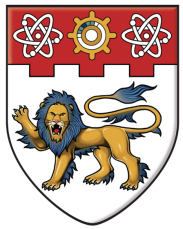
A Heuristic Algorithm to Optimize Execution Time of Multi-Robot Path

Diego Deplano

Nanyang Technological University & University of Cagliari

ICCA, Ohrid, Macedonia, 4 July 2017



NANYANG TECHNOLOGICAL UNIVERSITY

Outline

Outline

State identification problem for discrete event systems

Problem

Given a trajectory solution for a motion planning problem, how can we optimize it in terms of execution time?

A way to do it for systems modeled by timed automata is to rearrange events in the trajectory in order to maximize parrallelism.

- Preserving priorities:
  the aim is to establish the earliest firing time for each event in the trajectory.

- Changing priorities:
  the aim is to identify sub-trajectories requiring common resources and change their priorities over such a resources.

State identification problem for discrete event systems

Problem

Given a trajectory solution for a motion planning problem, how can we optimize it in terms of execution time?

A way to do it for systems modeled by timed automata is to rearrange events in the trajectory in order to maximize parrallelism.

- Preserving priorities:
  the aim is to establish the earliest firing time for each event in the trajectory.

- Changing priorities:
  the aim is to identify sub-trajectories requiring common resources and change their priorities over such a resources.

State identification problem for discrete event systems

Problem

Given a trajectory solution for a motion planning problem, how can we optimize it in terms of execution time?

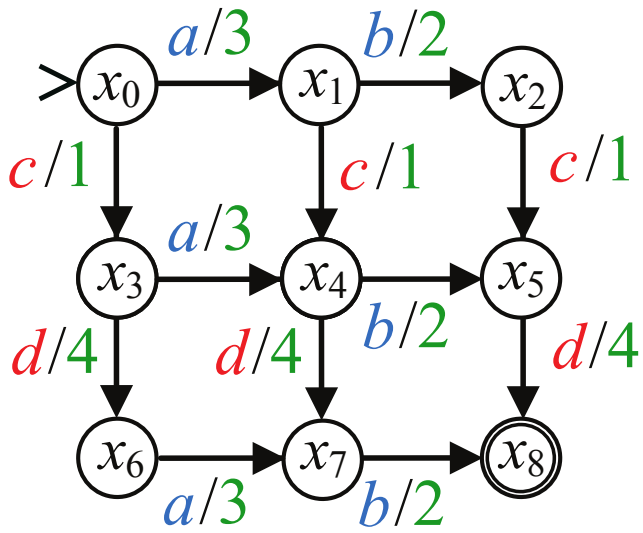
A way to do it for systems modeled by timed automata is to rearrange events in the trajectory in order to maximize parrallelism.

- Preserving priorities:
  the aim is to establish the earliest firing time for each event in the trajectory.

- Changing priorities:
  the aim is to identify sub-trajectories requiring common resources and change their priorities over such a resources.

Example

Consider this timed automaton $\mathcal{G}$ with events of machine 1 and machine 2 and their time transitions. Where a, c require the same resource.



Given $t = abcd \in L_m(\mathcal{G})$

Preserving priorities

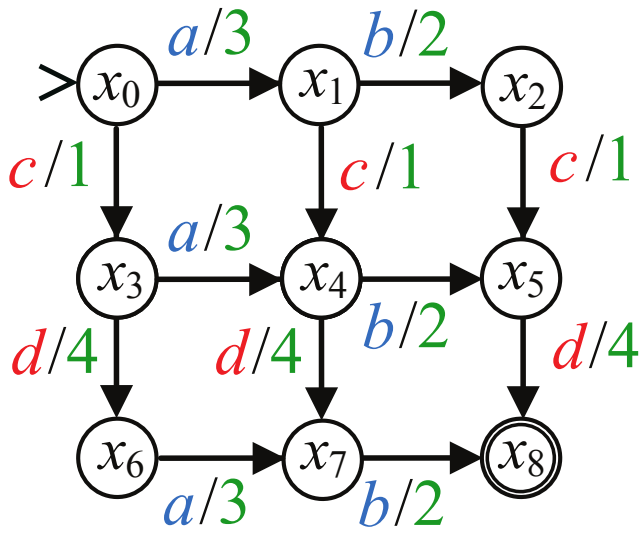
output trajectory $t_1 = acbd \in L_m(\mathcal{G})$

Changing priorities

output trajectory $t_2 = cdab \in L_m(\mathcal{G})$

Example

Consider this timed automaton $\mathcal{G}$ with events of machine 1 and machine 2 and their time transitions. Where a, c require the same resource.



Given $t = abcd \in L_m(\mathcal{G})$

Preserving priorities

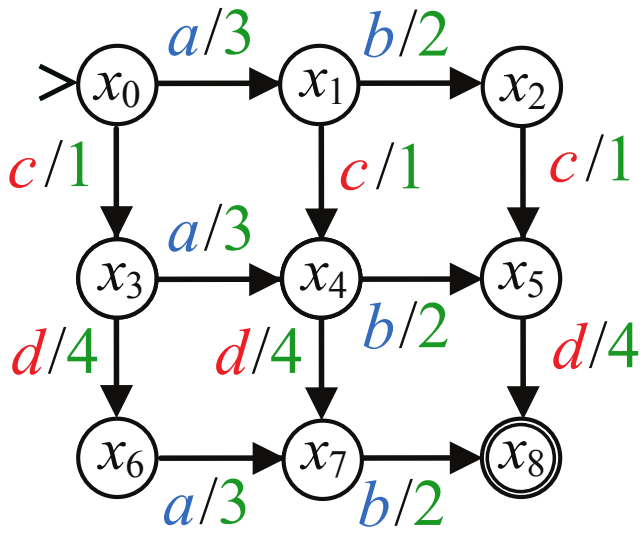
output trajectory $t_1 = a\boldsymbol{c}bd \in L_m(\mathcal{G})$

Changing priorities

output trajectory $t_2 = \boldsymbol{c}\boldsymbol{d}ab \in L_m(\mathcal{G})$

Example

Consider this timed automaton $\mathcal{G}$ with events of machine 1 and machine 2 and their time transitions. Where a, c require the same resource.



Given $t = abcd \in L_m(\mathcal{G})$

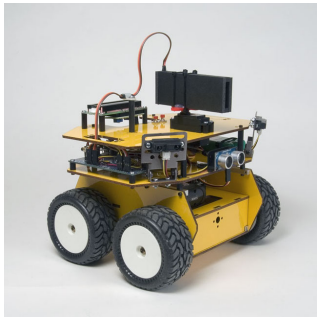
Preserving priorities

output trajectory $t_1 = a\boldsymbol{c}bd \in L_m(\mathcal{G})$

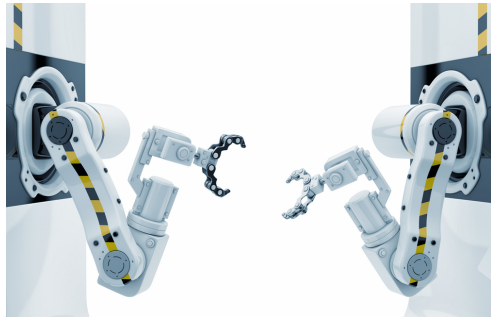
Changing priorities

output trajectory $t_2 = \boldsymbol{cd}ab \in L_m(\mathcal{G})$

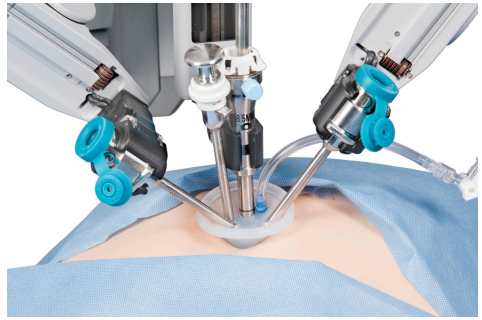
Applications



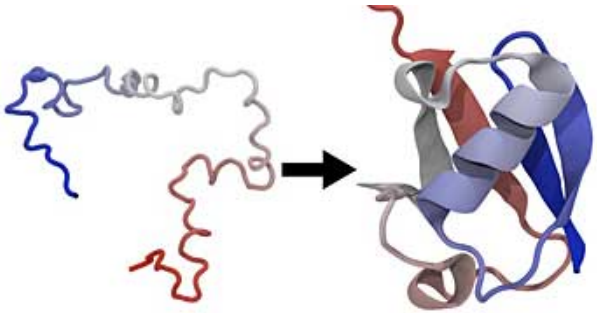
(navigation)



(manufacturing)



(surgery)



(protein folding)

(videogames)

(graphic animation)

Field of interest: Multi-Robot Motion-Planning

Question: How can we find a trajectory which directs each robot from its current position to its destination such that the total execution time is small (but not necessarily optimal)?

Main problems:

- State space grows exponentially with # of robots being coordinate and dimensions of the environment (state space explosion);
- Algorithms to compute optimal trajectory: PSPACE-Hard;
- Algorithms to compute sub-optimal trajectory: far from the optimum.

Answer: Implementing an algorithm which efficiently improves a given trajectory by merging preserving and changing priorities techniques.

Field of interest: Multi-Robot Motion-Planning

Question: How can we find a trajectory which directs each robot from its current position to its destination such that the total execution time is small (but not necessarily optimal)?

Main problems:

- State space grows exponentially with # of robots being coordinate and dimensions of the environment (state space explosion);
- Algorithms to compute optimal trajectory: PSPACE-Hard;
- Algorithms to compute sub-optimal trajectory: far from the optimum.

Answer: Implementing an algorithm which efficiently improves a given trajectory by merging preserving and changing priorities techniques.

Field of interest: Multi-Robot Motion-Planning

Question: How can we find a trajectory which directs each robot from its current position to its destination such that the total execution time is small (but not necessarily optimal)?

Main problems:

- State space grows exponentially with # of robots being coordinate and dimensions of the environment (state space explosion);
- Algorithms to compute optimal trajectory: PSPACE-Hard;
- Algorithms to compute sub-optimal trajectory: far from the optimum.

Answer: Implementing an algorithm which efficiently improves a given trajectory by merging preserving and changing priorities techniques.

Field of interest: Multi-Robot Motion-Planning

Question: How can we find a trajectory which directs each robot from its current position to its destination such that the total execution time is small (but not necessarily optimal)?

Main problems:

- State space grows exponentially with # of robots being coordinate and dimensions of the environment (state space explosion);
- Algorithms to compute optimal trajectory: PSPACE-Hard;
- Algorithms to compute sub-optimal trajectory: far from the optimum.

Answer: Implementing an algorithm which efficiently improves a given trajectory by merging preserving and changing priorities techniques.

Field of interest: Multi-Robot Motion-Planning

Question: How can we find a trajectory which directs each robot from its current position to its destination such that the total execution time is small (but not necessarily optimal)?

Main problems:

- State space grows exponentially with # of robots being coordinate and dimensions of the environment (state space explosion);
- Algorithms to compute optimal trajectory: PSPACE-Hard;
- Algorithms to compute sub-optimal trajectory: far from the optimum.

Answer: Implementing an algorithm which efficiently improves a given trajectory by merging preserving and changing priorities techniques.

Topics that will be discussed

- Description of the Multi-Robot Motion-Planning problem in a DES framework
- Execution time of the solution
- A preserving priorities approach to optimize a trajectory.
- The novel changing priorities approach subject of this work to improve a trajectory.
- Some experimental results to evaluate effectiveness the presented approach.

Topics that will be discussed

- Description of the Multi-Robot Motion-Planning problem in a DES framework
- Execution time of the solution
- A preserving priorities approach to optimize a trajectory.
- The novel changing priorities approach subject of this work to improve a trajectory.
- Some experimental results to evaluate effectiveness the presented approach.

Topics that will be discussed

- Description of the Multi-Robot Motion-Planning problem in a DES framework
- Execution time of the solution
- A preserving priorities approach to optimize a trajectory.
- The novel changing priorities approach subject of this work to improve a trajectory.
- Some experimental results to evaluate effectiveness the presented approach.

Topics that will be discussed

- Description of the Multi-Robot Motion-Planning problem in a DES framework
- Execution time of the solution
- A preserving priorities approach to optimize a trajectory.
- The novel changing priorities approach subject of this work to improve a trajectory.
- Some experimental results to evaluate effectiveness the presented approach.

Topics that will be discussed

- Description of the Multi-Robot Motion-Planning problem in a DES framework
- Execution time of the solution
- A preserving priorities approach to optimize a trajectory.
- The novel changing priorities approach subject of this work to improve a trajectory.
- Some experimental results to evaluate effectiveness the presented approach.

Topics that will be discussed

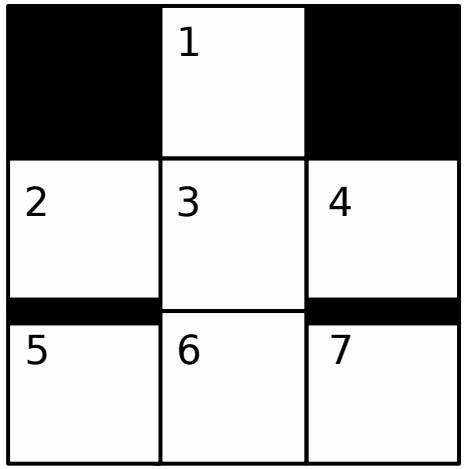
- Description of the Multi-Robot Motion-Planning problem in a DES framework
- Execution time of the solution
- A preserving priorities approach to optimize a trajectory.
- The novel changing priorities approach subject of this work to improve a trajectory.
- Some experimental results to evaluate effectiveness the presented approach.

Outline

Environment

The environment where a robot operates can be partitioned into a grid, where:

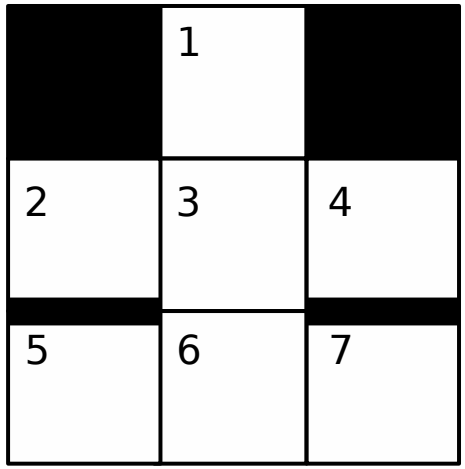
- **White squares** denote accessible positions;
- **Black squares** denote inaccessible positions;
- **Thin sides** denote an available transition;
- **Thick sides** denote an unavailable transition.



Environment

The environment where a robot operates can be partitioned into a grid, where:

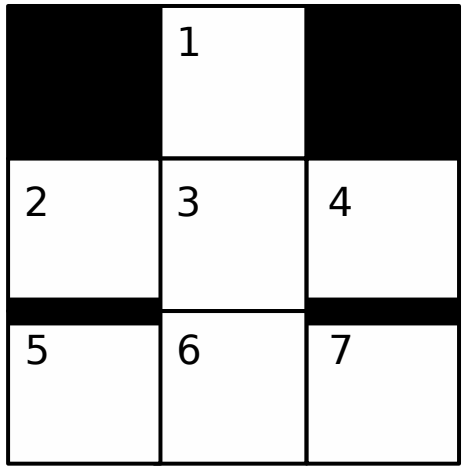
- **White squares** denote accessible positions;
- **Black squares** denote inaccessible positions;
- **Thin sides** denote an available transition;
- **Thick sides** denote an unavailable transition.



Environment

The environment where a robot operates can be partitioned into a grid, where:

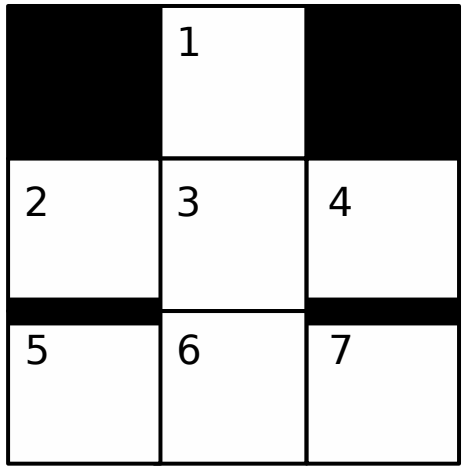
- **White squares** denote accessible positions;
- **Black squares** denote inaccessible positions;
- **Thin sides** denote an available transition;
- **Thick sides** denote an unavailable transition.



Environment

The environment where a robot operates can be partitioned into a grid, where:

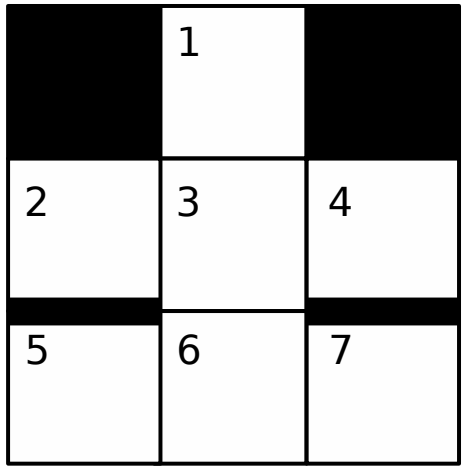
- **White squares** denote accessible positions;
- **Black squares** denote inaccessible positions;
- **Thin sides** denote an available transition;
- **Thick sides** denote an unavailable transition.



Environment

The environment where a robot operates can be partitioned into a grid, where:

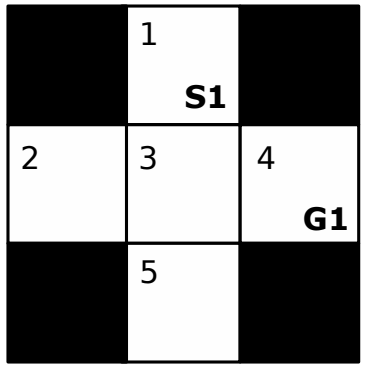
- **White squares** denote accessible positions;
- **Black squares** denote inaccessible positions;
- **Thin sides** denote an available transition;
- **Thick sides** denote an unavailable transition.

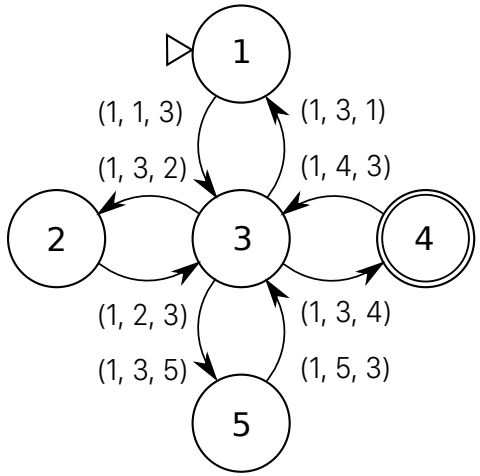


Automaton representation

For each robot we define an automaton $G = (X, \Sigma, \delta, x_0, x_m)$ with

- X is a set of states: each state is a position in the environment;
- Σ is a set of events: each event is a transition of the robot;
- $\delta : X \times \Sigma \to X$ is a state transition function;
- x_0 is the initial state: the initial position of the robot;
- x_m is the final state: the goal position of the robot.

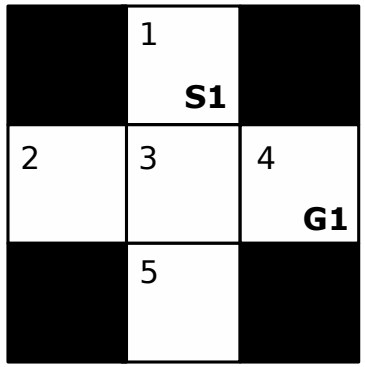


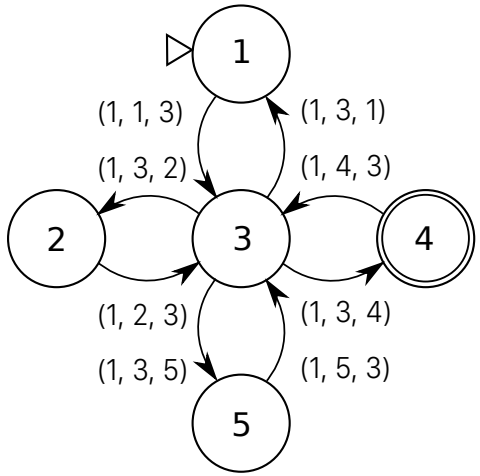


Automaton representation

For each robot we define an automaton $G = (X, \Sigma, \delta, x_0, x_m)$ with

- X is a set of states: each state is a position in the environment;
- Σ is a set of events: each event is a transition of the robot;
- $\delta : X \times \Sigma \to X$ is a state transition function;
- x_0 is the initial state: the initial position of the robot;
- x_m is the final state: the goal position of the robot.

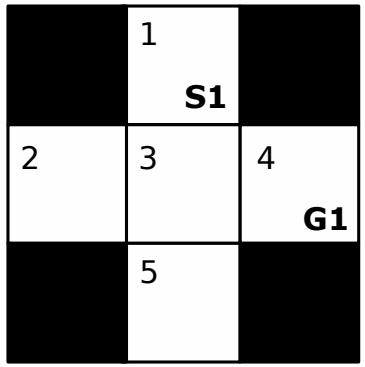


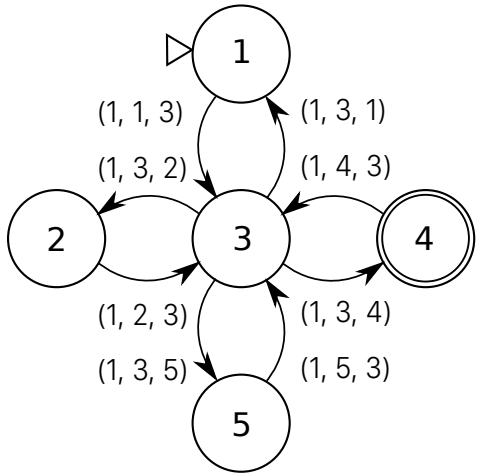


Automaton representation

For each robot we define an automaton $G = (X, \Sigma, \delta, x_0, x_m)$ with

- X is a set of states: each state is a position in the environment;
- Σ is a set of events: each event is a transition of the robot;
- $\delta : X \times \Sigma \to X$ is a state transition function;
- x_0 is the initial state: the initial position of the robot;
- x_m is the final state: the goal position of the robot.

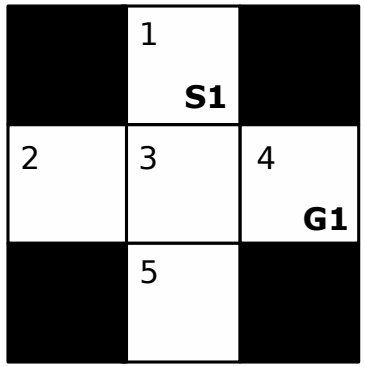


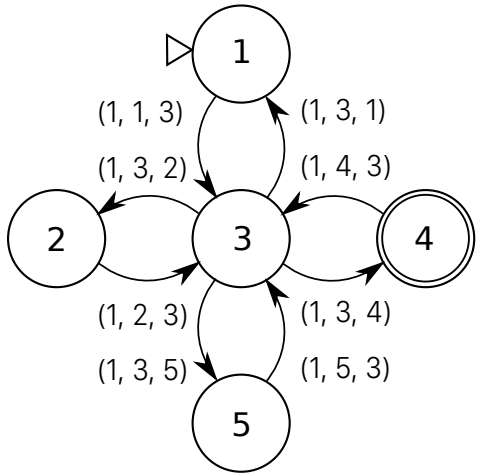


Automaton representation

For each robot we define an automaton $G = (X, \Sigma, \delta, x_0, x_m)$ with

- X is a set of states: each state is a position in the environment;
- Σ is a set of events: each event is a transition of the robot;
- $\delta : X \times \Sigma \to X$ is a state transition function;
- x_0 is the initial state: the initial position of the robot;
- x_m is the final state: the goal position of the robot.

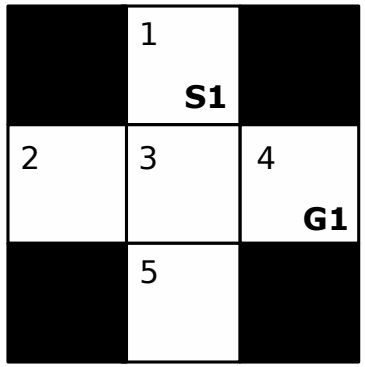


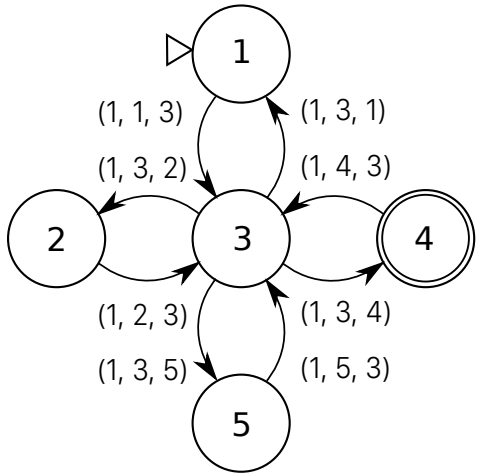


Automaton representation

For each robot we define an automaton $G = (X, \Sigma, \delta, x_0, x_m)$ with

- X is a set of states: each state is a position in the environment;
- Σ is a set of events: each event is a transition of the robot;
- $\delta : X \times \Sigma \to X$ is a state transition function;
- x_0 is the initial state: the initial position of the robot;
- x_m is the final state: the goal position of the robot.

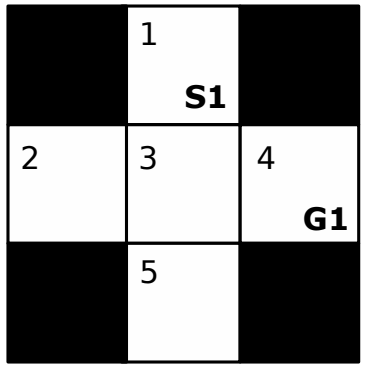


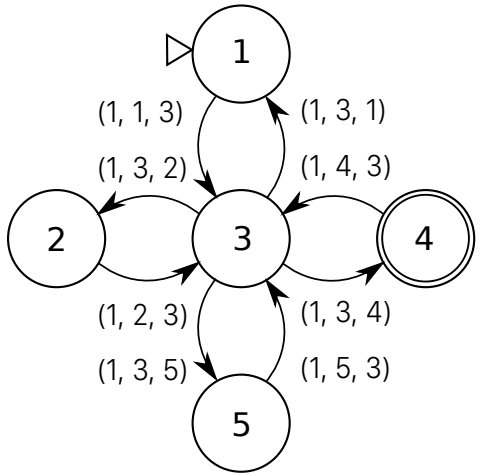


Automaton representation

For each robot we define an automaton $G = (X, \Sigma, \delta, x_0, x_m)$ with

- X is a set of states: each state is a position in the environment;
- Σ is a set of events: each event is a transition of the robot;
- $\delta : X \times \Sigma \to X$ is a state transition function;
- x_0 is the initial state: the initial position of the robot;
- x_m is the final state: the goal position of the robot.

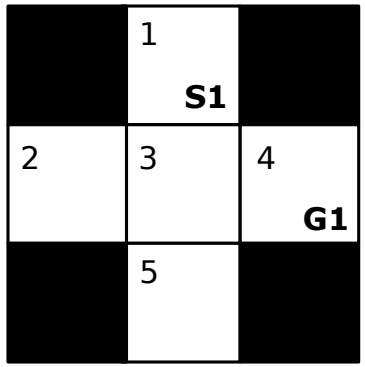


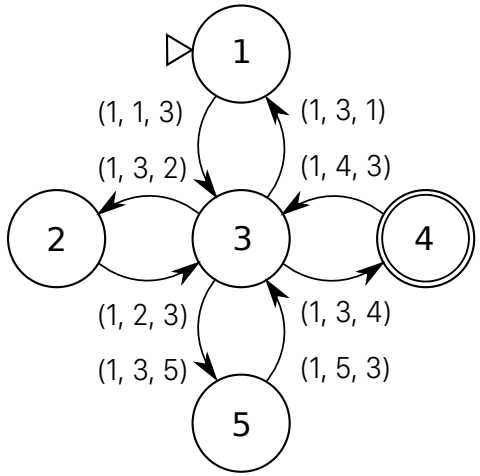


Automaton representation

Example of $G = (X, \Sigma, \delta, x_0, x_m)$ automaton of the robot 1

- $X = \{1, 2, 3, 4, 5\}$;
- $\Sigma = \{(1,3,2), (1,3,1), (1,4,3), (1,2,3), (1,3,5), (1,3,4), (1,5,3)\}$;
- $\delta : X \times \Sigma \to X$;
- $x_0 = 1$;
- $x_m = 4$.

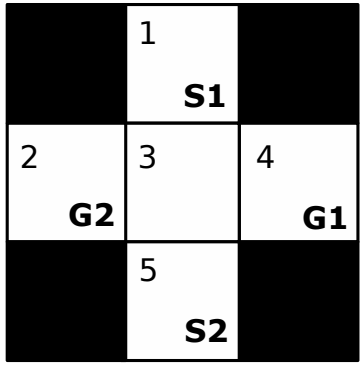




Solution

Defined $G = G_1 \times \cdots G_n$ the product of the n robot automata in the environment, a string $s \in L_m(G)$ is a valid solution to the problem – we call it trajectory – if it directs each robot from it current position to its destination without collisions.

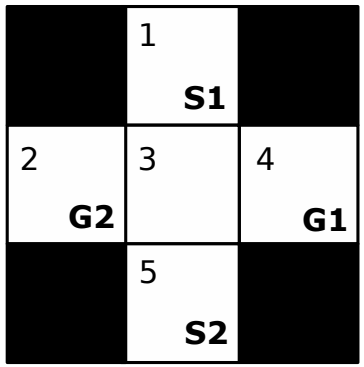
Example: $t = (1,1,3)(1,3,4)(2,5,3)(2,3,2)$ is a valid trajectory while $t' = (1,1,3)(2,5,3)(1,3,4)(2,3,2)$ is not because robots 1 and 2 collide in position 3.



Solution

Defined $G = G_1 \times \cdots G_n$ the product of the n robot automata in the environment, a string $s \in L_m(G)$ is a valid solution to the problem – we call it trajectory – if it directs each robot from it current position to its destination without collisions.

Example: $t = (1,1,3)(1,3,4)(2,5,3)(2,3,2)$ is a valid trajectory while $t' = (1,1,3)(2,5,3)(1,3,4)(2,3,2)$ is not because robots 1 and 2 collide in position 3.



Outline

Execution time

We now introduce the concept of the time in the system. When a string s is executed the events must be fired sequentially based on the order in the string. Therefore, each event $\sigma_k \in s$ has:

1. A starting time t_k;
2. An execution time $f(\sigma_k)$;
3. A completion time $t_k + f(\sigma_k)$.

A time stamp $\rho = [t_k \in \mathbb{R}_0^+ | k \in \{1, \cdots, |s|\}]$ of the string s with respect to G which establishes the starting time of each event is a nondecreasing list of nonnegative real numbers.
A time stamp is legal if will fire each event avoiding simultaneous execution of mutually exclusive events.
A time stamp is optimal if the execution time of the string is minimum.

Execution time

We now introduce the concept of the time in the system. When a string s is executed the events must be fired sequentially based on the order in the string. Therefore, each event $\sigma_k \in s$ has:

1. A starting time t_k;
2. An execution time $f(\sigma_k)$;
3. A completion time $t_k + f(\sigma_k)$.

A time stamp $\rho = [t_k \in \mathbb{R}_0^+ | k \in \{1, \cdots, |s|\}]$ of the string s with respect to G which establishes the starting time of each event is a nondecreasing list of nonnegative real numbers.
A time stamp is legal if will fire each event avoiding simultaneous execution of mutually exclusive events.
A time stamp is optimal if the execution time of the string is minimum.

Execution time

We now introduce the concept of the time in the system. When a string s is executed the events must be fired sequentially based on the order in the string. Therefore, each event $\sigma_k \in s$ has:

1. A starting time t_k;
2. An execution time $f(\sigma_k)$;
3. A completion time $t_k + f(\sigma_k)$.

A time stamp $\rho = [t_k \in \mathbb{R}_0^+ | k \in \{1, \cdots, |s|\}]$ of the string s with respect to G which establishes the starting time of each event is a nondecreasing list of nonnegative real numbers.
A time stamp is legal if will fire each event avoiding simultaneous execution of mutually exclusive events.
A time stamp is optimal if the execution time of the string is minimum.

Execution time

We now introduce the concept of the time in the system. When a string s is executed the events must be fired sequentially based on the order in the string. Therefore, each event $\sigma_k \in s$ has:

1. A starting time t_k;
2. An execution time $f(\sigma_k)$;
3. A completion time $t_k + f(\sigma_k)$.

A time stamp $\rho = [t_k \in \mathbb{R}_0^+ | k \in \{1, \cdots, |s|\}]$ of the string s with respect to G which establishes the starting time of each event is a nondecreasing list of nonnegative real numbers.
A time stamp is legal if will fire each event avoiding simultaneous execution of mutually exclusive events.
A time stamp is optimal if the execution time of the string is minimum.

Execution time

We now introduce the concept of the time in the system. When a string s is executed the events must be fired sequentially based on the order in the string. Therefore, each event $\sigma_k \in s$ has:

1. A starting time t_k;
2. An execution time $f(\sigma_k)$;
3. A completion time $t_k + f(\sigma_k)$.

A time stamp $\rho = [t_k \in \mathbb{R}_0^+ | k \in \{1, \cdots, |s|\}]$ of the string s with respect to G which establishes the starting time of each event is a nondecreasing list of nonnegative real numbers.

A time stamp is legal if will fire each event avoiding simultaneous execution of mutually exclusive events.

A time stamp is optimal if the execution time of the string is minimum.

Execution time

We now introduce the concept of the time in the system. When a string s is executed the events must be fired sequentially based on the order in the string. Therefore, each event $\sigma_k \in s$ has:

1. A starting time t_k;
2. An execution time $f(\sigma_k)$;
3. A completion time $t_k + f(\sigma_k)$.

A time stamp $\rho = [t_k \in \mathbb{R}_0^+ | k \in \{1, \cdots, |s|\}]$ of the string s with respect to G which establishes the starting time of each event is a nondecreasing list of nonnegative real numbers.
A time stamp is legal if will fire each event avoiding simultaneous execution of mutually exclusive events.
A time stamp is optimal if the execution time of the string is minimum.

Execution time

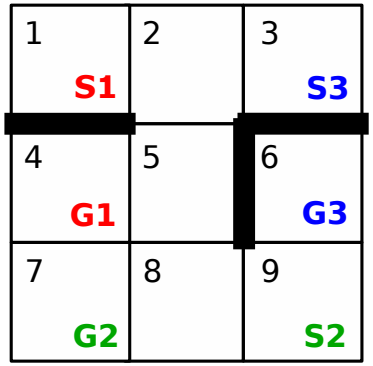
We now introduce the concept of the time in the system. When a string s is executed the events must be fired sequentially based on the order in the string. Therefore, each event $\sigma_k \in s$ has:

1. A starting time t_k;
2. An execution time $f(\sigma_k)$;
3. A completion time $t_k + f(\sigma_k)$.

A time stamp $\rho = [t_k \in \mathbb{R}_0^+ | k \in \{1, \cdots, |s|\}]$ of the string s with respect to G which establishes the starting time of each event is a nondecreasing list of nonnegative real numbers.
A time stamp is legal if will fire each event avoiding simultaneous execution of mutually exclusive events.
A time stamp is optimal if the execution time of the string is minimum.

Example

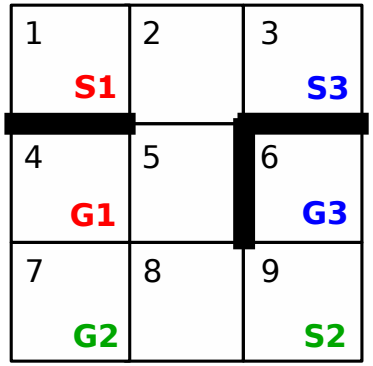
- All event time executions are equal to 1;
- A trajectory can be $t = \sigma_1 \cdots \sigma_{10} = (1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The optimal time stamp for trajectory t is $\rho = [0,1,2,2,3,3,4,5,6,7]$.
- Sets of mutually exclusive events $R = \{r_1 = \{\sigma_1, \sigma_2, \sigma_3\}, r_2 = \{\sigma_4, \sigma_5\}, r_3 = \{\sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}\}, r_4 = \{\sigma_1, \sigma_2, \sigma_6, \sigma_7\}, r_5 = \{\sigma_2, \sigma_3, \sigma_7, \sigma_8\}\}, r_6 = \{\sigma_4, \sigma_5, \sigma_8, \sigma_9\}, r_6 = \{\sigma_4, \sigma_9, \sigma_{10}\}\}$ it should be noted that r_1, r_2, r_3 contain all events of robots 1,2,3 while r_3, r_4, r_5, r_6 contain events across all robots of positions 2,5,8,9.



The execution time v of the trajectory t is $v(t) = 8$

Example

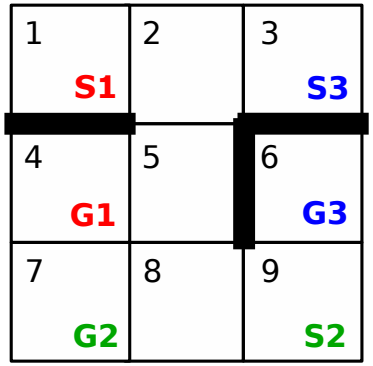
- All event time executions are equal to 1;
- A trajectory can be $t = \sigma_1 \cdots \sigma_{10} =$ $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The optimal time stamp for trajectory t is $\rho = [0,1,2,2,3,3,4,5,6,7]$.
- Sets of mutually exclusive events $R = \{r_1 = \{\sigma_1, \sigma_2, \sigma_3\}, r_2 = \{\sigma_4, \sigma_5\},$ $r_3 = \{\sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}\}, r_4 = \{\sigma_1, \sigma_2, \sigma_6, \sigma_7\},$ $r_5 = \{\sigma_2, \sigma_3, \sigma_7, \sigma_8\}\}, r_6 = \{\sigma_4, \sigma_5, \sigma_8, \sigma_9\},$ $r_6 = \{\sigma_4, \sigma_9, \sigma_{10}\}\}$ it should be noted that r_1, r_2, r_3 contain all events of robots 1,2,3 while r_3, r_4, r_5, r_6 contain events across all robots of positions 2,5,8,9.



The execution time v of the trajectory t is $v(t) = 8$

Example

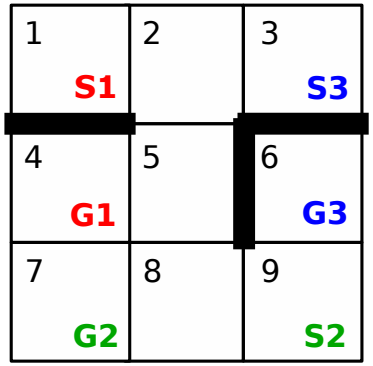
- All event time executions are equal to 1;
- A trajectory can be $t = \sigma_1 \cdots \sigma_{10} = (1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The optimal time stamp for trajectory t is $\rho = [0,1,2,2,3,3,4,5,6,7]$.
- Sets of mutually exclusive events $R = \{r_1 = \{\sigma_1, \sigma_2, \sigma_3\}, r_2 = \{\sigma_4, \sigma_5\}, r_3 = \{\sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}\}, r_4 = \{\sigma_1, \sigma_2, \sigma_6, \sigma_7\}, r_5 = \{\sigma_2, \sigma_3, \sigma_7, \sigma_8\}\}, r_6 = \{\sigma_4, \sigma_5, \sigma_8, \sigma_9\}, r_6 = \{\sigma_4, \sigma_9, \sigma_{10}\}\}$ it should be noted that r_1, r_2, r_3 contain all events of robots 1,2,3 while r_3, r_4, r_5, r_6 contain events across all robots of positions 2,5,8,9.



The execution time v of the trajectory t is $v(t) = 8$

Example

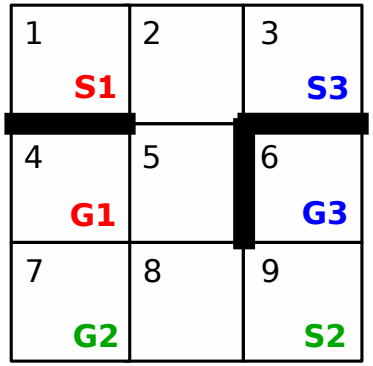
- All event time executions are equal to 1;
- A trajectory can be $t = \sigma_1 \cdots \sigma_{10} =$ $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The optimal time stamp for trajectory t is $\rho = [0,1,2,2,3,3,4,5,6,7]$.
- Sets of mutually exclusive events $R = \{r_1 = \{\sigma_1, \sigma_2, \sigma_3\}, r_2 = \{\sigma_4, \sigma_5\},$ $r_3 = \{\sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}\}, r_4 = \{\sigma_1, \sigma_2, \sigma_6, \sigma_7\},$ $r_5 = \{\sigma_2, \sigma_3, \sigma_7, \sigma_8\}\}, r_6 = \{\sigma_4, \sigma_5, \sigma_8, \sigma_9\},$ $r_6 = \{\sigma_4, \sigma_9, \sigma_{10}\}\}$ it should be noted that r_1, r_2, r_3 contain all events of robots 1,2,3 while r_3, r_4, r_5, r_6 contain events across all robots of positions 2,5,8,9.



The execution time v of the trajectory t is $v(t) = 8$

Example

- All event time executions are equal to 1;
- A trajectory can be $t = \sigma_1 \cdots \sigma_{10} =$ $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The optimal time stamp for trajectory t is $\rho = [0,1,2,2,3,3,4,5,6,7]$.
- Sets of mutually exclusive events $R = \{r_1 = \{\sigma_1, \sigma_2, \sigma_3\}, r_2 = \{\sigma_4, \sigma_5\},$ $r_3 = \{\sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}\}, r_4 = \{\sigma_1, \sigma_2, \sigma_6, \sigma_7\},$ $r_5 = \{\sigma_2, \sigma_3, \sigma_7, \sigma_8\}\}, r_6 = \{\sigma_4, \sigma_5, \sigma_8, \sigma_9\},$ $r_6 = \{\sigma_4, \sigma_9, \sigma_{10}\}\}$ it should be noted that r_1, r_2, r_3 contain all events of robots 1,2,3 while r_3, r_4, r_5, r_6 contain events across all robots of positions 2,5,8,9.



The execution time v of the trajectory t is $\boldsymbol{v(t) = 8}$

Mutual exclusiveness and resources

Definition 2

Two events σ_1, σ_2, with t_1, t_2 as their starting times and $f(\sigma_1), f(\sigma_2)$ as their execution times are mutually exclusive if it is not allowed that $t_1 \in [t_2, t_2 + f(\sigma_2)]$ or $t_2 \in [t_1, t_1 + f(\sigma_2)]$, i.e. if one event is under execution the other cannot be fired.

Definition 3

A resource $r = \{\sigma | \sigma \in \Sigma\}$ is a set of mutually exclusive events within the same alphabet Σ. □

Definition 4

A set of resources $R = \{r | r \subseteq \Sigma\}$ is a finite set of resources where for every pair of events $\sigma, \sigma' \in \Sigma$ there exists a resource $r \in R$ such that $\sigma, \sigma' \in r$ if and only if σ and σ' are mutually exclusive. □

Mutual exclusiveness and resources

Definition 2

Two events σ_1, σ_2, with t_1, t_2 as their starting times and $f(\sigma_1), f(\sigma_2)$ as their execution times are mutually exclusive if it is not allowed that $t_1 \in [t_2, t_2 + f(\sigma_2)]$ or $t_2 \in [t_1, t_1 + f(\sigma_2)]$, i.e. if one event is under execution the other cannot be fired.

Definition 3

A resource $r = \{\sigma | \sigma \in \Sigma\}$ is a set of mutually exclusive events within the same alphabet Σ. □

Definition 4

A set of resources $R = \{r | r \subseteq \Sigma\}$ is a finite set of resources where for every pair of events $\sigma, \sigma' \in \Sigma$ there exists a resource $r \in R$ such that $\sigma, \sigma' \in r$ if and only if σ and σ' are mutually exclusive. □

Mutual exclusiveness and resources

Definition 2

Two events σ_1, σ_2, with t_1, t_2 as their starting times and $f(\sigma_1), f(\sigma_2)$ as their execution times are mutually exclusive if it is not allowed that $t_1 \in [t_2, t_2 + f(\sigma_2)]$ or $t_2 \in [t_1, t_1 + f(\sigma_2)]$, i.e. if one event is under execution the other cannot be fired.

Definition 3

A resource $r = \{\sigma | \sigma \in \Sigma\}$ is a set of mutually exclusive events within the same alphabet Σ. □

Definition 4

A set of resources $R = \{r | r \subseteq \Sigma\}$ is a finite set of resources where for every pair of events $\sigma, \sigma' \in \Sigma$ there exists a resource $r \in R$ such that $\sigma, \sigma' \in r$ if and only if σ and σ' are mutually exclusive. □

Problem as a timed automaton

We define a novel kind of timed automaton by coupling a positive real number, the execution time, with each event in the alphabet of the composite automaton G and including the concept of mutual exclusive events in the definition.
We construct the timed automaton $\mathcal{G} = (G, f, R)$ with

- $G = G_1 \times \cdots G_n$ is the product of the n robot automata in the environment;
- $f : \Sigma = \Sigma_1 \cup \cdots \Sigma_n \to \mathbb{R}$ is a weight function which assigns a weight to all events equal to the execution time of the event;
- R is a set of resources such that it contains:
  1. **Robot-resources**: one for each robot such that all events of the robot is in that resource.
  2. **Position-resources**: one for each position such that all events (across all robots) which involves the position is in that resource.

Problem as a timed automaton

We define a novel kind of timed automaton by coupling a positive real number, the execution time, with each event in the alphabet of the composite automaton G and including the concept of mutual exclusive events in the definition.
We construct the timed automaton $\mathcal{G} = (G, f, R)$ with

- $G = G_1 \times \cdots G_n$ is the product of the n robot automata in the environment;
- $f : \Sigma = \Sigma_1 \cup \cdots \Sigma_n \to \mathbb{R}$ is a weight function which assigns a weight to all events equal to the execution time of the event;
- R is a set of resources such that it contains:
  1. **Robot-resources**: one for each robot such that all events of the robot is in that resource.
  2. **Position-resources**: one for each position such that all events (across all robots) which involves the position is in that resource.

Problem as a timed automaton

We define a novel kind of timed automaton by coupling a positive real number, the execution time, with each event in the alphabet of the composite automaton G and including the concept of mutual exclusive events in the definition.
We construct the timed automaton $\mathcal{G} = (G, f, R)$ with

- $G = G_1 \times \cdots G_n$ is the product of the n robot automata in the environment;
- $f : \Sigma = \Sigma_1 \cup \cdots \Sigma_n \rightarrow \mathbb{R}$ is a weight function which assigns a weight to all events equal to the execution time of the event;
- R is a set of resources such that it contains:
  1. **Robot-resources**: one for each robot such that all events of the robot is in that resource.
  2. **Position-resources**: one for each position such that all events (across all robots) which involves the position is in that resource.

Problem as a timed automaton

We define a novel kind of timed automaton by coupling a positive real number, the execution time, with each event in the alphabet of the composite automaton G and including the concept of mutual exclusive events in the definition.
We construct the timed automaton $\mathcal{G} = (G, f, R)$ with

- $G = G_1 \times \cdots G_n$ is the product of the n robot automata in the environment;
- $f : \Sigma = \Sigma_1 \cup \cdots \Sigma_n \to \mathbb{R}$ is a weight function which assigns a weight to all events equal to the execution time of the event;
- R is a set of resources such that it contains:
  1. **Robot-resources**: one for each robot such that all events of the robot is in that resource.
  2. **Position-resources**: one for each position such that all events (across all robots) which involves the position is in that resource.

Problem as a timed automaton

We define a novel kind of timed automaton by coupling a positive real number, the execution time, with each event in the alphabet of the composite automaton G and including the concept of mutual exclusive events in the definition.
We construct the timed automaton $\mathcal{G} = (G, f, R)$ with

- $G = G_1 \times \cdots G_n$ is the product of the n robot automata in the environment;
- $f : \Sigma = \Sigma_1 \cup \cdots \Sigma_n \rightarrow \mathbb{R}$ is a weight function which assigns a weight to all events equal to the execution time of the event;
- R is a set of resources such that it contains:
  1. **Robot-resources**: one for each robot such that all events of the robot is in that resource.
  2. **Position-resources**: one for each position such that all events (across all robots) which involves the position is in that resource.

Problem as a timed automaton

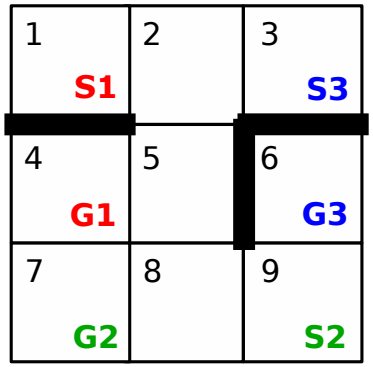
We define a novel kind of timed automaton by coupling a positive real number, the execution time, with each event in the alphabet of the composite automaton G and including the concept of mutual exclusive events in the definition.
We construct the timed automaton $\mathcal{G} = (G, f, R)$ with

- $G = G_1 \times \cdots G_n$ is the product of the n robot automata in the environment;
- $f : \Sigma = \Sigma_1 \cup \cdots \Sigma_n \to \mathbb{R}$ is a weight function which assigns a weight to all events equal to the execution time of the event;
- R is a set of resources such that it contains:
  1. **Robot-resources**: one for each robot such that all events of the robot is in that resource.
  2. **Position-resources**: one for each position such that all events (across all robots) which involves the position is in that resource.

Outline

Rearranging events - Preserving Priorities

- Preserving priorities in rearranging events within a trajectory means that if a resource is used first by event e_1 and then from e_2 it is not allowed to move e_2 before e_1. Let's see an example:
- All event time executions are equal to 1;
- The input trajectory is $t \in L_m(G)$ such that $t_p = \sigma_1 \cdots \sigma_{10} = (1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_p \in L_m(G)$ such that $t_p = (1,1,2)\mathbf{(2,9,8)}(1,2,5)\mathbf{(2,8,7)}(1,5,4)(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$
- The optimal time stamp for trajectory t_p is $\rho = [0,0,1,1,2,2,3,4,5,6]$.

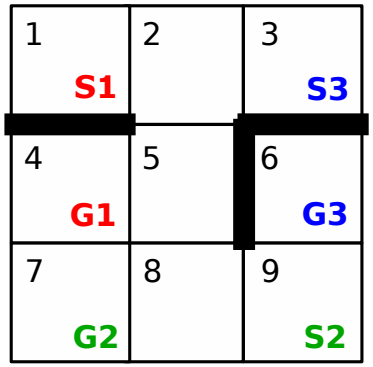


The execution times v of trajectories t, t_p are

$$v(t) = 8$$
$$v(t_p) = 7$$

Rearranging events - Preserving Priorities

- Preserving priorities in rearranging events within a trajectory means that if a resource is used first by event e_1 and then from e_2 it is not allowed to move e_2 before e_1. Let's see an example:
- All event time executions are equal to 1;
- The input trajectory is $t \in L_m(G)$ such that $t_p = \sigma_1 \cdots \sigma_{10} = (1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_p \in L_m(G)$ such that $t_p = (1,1,2)\mathbf{(2,9,8)}(1,2,5)\mathbf{(2,8,7)}(1,5,4)(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$
- The optimal time stamp for trajectory t_p is $\rho = [0,0,1,1,2,2,3,4,5,6]$.

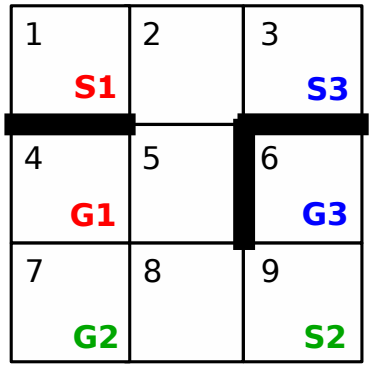


The execution times v of trajectories t, t_p are

$$v(t) = 8$$
$$v(t_p) = 7$$

Rearranging events - Preserving Priorities

- Preserving priorities in rearranging events within a trajectory means that if a resource is used first by event e_1 and then from e_2 it is not allowed to move e_2 before e_1. Let's see an example:
- All event time executions are equal to 1;
- The input trajectory is $t \in L_m(G)$ such that $t_p = \sigma_1 \cdots \sigma_{10} =$ $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_p \in L_m(G)$ such that $t_p = (1,1,2)\mathbf{(2,9,8)}(1,2,5)\mathbf{(2,8,7)}(1,5,4)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$
- The optimal time stamp for trajectory t_p is $\rho = [0,0,1,1,2,2,3,4,5,6]$.



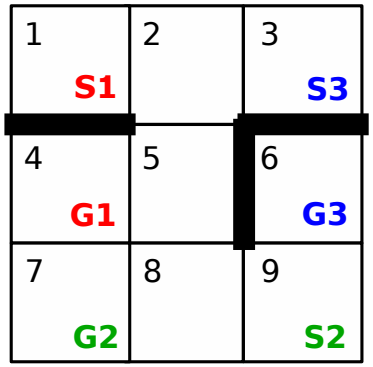
The execution times v of trajectories t, t_p are

$v(t) = 8$

$v(t_p) = 7$

Rearranging events - Preserving Priorities

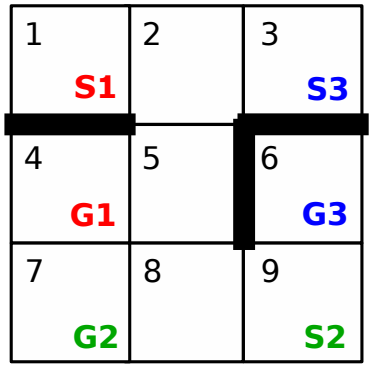
- Preserving priorities in rearranging events within a trajectory means that if a resource is used first by event e_1 and then from e_2 it is not allowed to move e_2 before e_1. Let's see an example:
- All event time executions are equal to 1;
- The input trajectory is $t \in L_m(G)$ such that
  $t_p = \sigma_1 \cdots \sigma_{10} =$
  $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$
  $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_p \in L_m(G)$ such that
  $t_p = (1,1,2)\mathbf{(2,9,8)}(1,2,5)\mathbf{(2,8,7)}(1,5,4)$
  $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$
- The optimal time stamp for trajectory t_p is
  $\rho = [0,0,1,1,2,2,3,4,5,6]$.



The execution times v of trajectories t, t_p are
$v(t) = 8$
$v(t_p) = 7$

Rearranging events - Preserving Priorities

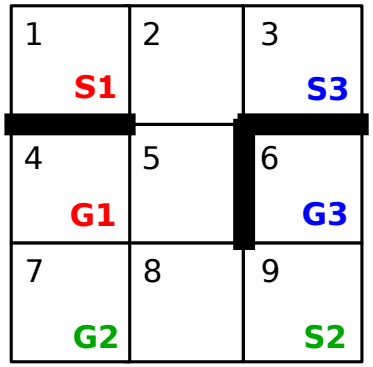
- Preserving priorities in rearranging events within a trajectory means that if a resource is used first by event e_1 and then from e_2 it is not allowed to move e_2 before e_1. Let's see an example:
- All event time executions are equal to 1;
- The input trajectory is $t \in L_m(G)$ such that
  $t_p = \sigma_1 \cdots \sigma_{10} =$
  $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$
  $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_p \in L_m(G)$ such that
  $t_p = (1,1,2)\mathbf{(2,9,8)}(1,2,5)\mathbf{(2,8,7)}(1,5,4)$
  $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$
- The optimal time stamp for trajectory t_p is
  $\rho = [0,0,1,1,2,2,3,4,5,6]$.



The execution times v of trajectories t, t_p are
$v(t) = 8$
$v(t_p) = 7$

Rearranging events - Preserving Priorities

- Preserving priorities in rearranging events within a trajectory means that if a resource is used first by event e_1 and then from e_2 it is not allowed to move e_2 before e_1. Let's see an example:
- All event time executions are equal to 1;
- The input trajectory is $t \in L_m(G)$ such that $t_p = \sigma_1 \cdots \sigma_{10} =$ $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_p \in L_m(G)$ such that $t_p = (1,1,2)\mathbf{(2,9,8)}(1,2,5)\mathbf{(2,8,7)}(1,5,4)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$
- The optimal time stamp for trajectory t_p is $\rho = [0,0,1,1,2,2,3,4,5,6]$.

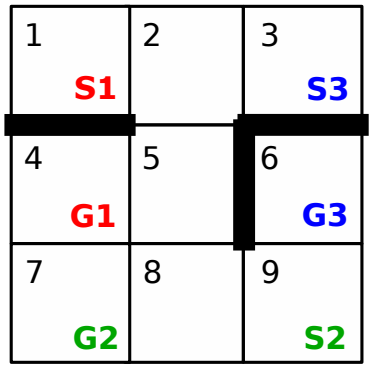


The execution times v of trajectories t, t_p are

$$\boldsymbol{v(t) = 8}$$
$$\boldsymbol{v(t_p) = 7}$$

Rearranging events - Changing Priorities

- Changing priorities in rearranging events within a trajectory means that if a resource is used first by event e_1 and then from e_2 it is allowed to move e_2 before e_1. Let's see an example:
- All event time executions are equal to 1;
- The input trajectory is $t = \sigma_1 \cdots \sigma_{10} =$ $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_c =$ $\mathbf{(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)}$ $(1,1,2)(1,2,5)(1,5,4)(3,8,9)(3,9,6)$
- The optimal time stamp for trajectory t_c is $\rho = [0,1,1,2,3,3,4,5,6,7]$.

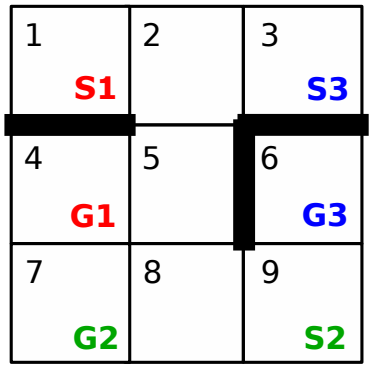


The execution times v of trajectories t, t_c are

$$v(t) = 8$$
$$v(t_c) = 8$$

Rearranging events - Changing Priorities

- Changing priorities in rearranging events within a trajectory means that if a resource is used first by event e_1 and then from e_2 it is allowed to move e_2 before e_1. Let's see an example:
- All event time executions are equal to 1;
- The input trajectory is $t = \sigma_1 \cdots \sigma_{10} =$ $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_c =$ $\mathbf{(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)}$ $(1,1,2)(1,2,5)(1,5,4)(3,8,9)(3,9,6)$
- The optimal time stamp for trajectory t_c is $\rho = [0,1,1,2,3,3,4,5,6,7]$.

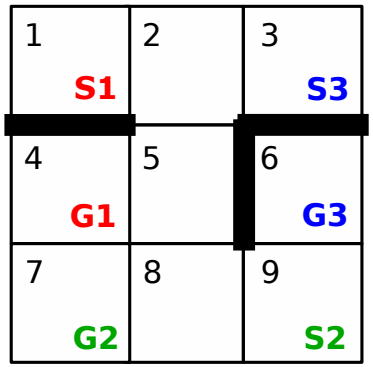


The execution times v of trajectories t, t_c are

$$v(t) = 8$$
$$v(t_c) = 8$$

Rearranging events - Changing Priorities

- Changing priorities in rearranging events within a trajectory means that if a resource is used first by event e_1 and then from e_2 it is allowed to move e_2 before e_1. Let's see an example:
- All event time executions are equal to 1;
- The input trajectory is $t = \sigma_1 \cdots \sigma_{10} = (1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_c = (2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)(1,1,2)(1,2,5)(1,5,4)(3,8,9)(3,9,6)$
- The optimal time stamp for trajectory t_c is $\rho = [0,1,1,2,3,3,4,5,6,7]$.

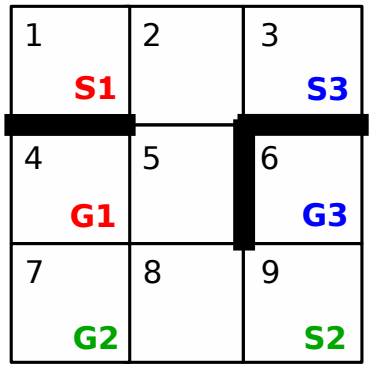


The execution times v of trajectories t, t_c are

$$v(t) = 8$$
$$v(t_c) = 8$$

Rearranging events - Changing Priorities

- Changing priorities in rearranging events within a trajectory means that if a resource is used first by event e_1 and then from e_2 it is allowed to move e_2 before e_1. Let's see an example:
- All event time executions are equal to 1;
- The input trajectory is $t = \sigma_1 \cdots \sigma_{10} =$ $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_c =$ $\mathbf{(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)}$ $(1,1,2)(1,2,5)(1,5,4)(3,8,9)(3,9,6)$
- The optimal time stamp for trajectory t_c is $\rho = [0,1,1,2,3,3,4,5,6,7]$.



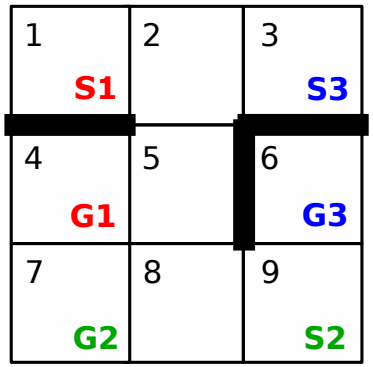
The execution times v of trajectories t, t_c are

$v(t) = 8$

$v(t_c) = 8$

Rearranging events - Changing Priorities

- Changing priorities in rearranging events within a trajectory means that if a resource is used first by event e_1 and then from e_2 it is allowed to move e_2 before e_1. Let's see an example:
- All event time executions are equal to 1;
- The input trajectory is $t = \sigma_1 \cdots \sigma_{10} =$ $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_c =$ $\mathbf{(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)}$ $(1,1,2)(1,2,5)(1,5,4)(3,8,9)(3,9,6)$
- The optimal time stamp for trajectory t_c is $\rho = [0,1,1,2,3,3,4,5,6,7]$.



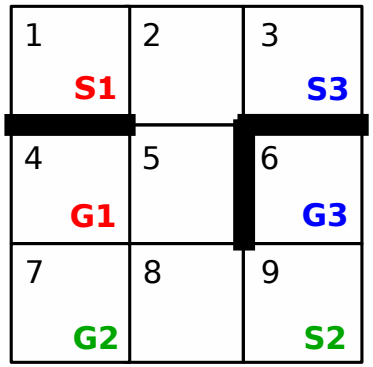
The execution times v of trajectories t, t_c are

$v(t) = 8$

$v(t_c) = 8$

Rearranging events - Changing Priorities

- Changing priorities in rearranging events within a trajectory means that if a resource is used first by event e_1 and then from e_2 it is allowed to move e_2 before e_1. Let's see an example:
- All event time executions are equal to 1;
- The input trajectory is $t = \sigma_1 \cdots \sigma_{10} =$ $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_c =$ $\mathbf{(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)}$ $(1,1,2)(1,2,5)(1,5,4)(3,8,9)(3,9,6)$
- The optimal time stamp for trajectory t_c is $\rho = [0,1,1,2,3,3,4,5,6,7]$.

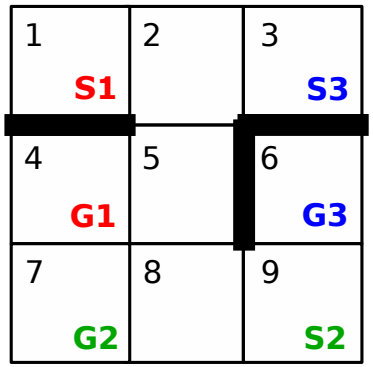


The execution times v of trajectories t, t_c are

$$\boldsymbol{v(t) = 8}$$
$$\boldsymbol{v(t_c) = 8}$$

Rearranging events - Proposed approach

- Now let's see the improvement obtained with the proposed approach which merges preserving an changing priorities techniques:
- All event time executions are equal to 1;
- The input trajectory is $t = \sigma_1 \cdots \sigma_{10} = (1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_m = \mathbf{(2,9,8)(3,3,2)(2,8,7)(3,2,5)(1,1,2)(3,5,8)(1,2,5)(3,8,9)(1,5,4)(3,9,6)}$
- The optimal time stamp for trajectory t_c is $\rho = [0,0,1,1,2,2,3,3,4,4]$.

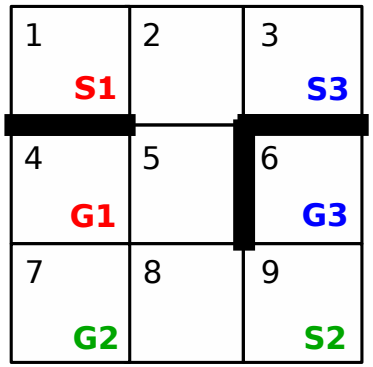


The execution times v of trajectories t, t_m are

$$v(t) = 8$$
$$v(t_m) = 5$$

Rearranging events - Proposed approach

- Now let's see the improvement obtained with the proposed approach which merges preserving an changing priorities techniques:
- All event time executions are equal to 1;
- The input trajectory is $t = \sigma_1 \cdots \sigma_{10} = (1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_m = (2,9,8)(3,3,2)(2,8,7)(3,2,5)(1,1,2)(3,5,8)(1,2,5)(3,8,9)(1,5,4)(3,9,6)$
- The optimal time stamp for trajectory t_c is $\rho = [0,0,1,1,2,2,3,3,4,4]$.

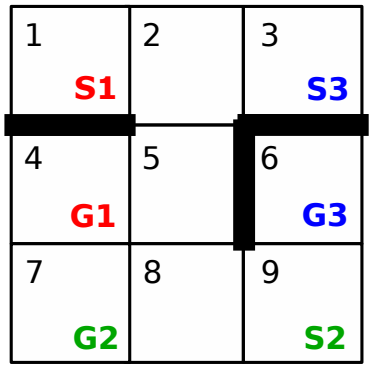


The execution times v of trajectories t, t_m are

$$v(t) = 8$$
$$v(t_m) = 5$$

Rearranging events - Proposed approach

- Now let's see the improvement obtained with the proposed approach which merges preserving an changing priorities techniques:
- All event time executions are equal to 1;
- The input trajectory is $t = \sigma_1 \cdots \sigma_{10} = (1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_m = (2,9,8)(3,3,2)(2,8,7)(3,2,5)(1,1,2)(3,5,8)(1,2,5)(3,8,9)(1,5,4)(3,9,6)$
- The optimal time stamp for trajectory t_c is $\rho = [0,0,1,1,2,2,3,3,4,4]$.



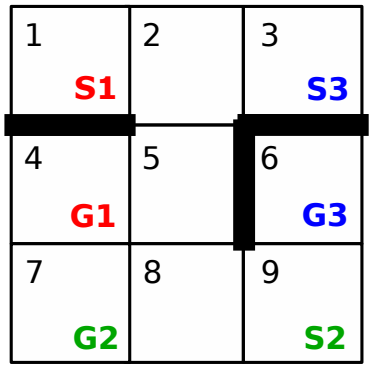
The execution times v of trajectories t, t_m are

$v(t) = 8$

$v(t_m) = 5$

Rearranging events - Proposed approach

- Now let's see the improvement obtained with the proposed approach which merges preserving an changing priorities techniques:
- All event time executions are equal to 1;
- The input trajectory is $t = \sigma_1 \cdots \sigma_{10} =$ $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_m =$ $\mathbf{(2,9,8)(3,3,2)(2,8,7)(3,2,5)(1,1,2)}$ $\mathbf{(3,5,8)(1,2,5)(3,8,9)(1,5,4)(3,9,6)}$
- The optimal time stamp for trajectory t_c is $\rho = [0,0,1,1,2,2,3,3,4,4]$.



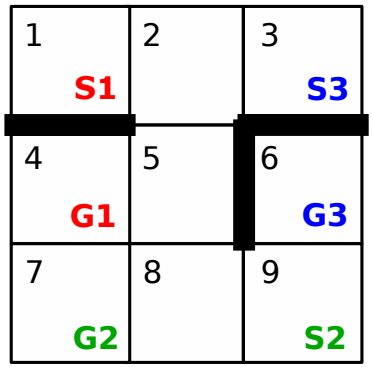
The execution times v of trajectories t, t_m are

$v(t) = 8$

$v(t_m) = 5$

Rearranging events - Proposed approach

- Now let's see the improvement obtained with the proposed approach which merges preserving an changing priorities techniques:
- All event time executions are equal to 1;
- The input trajectory is $t = \sigma_1 \cdots \sigma_{10} =$ $(1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)$ $(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_m =$ $\mathbf{(2,9,8)(3,3,2)(2,8,7)(3,2,5)(1,1,2)}$ $\mathbf{(3,5,8)(1,2,5)(3,8,9)(1,5,4)(3,9,6)}$
- The optimal time stamp for trajectory t_c is $\rho = [0,0,1,1,2,2,3,3,4,4]$.



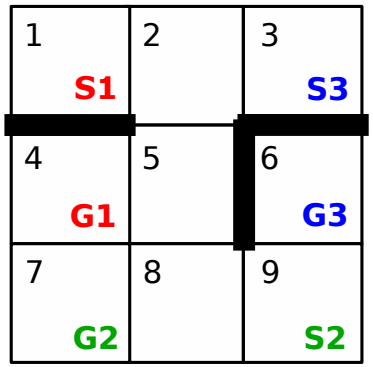
The execution times v of trajectories t, t_m are

$v(t) = 8$

$v(t_m) = 5$

Rearranging events - Proposed approach

- Now let's see the improvement obtained with the proposed approach which merges preserving an changing priorities techniques:
- All event time executions are equal to 1;
- The input trajectory is $t = \sigma_1 \cdots \sigma_{10} = (1,1,2)(1,2,5)(1,5,4)(2,9,8)(2,8,7)(3,3,2)(3,2,5)(3,5,8)(3,8,9)(3,9,6)$;
- The output trajectory is $t_m = \mathbf{(2,9,8)(3,3,2)(2,8,7)(3,2,5)(1,1,2)(3,5,8)(1,2,5)(3,8,9)(1,5,4)(3,9,6)}$
- The optimal time stamp for trajectory t_c is $\rho = [0,0,1,1,2,2,3,3,4,4]$.



The execution times v of trajectories t, t_m are

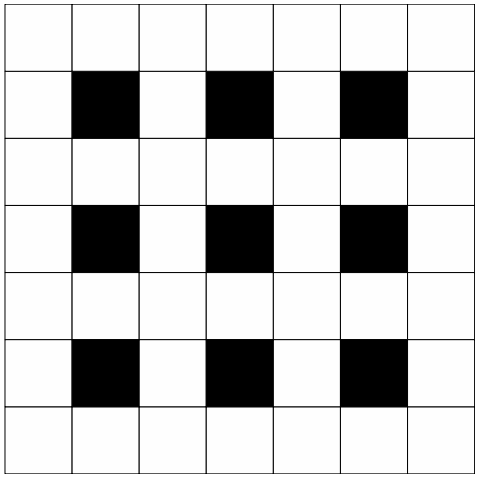
$$\boldsymbol{v(t) = 8}$$

$$\boldsymbol{v(t_m) = 5}$$

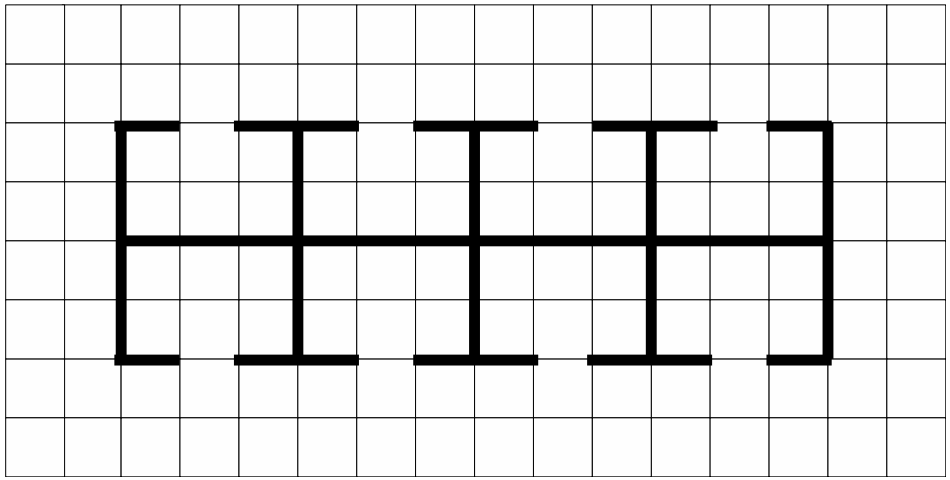
Outline

Evnvironments

Grid



Cyclic Corridor



Test description

- Random start and goal position from amongst all the possible free location in the environment, such that:
  1. Two robot can not share the same start position;
  2. Two robot can not share the same goal position;
  3. A robot can have another robot's start position as its goal position;
  4. A robot can have the same start and goal position.
- The grid was also expanded to dimensions 11 and 15.
- For each environment we run 100 different experiments and made the average of all results.
- For each experiment we calculated a lower bound (LB) for the minimum execution time of the optimal solution in the following manner:
  1. Calculate the shortest path for each individual robot assuming that there are no other robots to get in its way;
  2. Find the robot with the longest journey and select it's duration as our LB.

Test description

- Random start and goal position from amongst all the possible free location in the environment, such that:
  1. Two robot can not share the same start position;
  2. Two robot can not share the same goal position;
  3. A robot can have another robot's start position as its goal position;
  4. A robot can have the same start and goal position.
- The grid was also expanded to dimensions 11 and 15.
- For each environment we run 100 different experiments and made the average of all results.
- For each experiment we calculated a lower bound (LB) for the minimum execution time of the optimal solution in the following manner:
  1. Calculate the shortest path for each individual robot assuming that there are no other robots to get in its way;
  2. Find the robot with the longest journey and select it's duration as our LB.

Test description

- Random start and goal position from amongst all the possible free location in the environment, such that:
  1. Two robot can not share the same start position;
  2. Two robot can not share the same goal position;
  3. A robot can have another robot's start position as its goal position;
  4. A robot can have the same start and goal position.
- The grid was also expanded to dimensions 11 and 15.
- For each environment we run 100 different experiments and made the average of all results.
- For each experiment we calculated a lower bound (LB) for the minimum execution time of the optimal solution in the following manner:
  1. Calculate the shortest path for each individual robot assuming that there are no other robots to get in its way;
  2. Find the robot with the longest journey and select it's duration as our LB.

Test description

- Random start and goal position from amongst all the possible free location in the environment, such that:
  1. Two robot can not share the same start position;
  2. Two robot can not share the same goal position;
  3. A robot can have another robot's start position as its goal position;
  4. A robot can have the same start and goal position.
- The grid was also expanded to dimensions 11 and 15.
- For each environment we run 100 different experiments and made the average of all results.
- For each experiment we calculated a lower bound (LB) for the minimum execution time of the optimal solution in the following manner:
  1. Calculate the shortest path for each individual robot assuming that there are no other robots to get in its way;
  2. Find the robot with the longest journey and select it's duration as our LB.

Test description

- Random start and goal position from amongst all the possible free location in the environment, such that:
  1. Two robot can not share the same start position;
  2. Two robot can not share the same goal position;
  3. A robot can have another robot's start position as its goal position;
  4. A robot can have the same start and goal position.
- The grid was also expanded to dimensions 11 and 15.
- For each environment we run 100 different experiments and made the average of all results.
- For each experiment we calculated a lower bound (LB) for the minimum execution time of the optimal solution in the following manner:
  1. Calculate the shortest path for each individual robot assuming that there are no other robots to get in its way;
  2. Find the robot with the longest journey and select it's duration as our LB.

Test description

- Random start and goal position from amongst all the possible free location in the environment, such that:
  1. Two robot can not share the same start position;
  2. Two robot can not share the same goal position;
  3. A robot can have another robot's start position as its goal position;
  4. A robot can have the same start and goal position.
- The grid was also expanded to dimensions 11 and 15.
- For each environment we run 100 different experiments and made the average of all results.
- For each experiment we calculated a lower bound (LB) for the minimum execution time of the optimal solution in the following manner:
  1. Calculate the shortest path for each individual robot assuming that there are no other robots to get in its way;
  2. Find the robot with the longest journey and select it's duration as our LB.

Test description

- Random start and goal position from amongst all the possible free location in the environment, such that:
  1. Two robot can not share the same start position;
  2. Two robot can not share the same goal position;
  3. A robot can have another robot's start position as its goal position;
  4. A robot can have the same start and goal position.
- The grid was also expanded to dimensions 11 and 15.
- For each environment we run 100 different experiments and made the average of all results.
- For each experiment we calculated a lower bound (LB) for the minimum execution time of the optimal solution in the following manner:
  1. Calculate the shortest path for each individual robot assuming that there are no other robots to get in its way;
  2. Find the robot with the longest journey and select it's duration as our LB.

Test description

- Random start and goal position from amongst all the possible free location in the environment, such that:
  1. Two robot can not share the same start position;
  2. Two robot can not share the same goal position;
  3. A robot can have another robot's start position as its goal position;
  4. A robot can have the same start and goal position.
- The grid was also expanded to dimensions 11 and 15.
- For each environment we run 100 different experiments and made the average of all results.
- For each experiment we calculated a lower bound (LB) for the minimum execution time of the optimal solution in the following manner:
  1. Calculate the shortest path for each individual robot assuming that there are no other robots to get in its way;
  2. Find the robot with the longest journey and select it's duration as our LB.

Test description

- Random start and goal position from amongst all the possible free location in the environment, such that:
  1. Two robot can not share the same start position;
  2. Two robot can not share the same goal position;
  3. A robot can have another robot's start position as its goal position;
  4. A robot can have the same start and goal position.
- The grid was also expanded to dimensions 11 and 15.
- For each environment we run 100 different experiments and made the average of all results.
- For each experiment we calculated a lower bound (LB) for the minimum execution time of the optimal solution in the following manner:
  1. Calculate the shortest path for each individual robot assuming that there are no other robots to get in its way;
  2. Find the robot with the longest journey and select it's duration as our LB.

Test description

For each experiment four types of test were carried out, for purposes of presentation we will refer to these cases by an acronym:

1. First we found a solution by applying an optimization algorithm based on priorities preservation previously developed from our group: we refer to this case by acronym "c" which stands for "Compress".
2. Second we applied the novel algorithm proposed in this work to the result of tests c; we refer to this case by acronym "se" which stands for "Swap at the End".
3. Third we found a new solution by applying the novel algorithm while computing the trajectory: we refer to this case by acronym "sw" which stands for "Swap While".
4. Forth we considered when at least one of succeeded tests se and sw and we took the best result; we refer to this case by acronym "bs" which stands for "Best Swap".

Test description

For each experiment four types of test were carried out, for purposes of presentation we will refer to these cases by an acronym:

1. First we found a solution by applying an optimization algorithm based on priorities preservation previously developed from our group: we refer to this case by acronym "c" which stands for "Compress".
2. Second we applied the novel algorithm proposed in this work to the result of tests c; we refer to this case by acronym "se" which stands for "Swap at the End".
3. Third we found a new solution by applying the novel algorithm while computing the trajectory: we refer to this case by acronym "sw" which stands for "Swap While".
4. Forth we considered when at least one of succeeded tests se and sw and we took the best result; we refer to this case by acronym "bs" which stands for "Best Swap".

Test description

For each experiment four types of test were carried out, for purposes of presentation we will refer to these cases by an acronym:

1. First we found a solution by applying an optimization algorithm based on priorities preservation previously developed from our group: we refer to this case by acronym "c" which stands for "Compress".
2. Second we applied the novel algorithm proposed in this work to the result of tests c; we refer to this case by acronym "se" which stands for "Swap at the End".
3. Third we found a new solution by applying the novel algorithm while computing the trajectory: we refer to this case by acronym "sw" which stands for "Swap While".
4. Forth we considered when at least one of succeeded tests se and sw and we took the best result; we refer to this case by acronym "bs" which stands for "Best Swap".

Test description

For each experiment four types of test were carried out, for purposes of presentation we will refer to these cases by an acronym:

1. First we found a solution by applying an optimization algorithm based on priorities preservation previously developed from our group: we refer to this case by acronym "c" which stands for "Compress".
2. Second we applied the novel algorithm proposed in this work to the result of tests c; we refer to this case by acronym "se" which stands for "Swap at the End".
3. Third we found a new solution by applying the novel algorithm while computing the trajectory: we refer to this case by acronym "sw" which stands for "Swap While".
4. Forth we considered when at least one of succeeded tests se and sw and we took the best result; we refer to this case by acronym "bs" which stands for "Best Swap".

Test description

For each experiment four types of test were carried out, for purposes of presentation we will refer to these cases by an acronym:

1. First we found a solution by applying an optimization algorithm based on priorities preservation previously developed from our group: we refer to this case by acronym "c" which stands for "Compress".
2. Second we applied the novel algorithm proposed in this work to the result of tests c; we refer to this case by acronym "se" which stands for "Swap at the End".
3. Third we found a new solution by applying the novel algorithm while computing the trajectory: we refer to this case by acronym "sw" which stands for "Swap While".
4. Forth we considered when at least one of succeeded tests se and sw and we took the best result; we refer to this case by acronym "bs" which stands for "Best Swap".

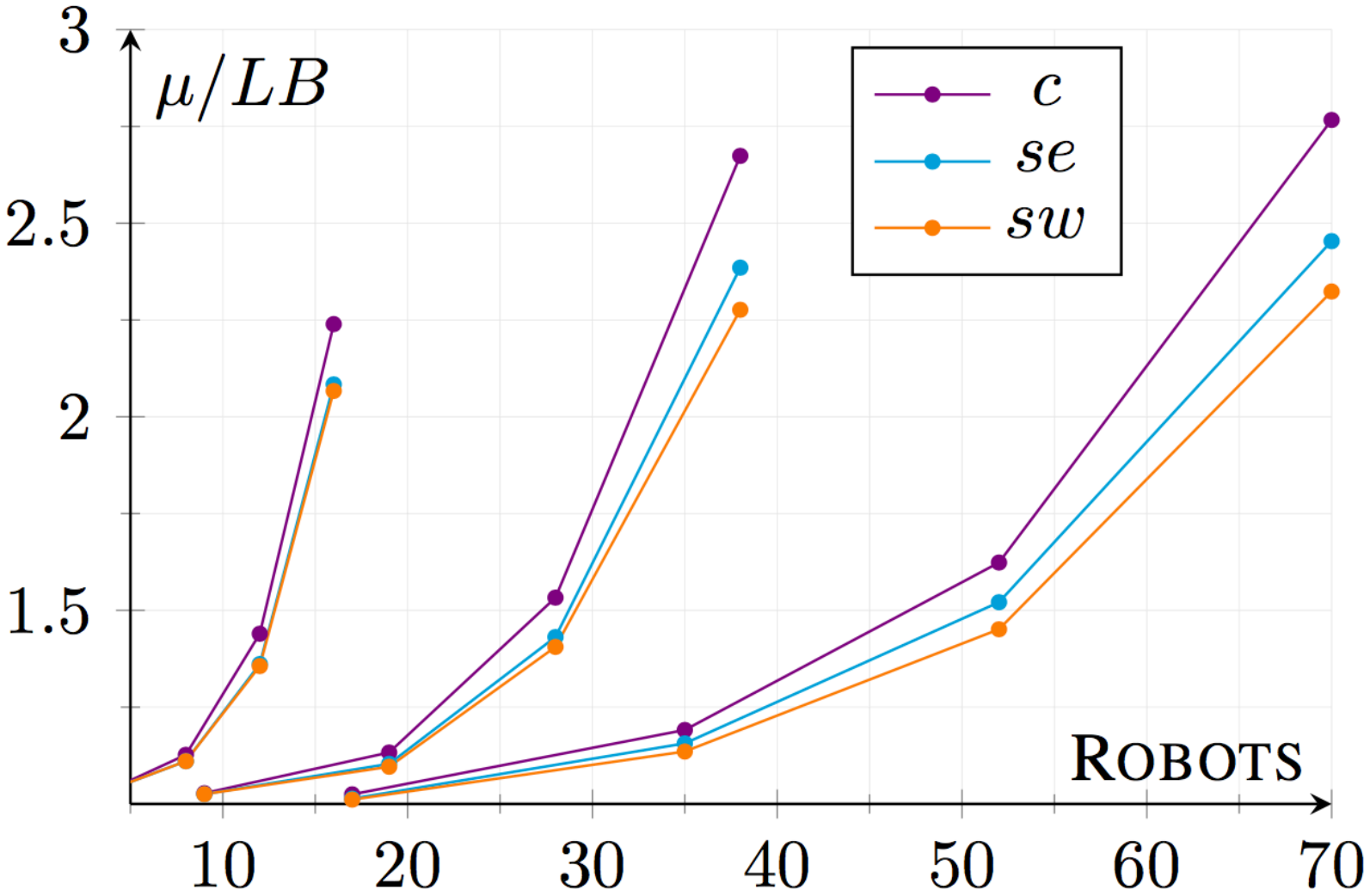
Grid results

	Unit	Grid 7x7				Grid 11x11				Grid 15x15			
ROBOTS		4	8	12	16	9	19	28	38	17	35	52	70
CONGESTION	%	10	20	30	40	10	20	30	40	10	20	30	40
$S_c \equiv S_{se}$	%	100	100	100	91	100	100	100	73	100	100	98	38
S_{sw}	%	100	100	100	91	100	100	100	74	100	100	98	40
S_{bs}	%	100	100	100	94	100	100	100	85	100	100	98	51
T_c	s	0.05	0.26	0.71	1.26	0.5	3.46	10.28	17.77	2.88	23.51	69.65	103.81
T_{se}	s	0.06	0.29	0.75	1.29	0.61	3.76	10.71	17.91	3.6	25.49	71.74	104.25
T_{sw}	s	0.07	0.36	0.94	1.61	0.9	5.8	15.54	26.03	7.62	51.23	125.23	179.5
T_{bs}	s	0.13	0.65	1.69	2.89	1.5	9.56	26.25	43.94	11.22	76.72	196.97	283.75
μ_c		7.89	9.57	12.82	20.8	13.55	16.94	23.84	42.49	20.06	25.04	35.66	63.03
μ_{se}		7.89	9.44	12.12	19.32	13.52	16.51	22.27	37.9	19.86	24.32	33.43	55.92
μ_{sw}		7.89	9.44	12.07	18.98	13.52	16.41	21.89	36.34	19.82	23.91	31.91	51.85
μ_{bs}		7.89	9.44	12.04	18.94	13.52	16.4	21.66	35.99	19.82	23.79	31.41	52.55
μ_c/LBE		1.04	1.13	1.44	2.24	1.03	1.13	1.53	2.67	1.02	1.19	1.62	2.77
μ_{se}/LBE		1.04	1.11	1.36	2.08	1.03	1.1	1.43	2.39	1.01	1.16	1.52	2.45
μ_{sw}/LBE		1.04	1.11	1.36	2.07	1.03	1.1	1.41	2.28	1.01	1.14	1.45	2.32
μ_{bs}/LBE		1.04	1.11	1.35	2.05	1.03	1.1	1.39	2.26	1.01	1.13	1.43	2.33
I_{se}	%	0	12.98	17.83	12.56	8.95	22.13	19.08	17.25	45.01	18.15	16.42	17.71
I_{sw}	%	0	12.98	19.01	13.92	8.95	27.71	23.86	23.75	54.46	29.11	27.63	25.09
I_{bs}	%	0	12.98	19.91	15.23	8.95	28.18	26.72	24.9	54.46	31.91	31.29	24.58

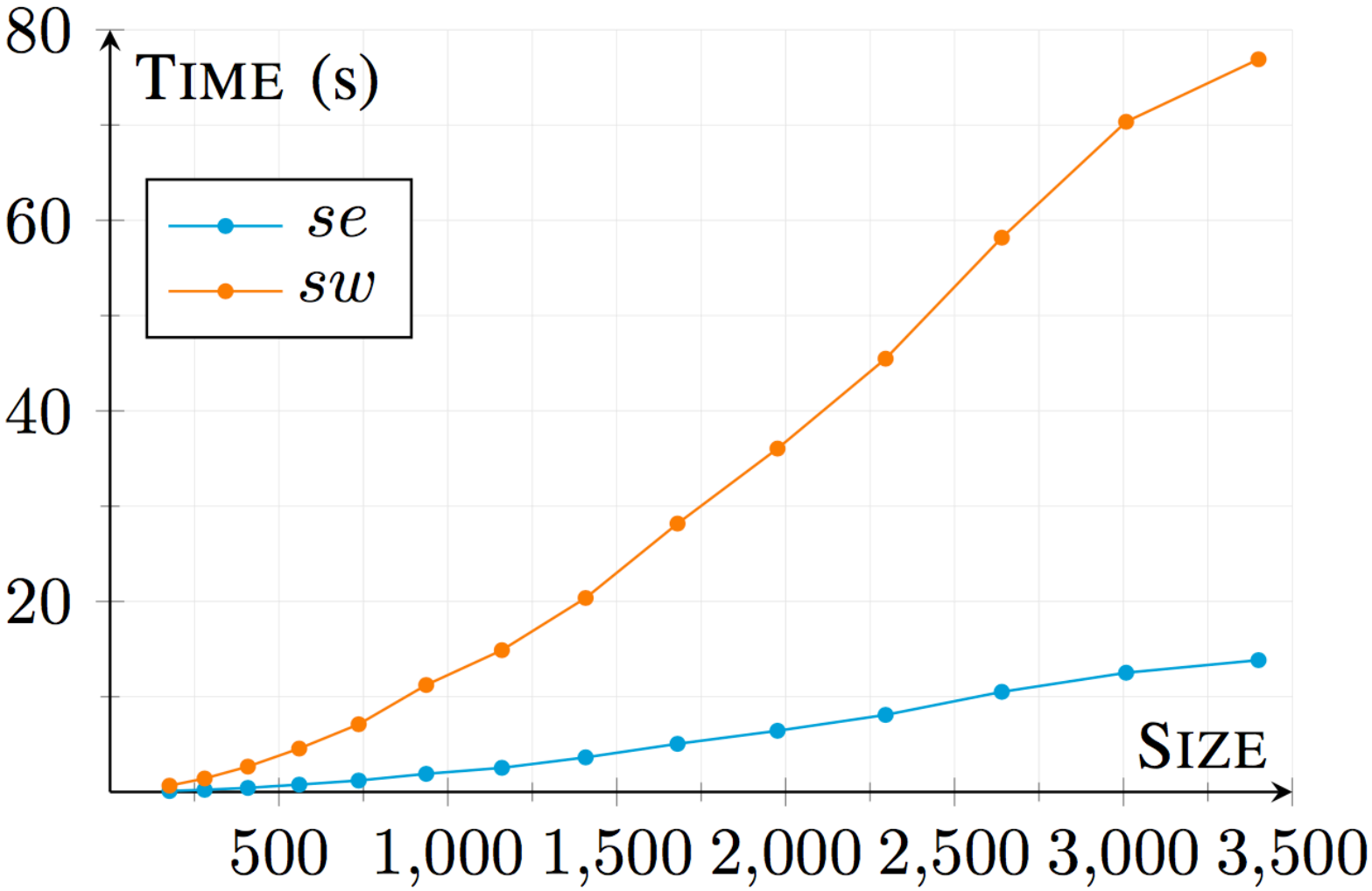
Cyclic corridor results

	Unit	Results							
Robots		5	10	15	20	25	30	35	40
Congestion	%	3	7	11	15	19	23	27	31
$S_c \equiv S_{se}$	%	100	100	100	100	100	100	100	96
S_{sw}	%	100	100	100	100	100	100	100	96
S_{bs}	%	100	100	100	100	100	100	100	98
T_c	s	0.26	1.42	4.23	9.44	17.01	27.78	39.28	51.24
T_{se}	s	0.36	1.72	4.77	10.25	18.02	29.03	40.51	52.33
T_{sw}	s	0.5	2.67	7.37	15.55	27.04	42.53	58.68	75.01
T_{bs}	s	0.86	4.39	12.14	25.8	45.06	71.56	99.2	127.34
μ_c		18.03	20.3	22.48	25.2	28.27	33.14	37.98	48.07
μ_{se}		17.95	20.08	22.08	24.51	27.49	31.99	36.4	44.93
μ_{sw}		17.95	20.02	21.91	24.05	26.57	31.13	35.42	43.83
μ_{bs}		17.95	20.02	21.89	24.02	26.45	30.78	34.81	42.95
μ_c/LBE		1.01	1.04	1.12	1.24	1.36	1.59	1.8	2.22
μ_{se}/LBE		1.01	1.03	1.1	1.21	1.32	1.53	1.73	2.08
μ_{sw}/LBE		1.01	1.03	1.09	1.18	1.28	1.49	1.68	2.03
μ_{bs}/LBE		1.01	1.03	1.09	1.18	1.27	1.47	1.65	1.99
I_{se}	%	34.01	25.26	16.91	14.3	10.2	9.39	9.32	11.94
I_{sw}	%	34.01	32.02	24.71	23.67	22.41	16.4	14.98	15.83
I_{bs}	%	34.01	32.02	25.49	24.27	23.94	19.32	18.67	19.25

Grid improvements



Time required



Outline

Conclusions

- In this work we proposed a heuristic algorithm to improve solutions to the multi-robot path-planning problem implemented in a DES context using timed discrete event automata.
- The algorithm was tested on a variety of problems and it was shown that its use in any planning strategy is useful in terms of:
  1. **Execution time**: it was reduced up to 45%;
  2. **Number of solutions**: it was incremented up to 13% more.

Conclusions

- In this work we proposed a heuristic algorithm to improve solutions to the multi-robot path-planning problem implemented in a DES context using timed discrete event automata.
- The algorithm was tested on a variety of problems and it was shown that its use in any planning strategy is useful in terms of:
  1. **Execution time**: it was reduced up to 45%;
  2. **Number of solutions**: it was incremented up to 13% more.

Conclusions

- In this work we proposed a heuristic algorithm to improve solutions to the multi-robot path-planning problem implemented in a DES context using timed discrete event automata.
- The algorithm was tested on a variety of problems and it was shown that its use in any planning strategy is useful in terms of:
  1. **Execution time**: it was reduced up to 45%;
  2. **Number of solutions**: it was incremented up to 13% more.

Conclusions

- In this work we proposed a heuristic algorithm to improve solutions to the multi-robot path-planning problem implemented in a DES context using timed discrete event automata.
- The algorithm was tested on a variety of problems and it was shown that its use in any planning strategy is useful in terms of:
  1. **Execution time**: it was reduced up to 45%;
  2. **Number of solutions**: it was incremented up to 13% more.

Future works

Future works will aim at

- A smart choosing of the sub-trajectories to be prioritized;
- Planning strategies to avoid cases in which applying the swap strategy a solution is not found;
- Testing of more complicated environments, such as:
  1. The presence of places in which a vehicle cannot stop;
  2. Movements of the robot with different velocities.

Future works

Future works will aim at

- A smart choosing of the sub-trajectories to be prioritized;
- Planning strategies to avoid cases in which applying the swap strategy a solution is not found;
- Testing of more complicated environments, such as:
  1. The presence of places in which a vehicle cannot stop;
  2. Movements of the robot with different velocities.

Future works

Future works will aim at

- A smart choosing of the sub-trajectories to be prioritized;
- Planning strategies to avoid cases in which applying the swap strategy a solution is not found;
- Testing of more complicated environments, such as:
  1. The presence of places in which a vehicle cannot stop;
  2. Movements of the robot with different velocities.

Future works

Future works will aim at

- A smart choosing of the sub-trajectories to be prioritized;
- Planning strategies to avoid cases in which applying the swap strategy a solution is not found;
- Testing of more complicated environments, such as:
  1. The presence of places in which a vehicle cannot stop;
  2. Movements of the robot with different velocities.

Future works

Future works will aim at

- A smart choosing of the sub-trajectories to be prioritized;
- Planning strategies to avoid cases in which applying the swap strategy a solution is not found;
- Testing of more complicated environments, such as:
  1. The presence of places in which a vehicle cannot stop;
  2. Movements of the robot with different velocities.

Thank you for the attention