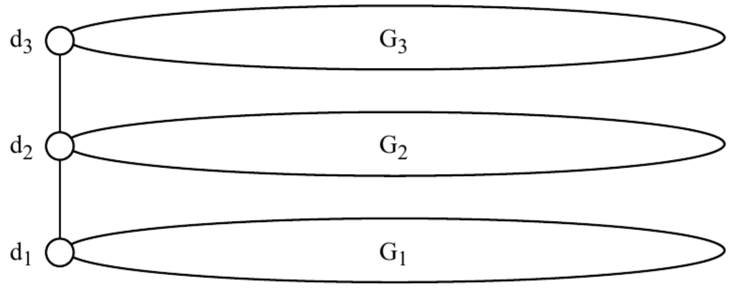
- *assign to each robot a tour*
- *starting from and ending at the depot*
- *such that all the vertices of the modules are eventually covered*
- *with the minimum makespan*



Shape index

$$\delta = \frac{\max_i t_{tsp}(i)}{\sum_i t(d_i, d_{i+1})}$$

where $t_{tsp}(i)$ is the time to cover the module G_i



$G_1 = G_2 = G_3$

$t_{tsp}(i) = 500$

$t(d_i, d_{i+1}) = 10$

$\Downarrow$

$\delta \approx 16.7$

Example of wide instance

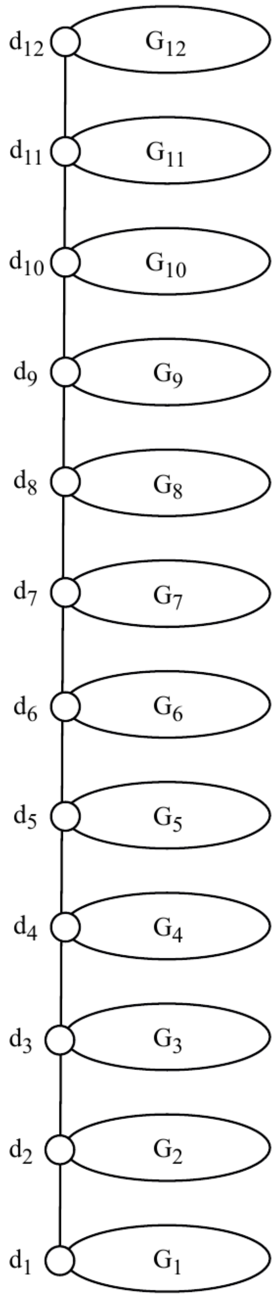
Shape index

$$\delta = \frac{\max_i t_{tsp}(i)}{\sum_i t(d_i, d_{i+1})}$$

where $t_{tsp}(i)$ is the time to cover the module G_i

wide instances: $\delta \rightarrow \infty$

deep instances: $\delta \rightarrow 0$



$G_1 = \ldots = G_i = \ldots = G_{12}$

$t_{tsp}(i) = 60$

$t(d_i, d_{i+1}) = 20$

⇩

$\delta = 0.25$

Example of deep instance

Exploiting the modularity

Modules already are clusters of vertices

Notice: n clusters, $m<n$ robots

How to exploit this peculiarity?

Intuitive approach: assign clusters to robots

Integer solutions

A solution is in integer form if for each robot *r* there exist *i,j* such that:

- for any $i \leq h \leq j$, the *h*-th module is entirely covered by *r*
- *r* does not take part to the coverage of any other module

Theorem

There must exist an integer solution such that:

$$1 \leq \frac{SOL_{int}}{OPT} \leq 1 + \frac{\delta}{2}$$

Sketch of the proof

$$OPT = (s_1^*, \ldots, s_m^*)$$

- T_r^* Time budget to cover modules
- σ_r^* Last module covered by *r*

$$SOL_{int} = (s_1, \ldots, s_m)$$

$$SOL_{int} \leq \mathrm{OPT} + \max_{1 \leq i \leq n} t_{tsp}(i)$$

Sketch of the proof

$$SOL_{int} \leq \mathrm{OPT} + \max_{1 \leq i \leq n} t_{tsp}(i)$$

$$OPT \geq 2 \sum_i t(d_i, d_{i+1})$$

$$\delta = \frac{\max_i t_{tsp}(i)}{\sum_i t(d_i, d_{i+1})}$$

$$\frac{SOL_{int}}{\mathrm{OPT}} \leq 1 + \frac{\delta}{2}$$

Approximation bound

From the Theorem: $SOL_{int} \leq (1 + \frac{\delta}{2})\, OPT$

α-approximation algorithm for TSP

$\downarrow$

α-approximation algorithm for modular mTSP

Using Christofides, 1976:

$$SOL_{int} \leq \frac{\mathbf{3}}{\mathbf{2}}(1 + \frac{\delta}{2})\ OPT$$

Approximation bound

Ours		Frederickson
$\frac{3}{2}(1+\frac{\delta}{2})$	$\overset{?}{<}$	$\frac{5}{2}-\frac{1}{m}$

For $\delta < 1 - \frac{1}{m}$ our bound is lower than that of Frederickson

For $\delta \to 0$ our bound approaches $\frac{3}{2}$

Finding integer solutions

Preliminary definitions

$$f(i, j, k)$$

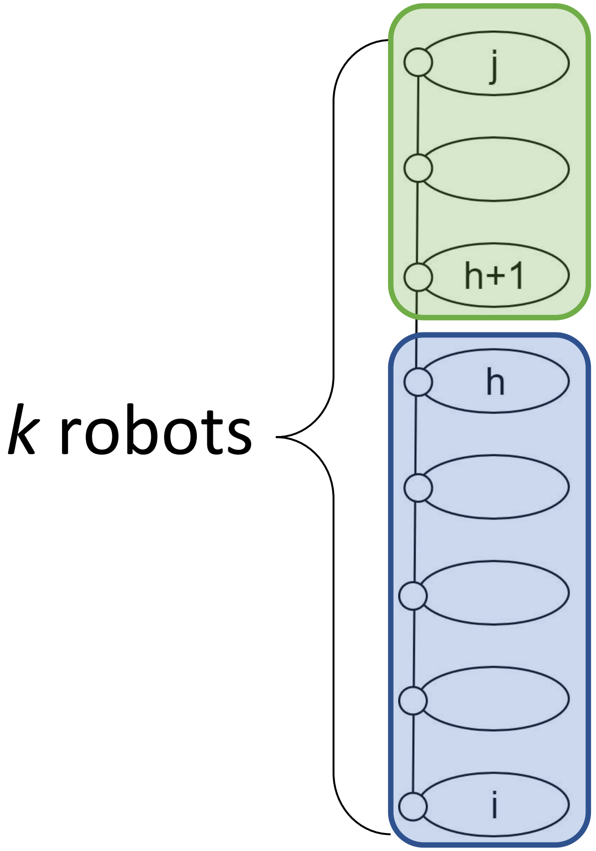
makespan for ***k*** robots

that cover modules from ***i*** to ***j***

Split point: module *h* such that

~*k/2* robots cover *(i, h)*

~*k/2* robots cover *(h+1, j)*



Finding integer solutions

Preliminary definitions

$$f(i,j,k) = \min_{i \le h \le j} \max \begin{cases} f(i,h,\lfloor k/2 \rfloor) \\ f(h+1,j,\lceil k/2 \rceil) \end{cases}$$

If all $f(\cdot,\cdot,k')$ $\quad k' < k$ are known, we can find $f(i,j,k)$ in $\mathcal{O}(j-i)$

Finding integer solutions

ALGORITHM 1. *Given a modular mTSP instance and the value of $t_{tsp}(i)$ for each module i of the instance:*

(a) *Compute $f(i, j, k)$ for the cases $i = j$ and $k = 1$.* → $\mathcal{O}(n^2)$

(b) *Set $k = 2$ robots.* → $\mathcal{O}(1)$

(c) *For any $1 \leq i \leq j \leq n$ compute $f(i, j, k)$ and store the corresponding split points.*

(d) *Increment k and repeat from (c) while $k \leq \lceil m/2 \rceil$.*

(c)–(d) → $\mathcal{O}(mn^3)$

(e) *Compute the split point for m robots visiting modules from 1 to n.* → $\mathcal{O}(n)$

(f) *For each of the resulting halves, list recursively all the split points.* → $\mathcal{O}(m)$

Naive complexity:

$$\mathcal{O}(mn^3)$$

Complexity improvements

Naive complexity $\mathcal{O}(mn^3)$

k is always divided by 2 $\mathcal{O}(n^3 \log m)$

binary search on prior *f(i, j, k)* values $\mathcal{O}(n^2 \log n \log m)$

Experimental analysis

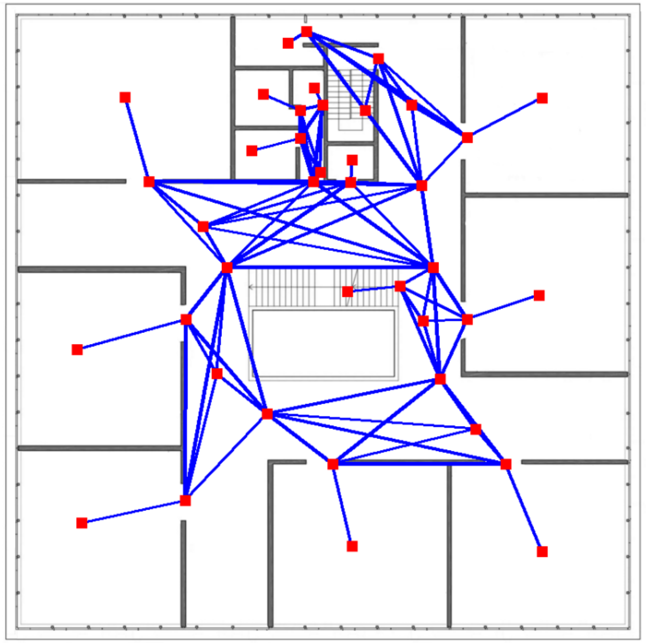
Analysis of the solutions on simple linear modular environments with identical modules, repeated

- varying *m*
- varying *n*
- varying distances between doorways (d_i, d_{i+1})

Comparison against two state-of-the-art mTSP algorithms

- Frederickson [Frederickson et al., 1976]
- AHP-mTSP [Vandermeulen et al., 2019]

Experimental analysis



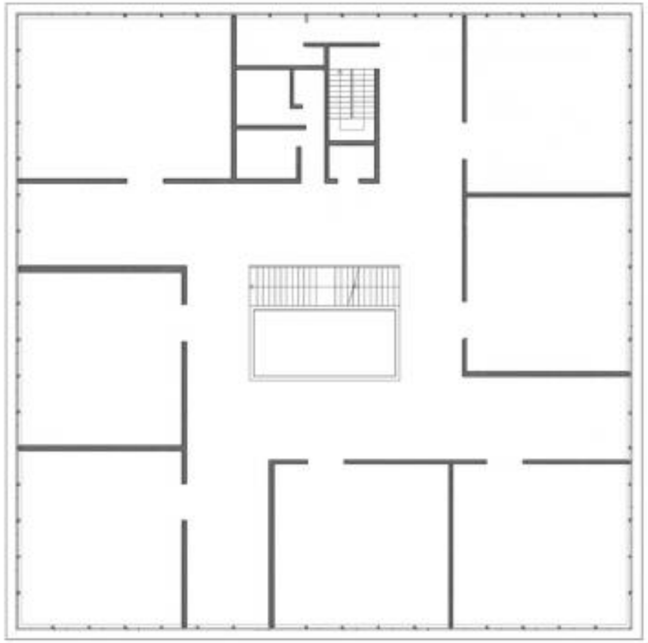
Module A

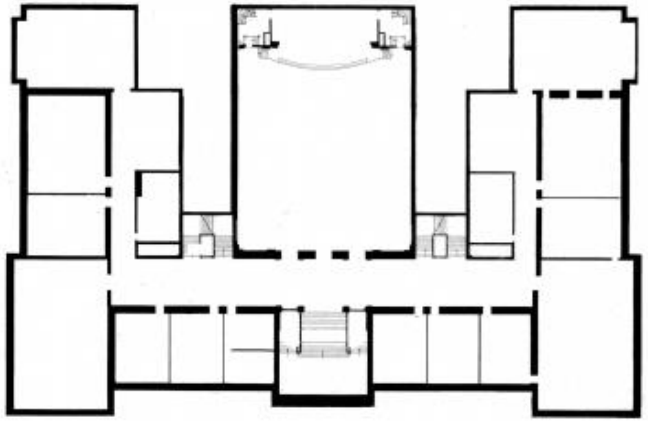
40 vertices

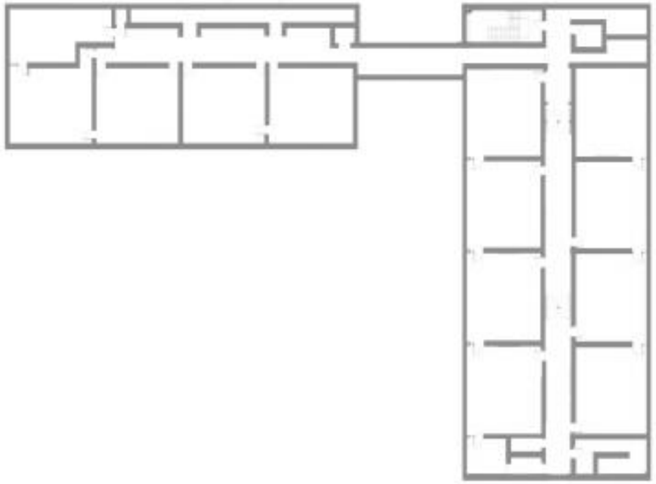
$t_{tsp}(A) = 198$ m

circular topology

Experimental analysis







Module A

40 vertices

$t_{tsp}(A) = 198$ m

circular topology

Module B

47 vertices

$t_{tsp}(A) = 347$ m

star topology

Module C

80 vertices

$t_{tsp}(A) = 438$ m

linear topology

Identical modules

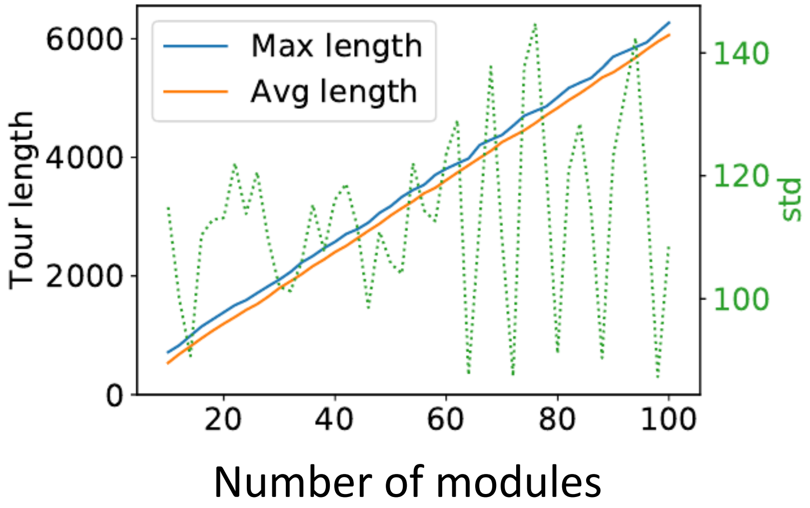
Environments with

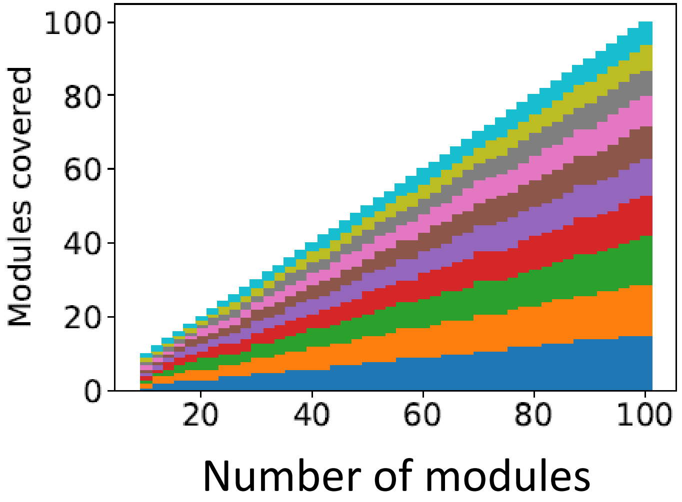
n (possibly varying) modules of type *Module B*

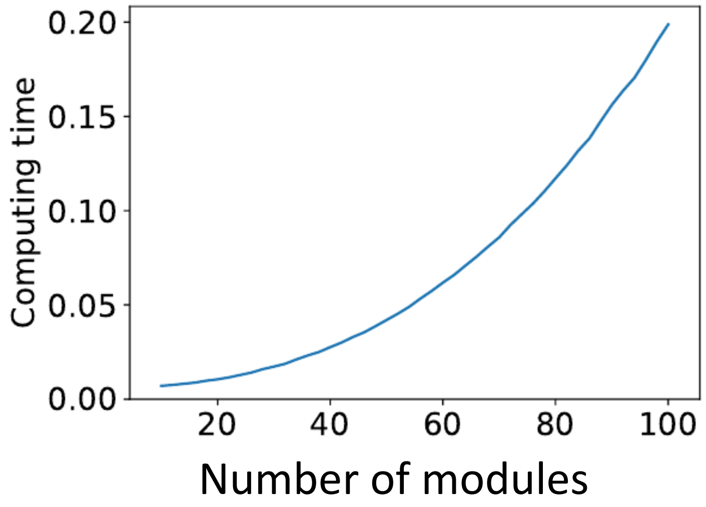
Covered with

m (possibly varying) robots

Identical modules

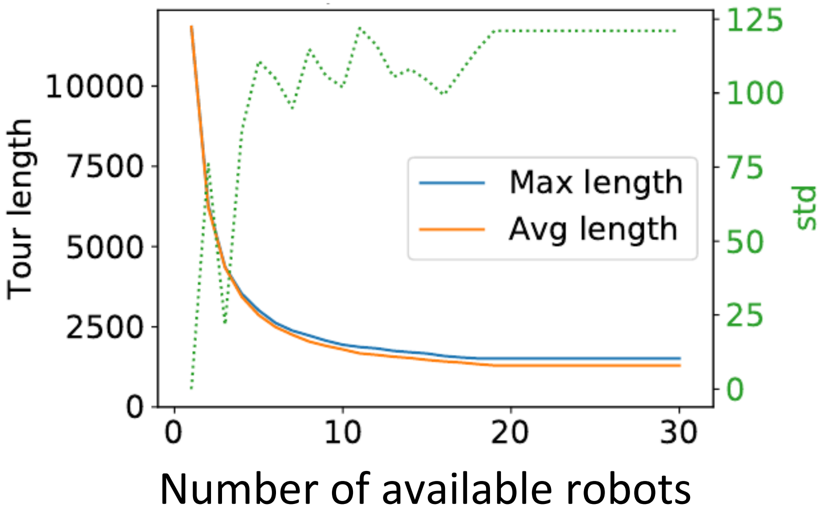


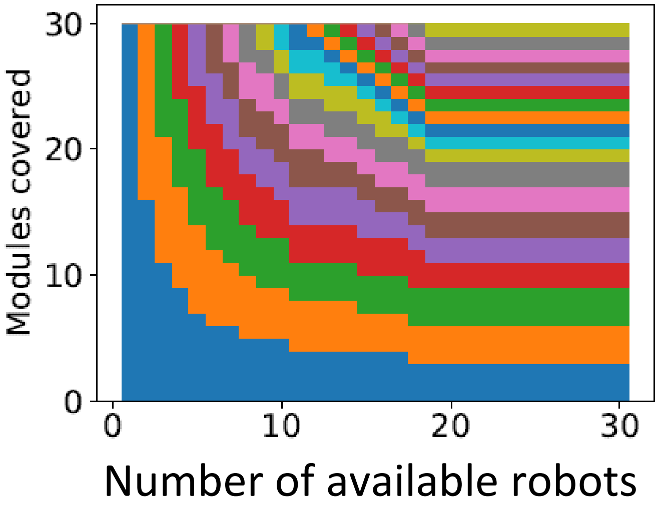


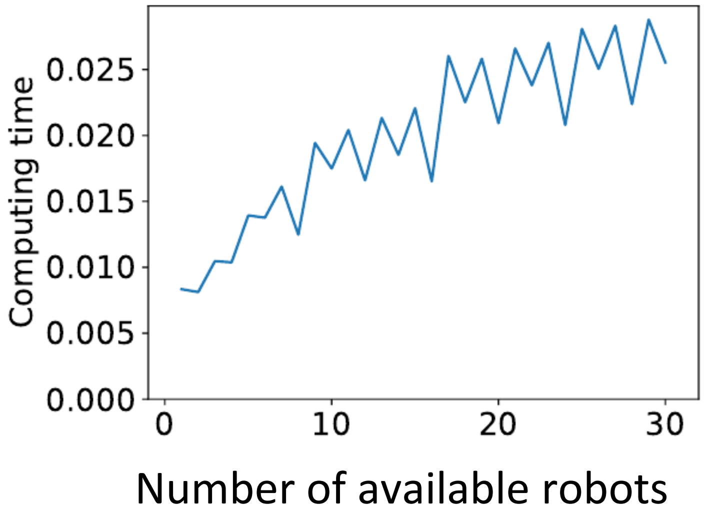


m=10, distances=20

Identical modules

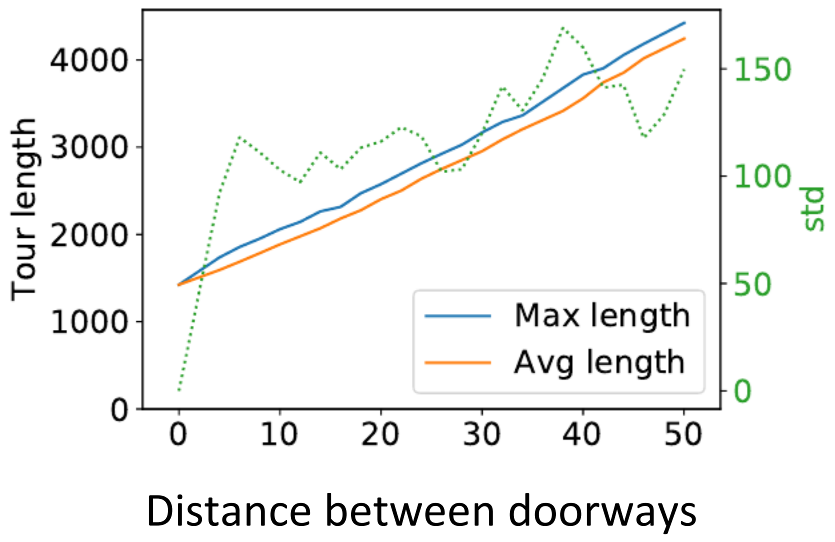


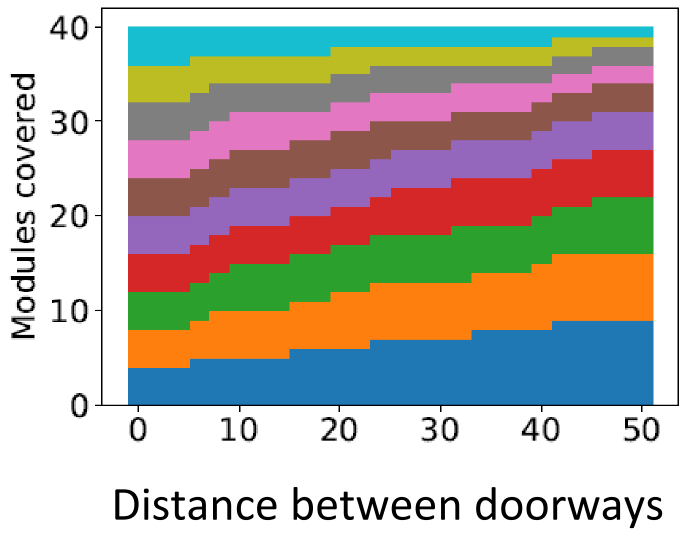


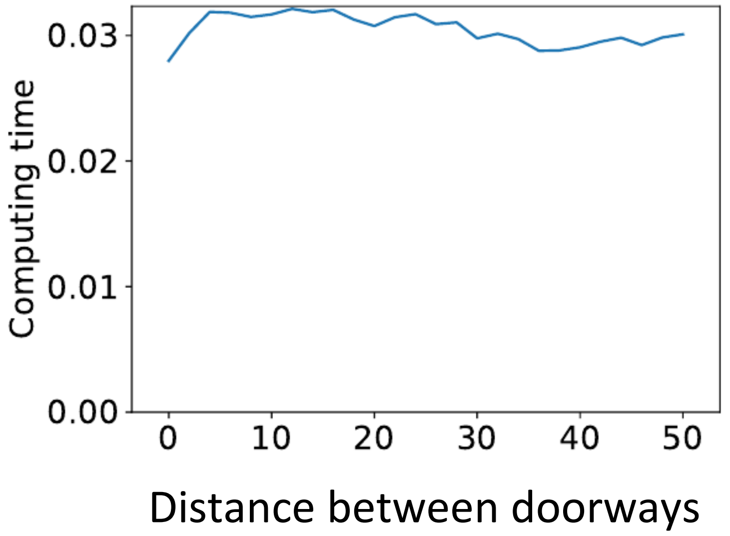


n=30, distances=20

Identical modules







m=10, n=40

Comparison on complex environments

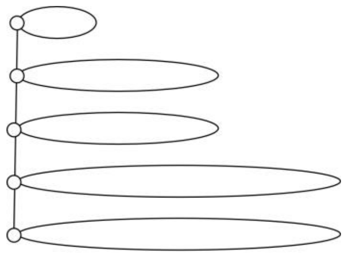
27 comparison cases:

- three patterns: *random*, *decreasing*, *increasing*
- three values of *n*: *30, 60, 120*
- three values of *m*: *5, 10, 20*

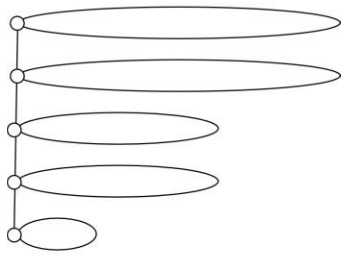
Distance between doorways fixed to 20

Timeout of 1 hour for each instance

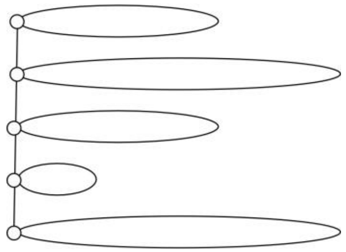
Decreasing pattern:



Increasing pattern:



Random pattern:



Comparison on complex environments

First comparison algorithm: Frederickson

- compute the single-tour *R* over the whole environment
- split the TSP according to a heuristic

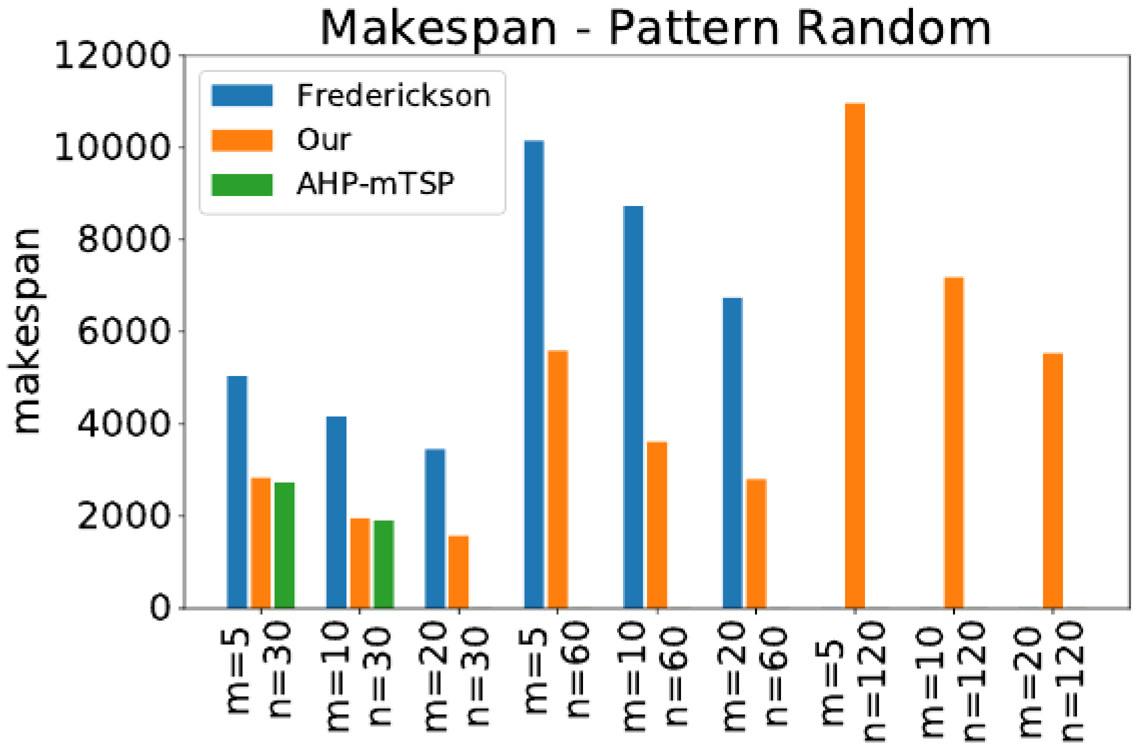
[Frederickson et al., 1976]

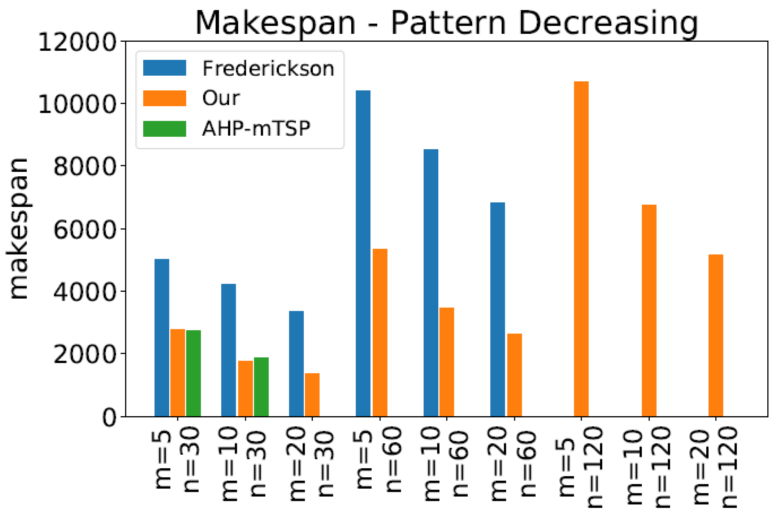
Second comparison algorithm: AHP-mTSP

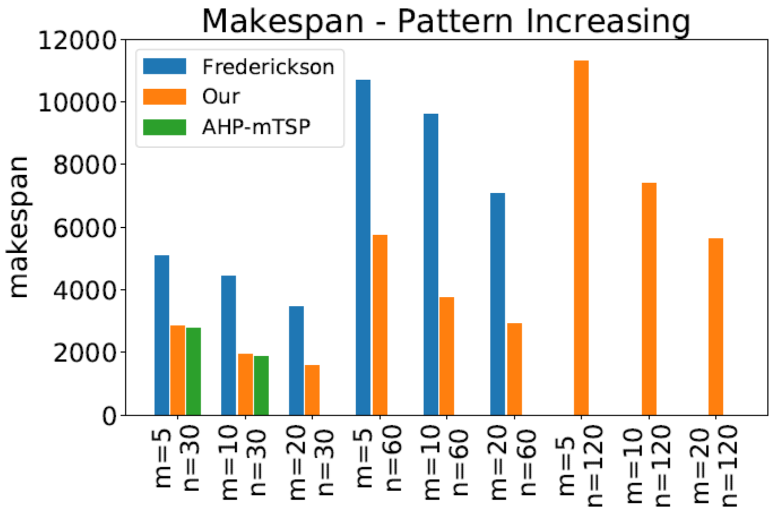
- random partition of the whole environment (into *m* partitions)
- apply a series of improvements to the *m* partitions, repeatedly

[I. Vandermeulen et al., 2019]

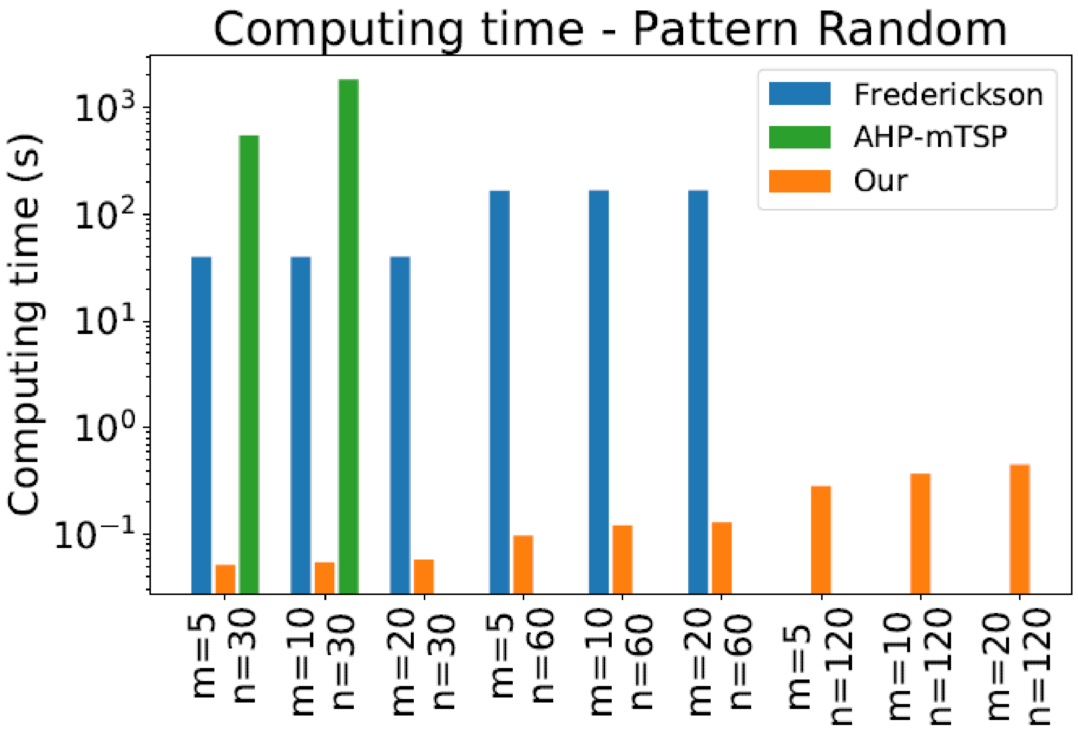
Comparison on complex environments

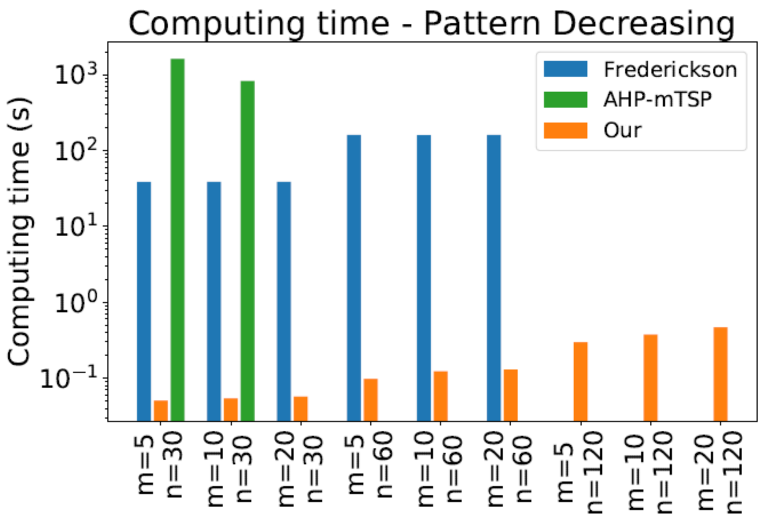


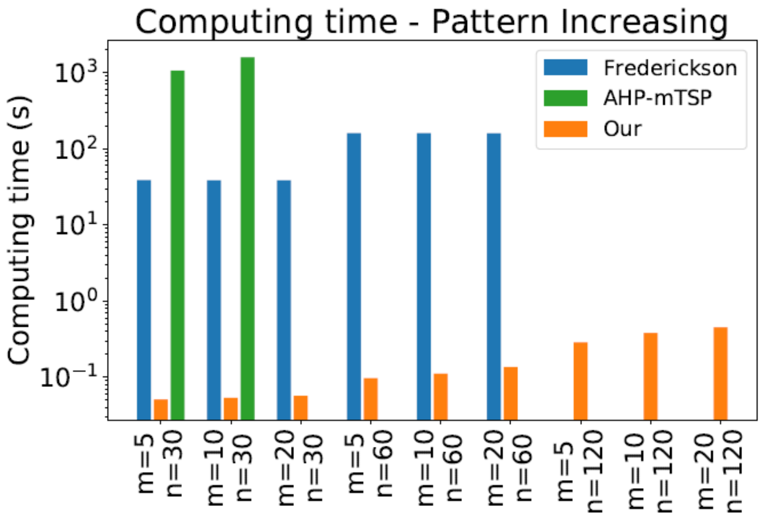




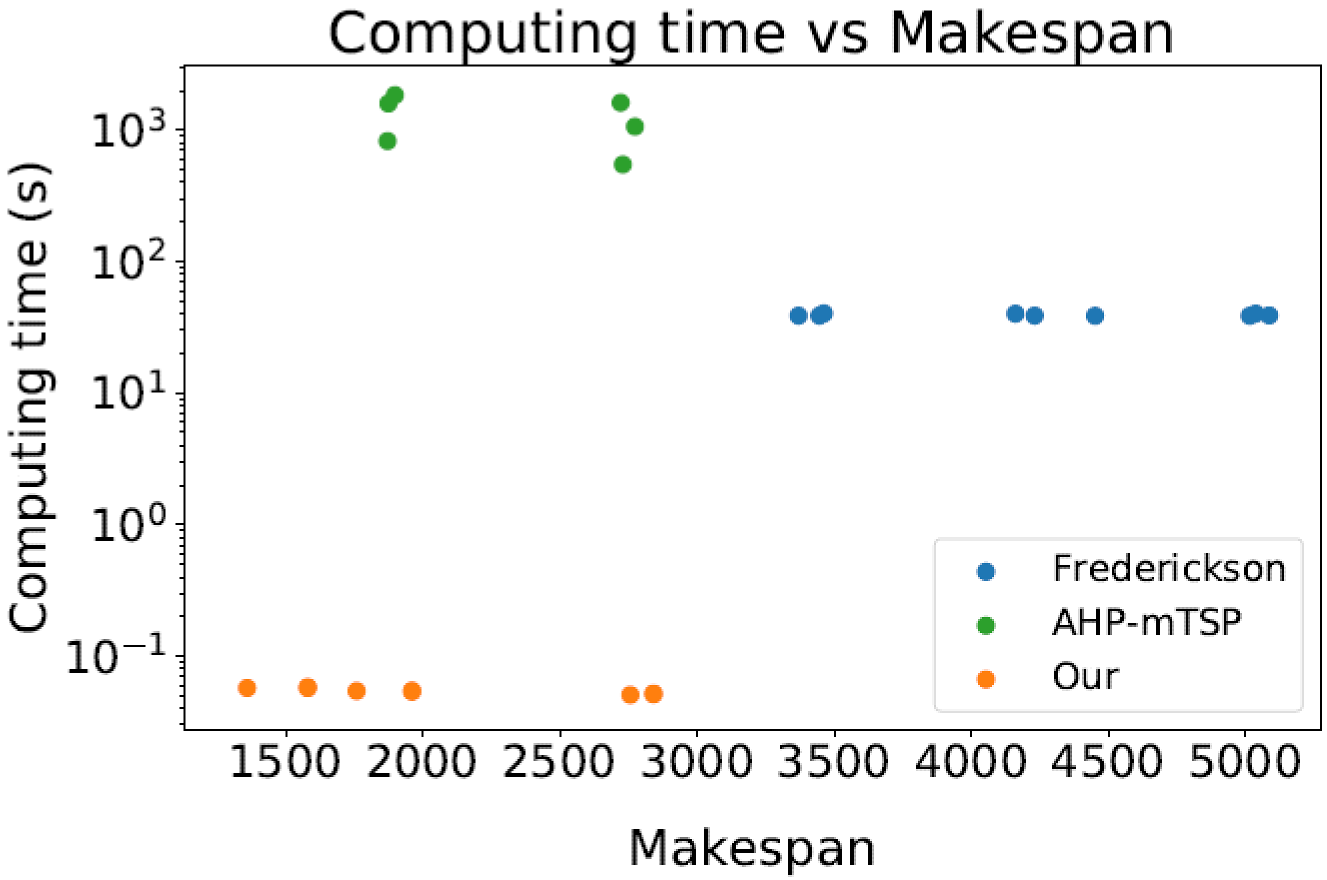
Comparison on complex environments







Comparison on complex environments



Conclusion and future work

Efficient approximation algorithms for linear modular environments with an approximation bound lower than Frederickson bound

Our approach experimentally outperforms state-of-the-art algorithms

Future extensions:

- non-linear modular environments: circles, grids, trees
- multiple doorways
- non-integer solutions

Thank you for the attention

Questions?