Multi-Robot Motion Planning

Kiril Solovey

TAU

March 6, 2011

What is Multi-Robot Motion Planning?

- Given:

What is Multi-Robot Motion Planning?

- Given:
  - m robots

What is Multi-Robot Motion Planning?

- Given:
  - m robots
  - common k dimensional workspace with obstacles

What is Multi-Robot Motion Planning?

- Given:
  - m robots
  - common k dimensional workspace with obstacles
  - each robot has a *start* and *goal* configurations

What is Multi-Robot Motion Planning?

- Given:
  - m robots
  - common k dimensional workspace with obstacles
  - each robot has a *start* and *goal* configurations
- Find:

What is Multi-Robot Motion Planning?

- Given:
  - m robots
  - common k dimensional workspace with obstacles
  - each robot has a *start* and *goal* configurations
- Find:
  - a path for each robot, from *start* to *goal*

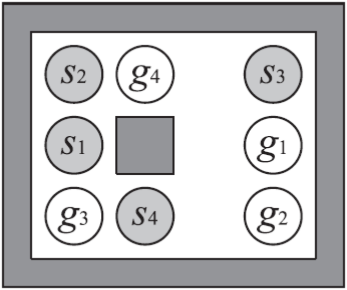
What is Multi-Robot Motion Planning?

- Given:
  - m robots
  - common k dimensional workspace with obstacles
  - each robot has a *start* and *goal* configurations
- Find:
  - a path for each robot, from *start* to *goal*
  - while avoiding collision with **obstacles** and **other robots**

What is Multi-Robot Motion Planning?

- Given:
  - m robots
  - common k dimensional workspace with obstacles
  - each robot has a *start* and *goal* configurations
- Find:
  - a path for each robot, from *start* to *goal*
  - while avoiding collision with **obstacles** and **other robots**

Example



Main Approaches

Decoupled VS Centralized

Decoupled Planners

- Decompose the general problem into small sub-problems

Decoupled Planners

- Decompose the general problem into small sub-problems
- Merge the results of the sub-problems into a global result

Decoupled Planners

- Decompose the general problem into small sub-problems
- Merge the results of the sub-problems into a global result
- This approach is usually very efficient

Decoupled Planners

- Decompose the general problem into small sub-problems
- Merge the results of the sub-problems into a global result
- This approach is usually very efficient
- Unfortunately it's not complete

Decoupled Planners

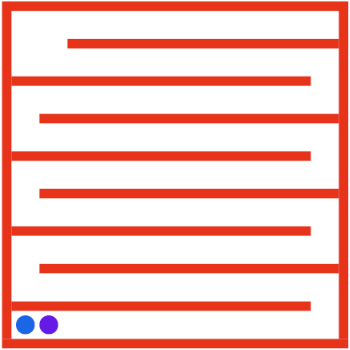
- Decompose the general problem into small sub-problems
- Merge the results of the sub-problems into a global result
- This approach is usually very efficient
- Unfortunately it's not complete

Decoupled Planners

The following problem cannot be solved using decoupled planners:

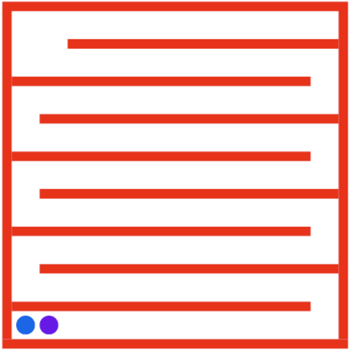
Decoupled Planners

The following problem cannot be solved using decoupled planners:



Decoupled Planners

The following problem cannot be solved using decoupled planners:



The two robots have to exchange positions.

Centralized Approach

- The robots are treated as a group and sometimes as one composite robot.

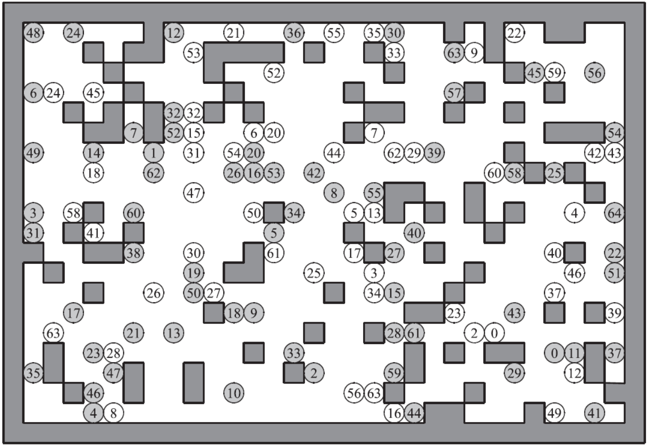
Centralized Approach

- The robots are treated as a group and sometimes as one composite robot.
- Usually yields a *complete* planner.

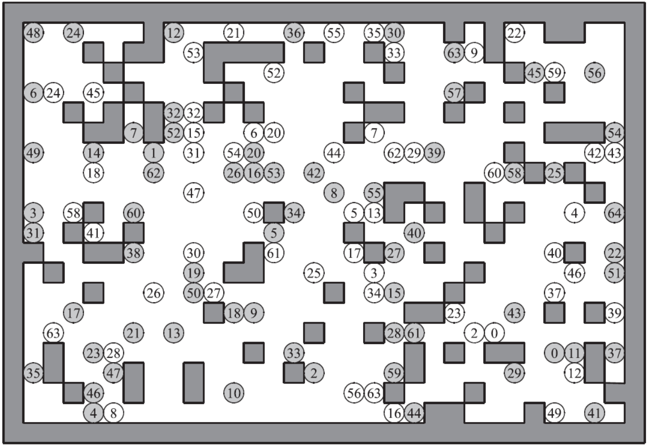
Centralized Approach

- The robots are treated as a group and sometimes as one composite robot.
- Usually yields a *complete* planner.
- Known techniques applicable only for small-scale problems with low degrees of freedom.

Centralized Approach



Centralized Approach



Introduction

Centralized Path Planning for Multiple Robots: Optimal Decoupling into Sequential Plans

Jur van den Berg   Jack Snoeyink   Ming Lin   Dinesh Manocha

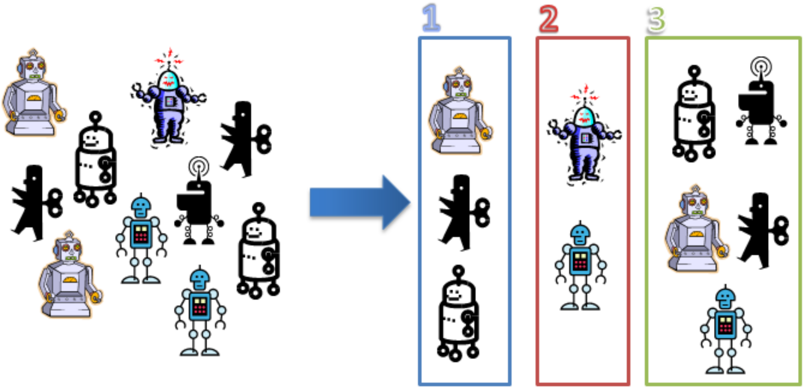
Introduction

- The algorithm decomposes the multi-robot motion planning problem into lower-dimensional sub-problems that can be executed **sequentially**.

Introduction

- The algorithm decomposes the multi-robot motion planning problem into lower-dimensional sub-problems that can be executed **sequentially**.
- In these sub-problems, individual robots will be coupled and considered as composite robots.

Introduction



Preliminaries

Notations:

Preliminaries

Notations:

- n robots, $r_1, \ldots, r_n$, common workspace.

Preliminaries

Notations:

- n robots, $r_1, \ldots, r_n$, common workspace.
- $C(r_i)$: Configuration space of robot r_i.

Preliminaries

Notations:

- n robots, $r_1, \ldots, r_n$, common workspace.
- $C(r_i)$: Configuration space of robot r_i.
- $s_i, g_i \in C(r_i)$: start and goal configurations of robot r_i.

Preliminaries

Notations:

- n robots, $r_1, \ldots, r_n$, common workspace.
- $C(r_i)$: Configuration space of robot r_i.
- $s_i, g_i \in C(r_i)$: start and goal configurations of robot r_i.

Goal

Compute a path $\pi : [0,1] \to C(r_1) \times \ldots \times C(r_n)$ such that initially $\pi(0) = (s_1, \ldots, s_n)$, finally $\pi(1) = (g_1, \ldots, g_n)$, and for every $t \in (0,1)$ $\pi(t)$ describes a collision free (obstacles, robots) position.

Coupled Relation

Definition

For a robot r_i the *Active Interval* τ_i is defined as the open interval from the first time r_i leaves its start position to the last time it reaches its goal position.

Coupled Relation

Definition

For a robot r_i the *Active Interval* τ_i is defined as the open interval from the first time r_i leaves its start position to the last time it reaches its goal position.

Definition

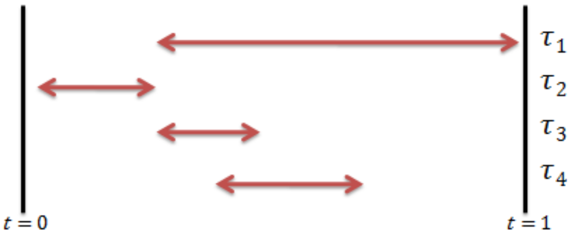
Two robots r_i, r_j are coupled if their active intervals intersect, $\tau_i \cap \tau_j \neq \emptyset$.

Example

Consider the following active intervals for a given path π:

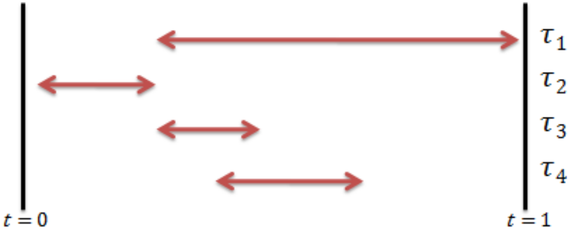
Example

Consider the following active intervals for a given path π:



Example

Consider the following active intervals for a given path π:



The coupled robots are $\{r_1, r_3\}, \{r_1, r_4\}, \{r_3, r_4\}$.

Execution Sequence

Definition

A *Composite Robot* R is a subset of $\{r_1, \ldots, r_n\}$.

Execution Sequence

Definition

A *Composite Robot* R is a subset of $\{r_1, \ldots, r_n\}$.

Definition

Execution Sequence is an ordered partition of the n robots into a sequence $S = (R_1, \ldots, R_k)$ of composite robots, such that

- $R_1 \cup \ldots \cup R_k = \{r_1, \ldots, r_n\}$
- For every $i \neq j$ $R_i \cap R_j = \emptyset$.

Execution Sequence

Definition

S is *valid* if it describes the *coupled* relation of the robots in some solution path.

Execution Sequence

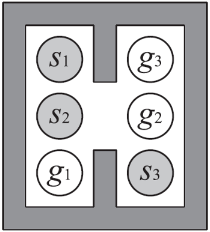
Definition

S is *valid* if it describes the *coupled* relation of the robots in some solution path.

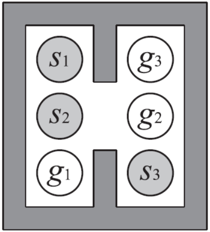
Definition

S is *optimal* if it is the solution sequence with the lowest degree of its largest *composite robot*.

Example

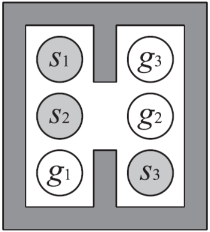


Example

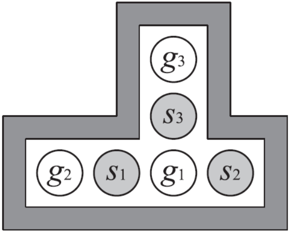


$$S = [r_3, r_2, r_1]$$

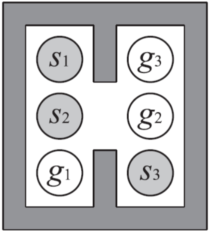
Example



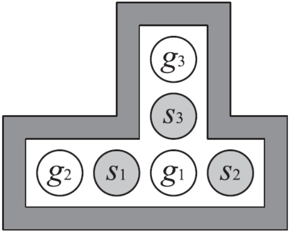
$$S = [r_3, r_2, r_1]$$



Example



$S = [r_3, r_2, r_1]$



$S = [r_3, r_1 r_2]$

Validating Execution Sequences

Theorem

S is valid if, for all $i \in [1, k]$, robot $R_i \in S$ can move from start to goal without colliding with

Validating Execution Sequences

Theorem

S is valid if, for all $i \in [1, k]$, robot $R_i \in S$ can move from start to goal without colliding with

- *$\{R_1, \ldots, R_{i-1}\}$ in their goal positions*

Validating Execution Sequences

Theorem

S is valid if, for all $i \in [1, k]$, robot $R_i \in S$ can move from start to goal without colliding with

- $\{R_1, \ldots, R_{i-1}\}$ *in their goal positions*
- $\{R_{i+1}, \ldots, R_n\}$ *in their start positions*

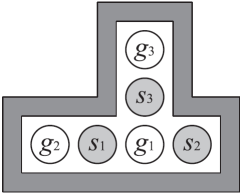
Validating Execution Sequences

Theorem

S is valid if, for all $i \in [1, k]$, robot $R_i \in S$ can move from start to goal without colliding with

- $\{R_1, \ldots, R_{i-1}\}$ *in their goal positions*
- $\{R_{i+1}, \ldots, R_n\}$ *in their start positions*

For example, $S = [r_3, r_1 r_2]$ is valid for the scene

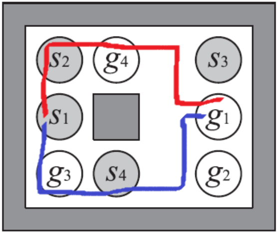


Key Observation

- How do we construct valid execution sequences?

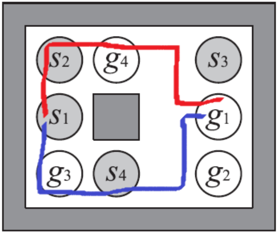
Key Observation

- How do we construct valid execution sequences?
- Consider the following scene:



Key Observation

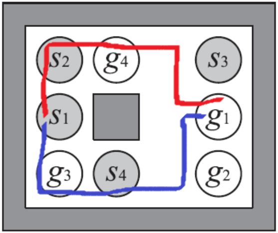
- How do we construct valid execution sequences?
- Consider the following scene:



- Assume we found the blue trajectory for r_1 and we want to use it in the global solution.

Key Observation

- How do we construct valid execution sequences?
- Consider the following scene:



- Assume we found the blue trajectory for r_1 and we want to use it in the global solution.
- What will be the order of r_1, r_3, r_4 in the solution sequence?

Order Constraints

- Consider a specific trajectory for the robot R.

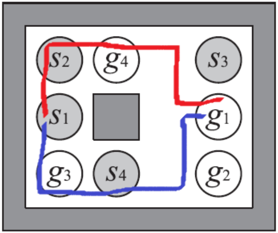
Order Constraints

- Consider a specific trajectory for the robot R.
- If this trajectory collides with a goal configuration of the robot r_j we write $R \prec r_j$.

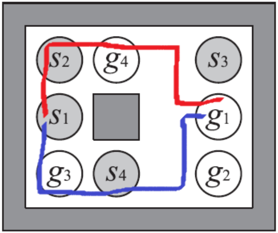
Order Constraints

- Consider a specific trajectory for the robot R.
- If this trajectory collides with a goal configuration of the robot r_j we write $R \prec r_j$.
- If the trajectory collides with the start configuration of r_i we write $r_i \prec R$.

Example



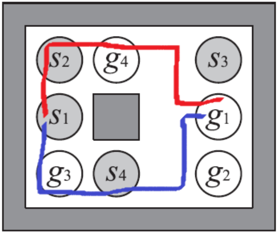
Example



For the first trajectory

$$r_2 \prec r_1 \wedge r_1 \prec r_4$$

Example



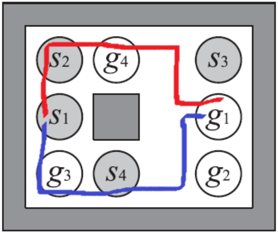
For the first trajectory

$$r_2 \prec r_1 \wedge r_1 \prec r_4$$

The second trajectory:

$$r_1 \prec r_3 \wedge r_4 \prec r_1$$

Example



For the first trajectory

$$r_2 \prec r_1 \wedge r_1 \prec r_4$$

The second trajectory:

$$r_1 \prec r_3 \wedge r_4 \prec r_1$$

And combined:

$$P(r_1) = (r_2 \prec r_1 \wedge r_1 \prec r_4) \vee (r_1 \prec r_3 \wedge r_4 \prec r_1)$$

Constraints from an Execution Sequence

Let $P(R)$ be the collection of all collision of R for **all** paths, in DNF formula, separated by "OR".

Constraints from an Execution Sequence

Let $P(R)$ be the collection of all collision of R for **all** paths, in DNF formula, separated by "OR".
If we AND the constraints for all robots in an execution sequence, we get the expression that must be satisfied for it to be a solution sequence.

Constraints from an Execution Sequence

Let $P(R)$ be the collection of all collision of R for **all** paths, in DNF formula, separated by "OR".
If we AND the constraints for all robots in an execution sequence, we get the expression that must be satisfied for it to be a solution sequence.

Observation

An execution sequence $S = (R_1, \ldots, R_k)$ *is a* solution sequence *iff* S *satisfies the constraints expression* $P(R_1) \wedge \ldots \wedge P(R_k)$.

The CR Planner

- How do we find the ordered constraints?

The CR Planner

- How do we find the ordered constraints?
- The CR planner produces constraints on the execution sequence induced by small subsets of robots.

The CR Planner

- How do we find the ordered constraints?
- The CR planner produces constraints on the execution sequence induced by small subsets of robots.
- Our algorithm incrementally calls the planner on higher and higher degree sub-problems, using the discovered constraints to determine what robots must be coupled.

The CR Planner

- How do we find the ordered constraints?
- The CR planner produces constraints on the execution sequence induced by small subsets of robots.
- Our algorithm incrementally calls the planner on higher and higher degree sub-problems, using the discovered constraints to determine what robots must be coupled.

Constraint Graphs

- The algorithm maintains the obtained constraints obtained in a constraint expression E.

Constraint Graphs

- The algorithm maintains the obtained constraints obtained in a constraint expression E.
- E is in a DNF form, i.e. $E = J_1 \vee J_2 \vee \ldots$, of conjunctions J_i.

Constraint Graphs

- The algorithm maintains the obtained constraints obtained in a constraint expression E.
- E is in a DNF form, i.e. $E = J_1 \vee J_2 \vee \ldots$, of conjunctions J_i.
- Each conjunction J can be represented as a constraints graph $G(J)$.

Conjunction as a Directed Graph

Each conjunction can be represented as a directed graph.

Conjunction as a Directed Graph

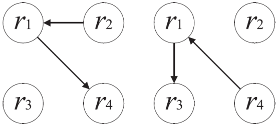
Each conjunction can be represented as a directed graph.
Example:

$$(r_2 \prec r_1 \land r_1 \prec r_4) \lor (r_1 \prec r_3 \land r_4 \prec r_1)$$

Conjunction as a Directed Graph

Each conjunction can be represented as a directed graph.
Example:

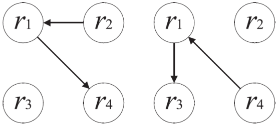
$$(r_2 \prec r_1 \wedge r_1 \prec r_4) \vee (r_1 \prec r_3 \wedge r_4 \prec r_1)$$



Conjunction as a Directed Graph

Each conjunction can be represented as a directed graph.
Example:

$$(r_2 \prec r_1 \wedge r_1 \prec r_4) \vee (r_1 \prec r_3 \wedge r_4 \prec r_1)$$

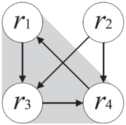


Constraint Graphs

- What happens if the graph contains a cycle?

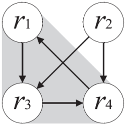
Constraint Graphs

- What happens if the graph contains a cycle?



Constraint Graphs

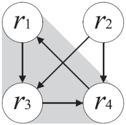
- What happens if the graph contains a cycle?



- There is a contradiction!

Constraint Graphs

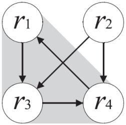
- What happens if the graph contains a cycle?



- There is a contradiction!
- This means that the involved robots need to be coordinated.

Constraint Graphs

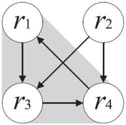
- What happens if the graph contains a cycle?



- There is a contradiction!
- This means that the involved robots need to be coordinated.
- Let $G^{SCC}(J)$ denote the *component graph* of $G(J)$, which contains a node for each *strongly connected component* in $G(J)$.

Constraint Graphs

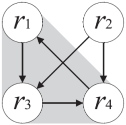
- What happens if the graph contains a cycle?



- There is a contradiction!
- This means that the involved robots need to be coordinated.
- Let $G^{SCC}(J)$ denote the *component graph* of $G(J)$, which contains a node for each *strongly connected component* in $G(J)$.
- Each node in $G^{SCC}(J)$ corresponds to a (composite)robot.

Constraint Graphs

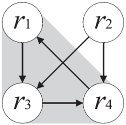
- What happens if the graph contains a cycle?



- There is a contradiction!
- This means that the involved robots need to be coordinated.
- Let $G^{SCC}(J)$ denote the *component graph* of $G(J)$, which contains a node for each *strongly connected component* in $G(J)$.
- Each node in $G^{SCC}(J)$ corresponds to a (composite)robot.
- Topologically sorting $G^{SCC}(J)$ gives an execution sequence $S(J)$ of composite robots.

Constraint Graphs

- What happens if the graph contains a cycle?



- There is a contradiction!
- This means that the involved robots need to be coordinated.
- Let $G^{SCC}(J)$ denote the *component graph* of $G(J)$, which contains a node for each *strongly connected component* in $G(J)$.
- Each node in $G^{SCC}(J)$ corresponds to a (composite)robot.
- Topologically sorting $G^{SCC}(J)$ gives an execution sequence $S(J)$ of composite robots.

Incrementally Building the Execution Sequence

- E is initialized to be T (true).

Incrementally Building the Execution Sequence

- E is initialized to be T (true).
- The algorithm selects iteratively the smallest composite robot $R_{min} \in E$ that haven't been planned yet.

Incrementally Building the Execution Sequence

- E is initialized to be T (true).
- The algorithm selects iteratively the smallest composite robot $R_{min} \in E$ that haven't been planned yet.
- CR planner is invoked on R_{min} and returns $P(R_{min})$.

Incrementally Building the Execution Sequence

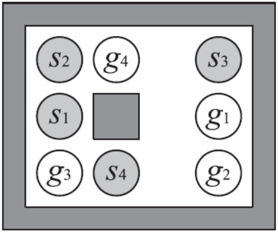
- E is initialized to be T (true).
- The algorithm selects iteratively the smallest composite robot $R_{min} \in E$ that haven't been planned yet.
- CR planner is invoked on R_{min} and returns $P(R_{min})$.
- $P(R_{min})$ is incorporated into E.

Incrementally Building the Execution Sequence

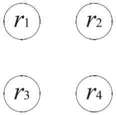
- E is initialized to be T (true).
- The algorithm selects iteratively the smallest composite robot $R_{min} \in E$ that haven't been planned yet.
- CR planner is invoked on R_{min} and returns $P(R_{min})$.
- $P(R_{min})$ is incorporated into E.
- We stop when there exists $J \in E$ for which all robots in $S(J)$ have been planned.

Incrementally Building the Execution Sequence: Example

We will run the algorithm on the problem:

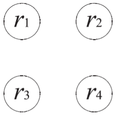


Incrementally Building the Execution Sequence: Example

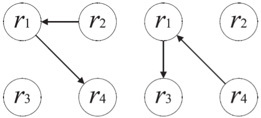


Iteration 1. $L = \emptyset$, $R_{\min} = r_1$, $\mathcal{P}(r_1) = (r_2 \prec r_1 \wedge r_1 \prec r_4) \vee (r_1 \prec r_3 \wedge r_4 \prec r_1)$.

Incrementally Building the Execution Sequence: Example

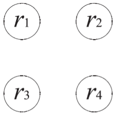


Iteration 1. $L = \emptyset$, $R_{\min} = r_1$, $\mathcal{P}(r_1) = (r_2 \prec r_1 \wedge r_1 \prec r_4) \vee (r_1 \prec r_3 \wedge r_4 \prec r_1)$.

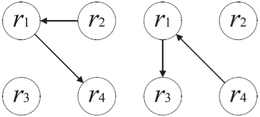


Iteration 2. $L = \{r_1\}$, $R_{\min} = r_2$, $\mathcal{P}(r_2) = r_2 \prec r_4 \vee (r_1 \prec r_2 \wedge r_2 \prec r_3 \wedge r_4 \prec r_2)$.

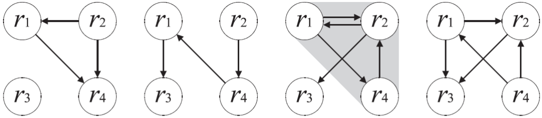
Incrementally Building the Execution Sequence: Example



Iteration 1. $L = \emptyset$, $R_{\min} = r_1$, $\mathcal{P}(r_1) = (r_2 \prec r_1 \wedge r_1 \prec r_4) \vee (r_1 \prec r_3 \wedge r_4 \prec r_1)$.

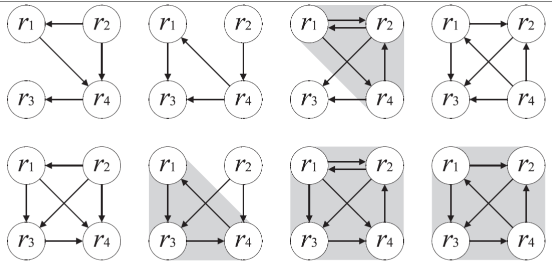


Iteration 2. $L = \{r_1\}$, $R_{\min} = r_2$, $\mathcal{P}(r_2) = r_2 \prec r_4 \vee (r_1 \prec r_2 \wedge r_2 \prec r_3 \wedge r_4 \prec r_2)$.



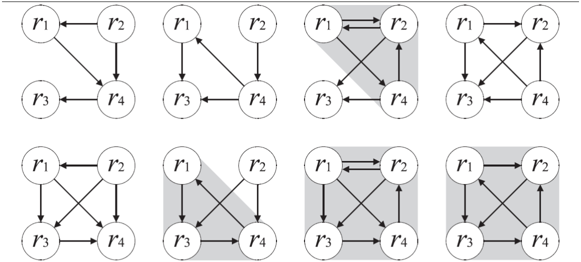
Iteration 3. $L = \{r_1, r_2\}$, $R_{\min} = r_3$, $\mathcal{P}(r_3) = r_4 \prec r_3 \vee (r_3 \prec r_4 \wedge r_2 \prec r_3 \wedge r_1 \prec r_3)$.

Incrementally Building the Execution Sequence: Example



Iteration 4. $L = \{r_1, r_2, r_3\}$, $R_{\min} = r_4$, $\mathcal{P}(r_4) = \top$, so no change to graphs.

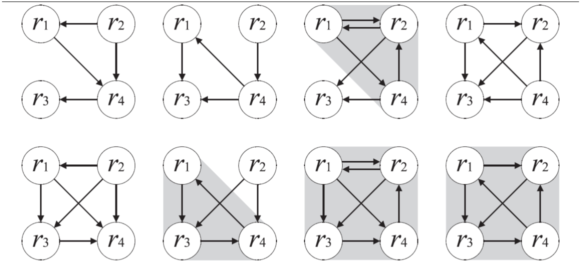
Incrementally Building the Execution Sequence: Example



Iteration 4. $L = \{r_1, r_2, r_3\}$, $R_{\min} = r_4$, $\mathcal{P}(r_4) = \top$, so no change to graphs.

- After the 4th iteration, $L = \{r_1, r_2, r_3, r_4\}$.

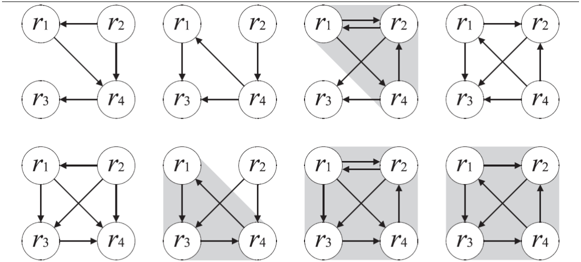
Incrementally Building the Execution Sequence: Example



Iteration 4. $L = \{r_1, r_2, r_3\}$, $R_{\min} = r_4$, $\mathcal{P}(r_4) = \top$, so no change to graphs.

- After the 4th iteration, $L = \{r_1, r_2, r_3, r_4\}$.
- J_1, J_2, J_4, J_5 have all their SCC in L.

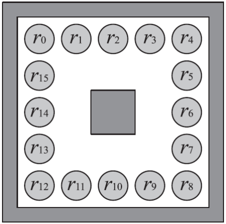
Incrementally Building the Execution Sequence: Example



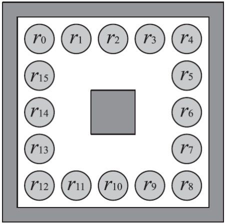
Iteration 4. $L = \{r_1, r_2, r_3\}$, $R_{\min} = r_4$, $\mathcal{P}(r_4) = \top$, so no change to graphs.

- After the 4th iteration, $L = \{r_1, r_2, r_3, r_4\}$.
- J_1, J_2, J_4, J_5 have all their SCC in L.
- The algorithm terminates and returns $S(J_1)$.

Scene 1

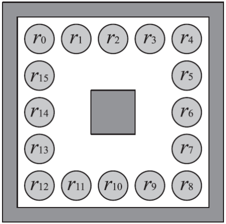


Scene 1



- Each robot r_i changes position with $r_{((i+8) \mod 16)}$

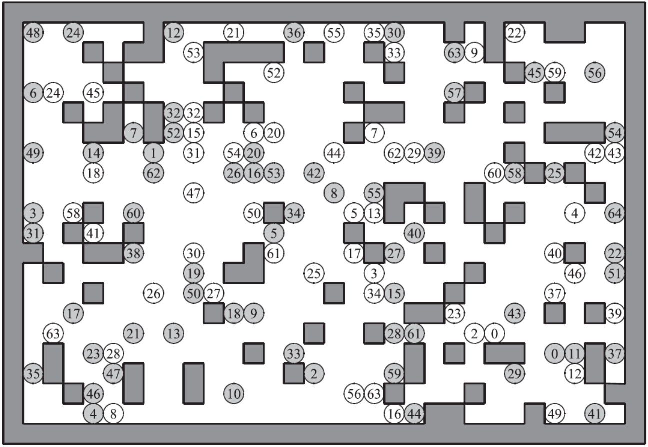
Scene 1



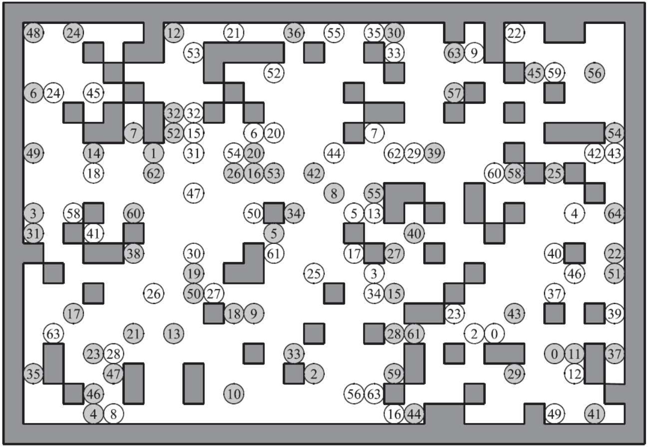
- Each robot r_i changes position with $r_{((i+8) \mod 16)}$
- The algorithm returned

$$(r_0 r_7 r_8 r_{15}, r_1 r_9, r_2 r_{10}, r_3 r_{11}, r_4 r_5 r_{12} r_{13}, r_6 r_{14}, r_7 r_{15})$$

Scene 2

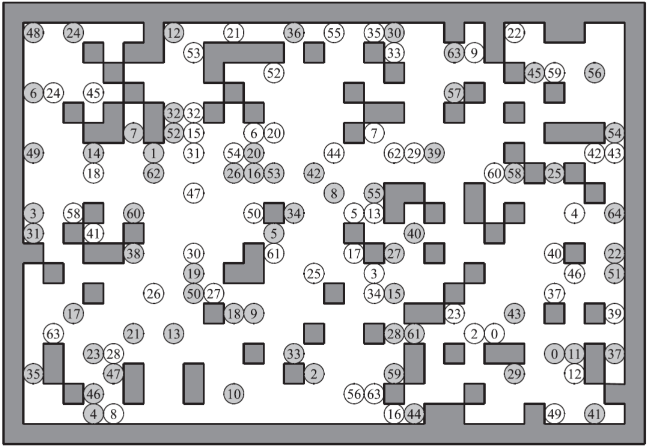


Scene 2



- 65 robots

Scene 2



- 65 robots
- Solution sequence involves only **individual** robots!

Project

- Two possible settings, two possible CR Planners:

Project

- Two possible settings, two possible CR Planners:
  - Grid workspace: motion planning on graphs.

Project

- Two possible settings, two possible CR Planners:
  - Grid workspace: motion planning on graphs.
  - Polygonal robots and obstacles (harder): sampling-based techniques.

Project

- Two possible settings, two possible CR Planners:
  - Grid workspace: motion planning on graphs.
  - Polygonal robots and obstacles (harder): sampling-based techniques.
- In both cases, the main algorithm remains the same.

The End

"... and the robots lived happily ever after."