Graph-based optimal reconfiguration planning for self-reconfigurable robots

Feili Hou, Wei-Min Shen

presented by Viktória Vozárová

March 19, 2018

Outline

- Motivation
- Problem definition
- Optimal algorithm (MDCOP)
- Greedy algorithm (GreedyCM)
- Experiments

Motivation

Self-reconfigurable robots:

- distributive
- modular
- reconfigurable

Motivation

Self-reconfigurable robots:

- distributive
- modular
- reconfigurable

Sounds amazing. How to implement it?

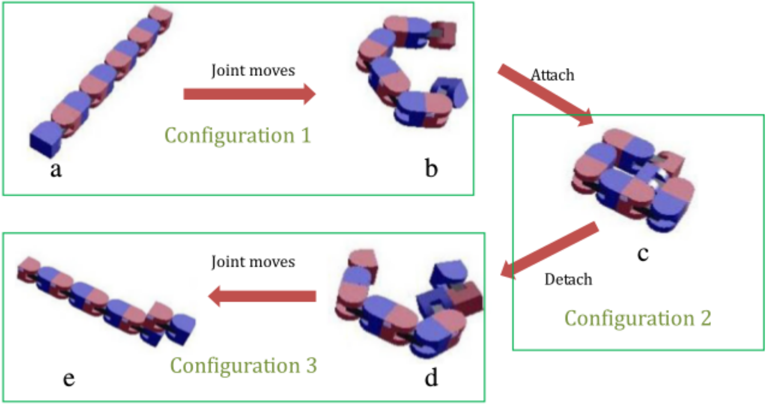
Motivation

Self-reconfigurable robots:

- distributive
- modular
- reconfigurable

Sounds amazing. How to implement it? Which layer?

Problem definition – example



Problem definition

Given *the initial configuration* and *the goal configuration*, determine the minimal number of attach/detach actions and provide a minimal plan.

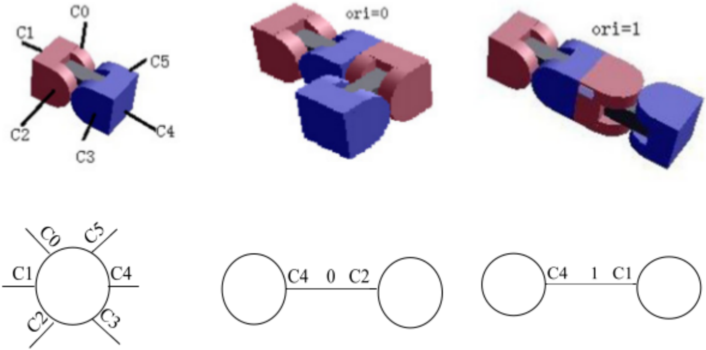
Problem definition

Given *the initial configuration* and *the goal configuration*, determine the minimal number of attach/detach actions and provide a minimal plan.

This problem definition:

- assumes that attach/detach actions are expensive.
- ignores joint movement.
- ignores real-world physics.

Robot configuration as c-graph



Maximum configuration matching

Definition

Configuration matching for configurations H_1 and H_2 is a bijection from nodes of H_1 to nodes of H_2.

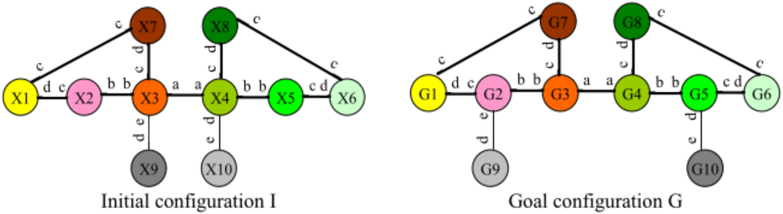
Definition

Given an initial configuration I, a goal configuration G and a configuration matching, edges $(u, v) \in I$ and $(u', v') \in G$ are *matched* if u is matched to u' and v is matched to v'.

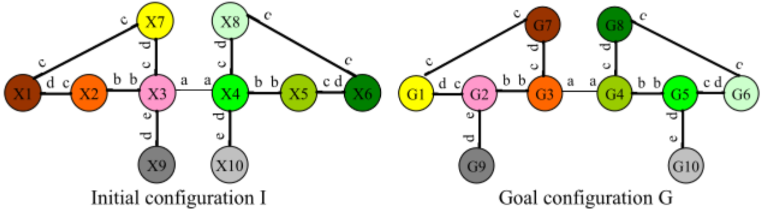
Find a configuration matching that maximizes the number of matched edges.

Getting a reconfiguration plan from matching is straightforward.

Maximum configuration matching – example



(a) An instance of configuration matching that is not maximum.



(b) An instance of maximum configuration matching.

Distributed constraint optimization problem

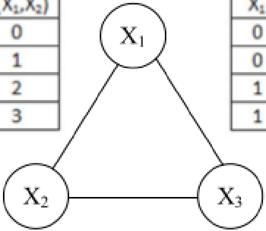
DCOP is defined as $\langle X, D, R \rangle$, where:

- $X = \{X_1, X_2, \ldots X_n\}$ is a set of agents,
- $D = \{D_1, D_2, \ldots D_n\}$ is a set of finite domains,
- $R = \{R_1, R_2, \ldots R_m\}$ is a set of binary constraints $R_k \colon D_i \times D_j \to \mathbb{N}$.

Each agent X_i chooses a value from D_i.
The goal is to minimalize the sum of all constraints.

DCOP – example

X_1	X_2	$R(X_1,X_2)$
0	0	0
0	1	1
1	0	2
1	1	3

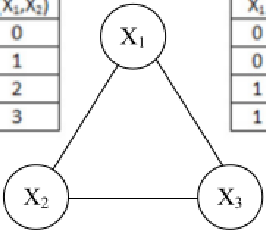


X_1	X_3	$R(X_1,X_3)$
0	0	2
0	1	1
1	0	3
1	1	0

X_2	X_3	$R(X_2,X_3)$
0	0	2
0	1	3
1	0	0
1	1	1

DCOP – example

X_1	X_2	$R(X_1,X_2)$
0	0	0
0	1	1
1	0	2
1	1	3



X_1	X_3	$R(X_1,X_3)$
0	0	2
0	1	1
1	0	3
1	1	0

X_2	X_3	$R(X_2,X_3)$
0	0	2
0	1	3
1	0	0
1	1	1

Minimal solutions:

$X_1 = 0, X_2 = 1, X_3 = 0$ $\qquad$ $X_1 = 0, X_2 = 1, X_3 = 1.$

Mapping to DCOP

Let X be robots in the initial configuration.
Each D_i is a set of nodes in the goal configuration.
Each X_i chooses one node in the goal configuration.

1. If agents X_i and X_j are connected and they choose connected nodes, then $R(X_i, X_j) = 0$. *(matched)*
2. If agents X_i and X_j are connected and they choose not connected nodes, then $R(X_i, X_j) = 1$. *(unmatched)*
3. If agents X_i and X_j choose the same node, then $R(X_i, X_j) = \infty$. *(invalid)*

Minimizing R maximizes the number of matched edges.

Mapping to DCOP – optimization

The third condition is too strict.

Let each D_i be a set of *candidate mates*[1] of X_i.
There can be no candidate: add $\emptyset$ as a wild card.
Add the third condition only if $D_i \cap D_j \neq \emptyset$.

After running DCOP, a node matched to $\emptyset$ can matched to any free node in G.

[1] nodes in G, which have at least one edge with the same connection type

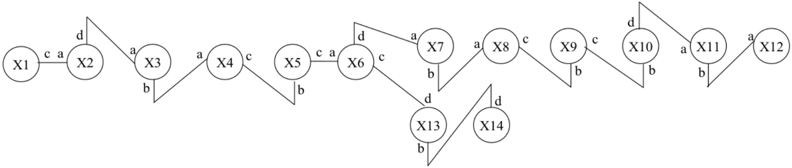
GreedyCM

1. Find maximum common edge sub-configuration (MCESC).
2. Erase found matching.
3. Repeat until all nodes are matched.

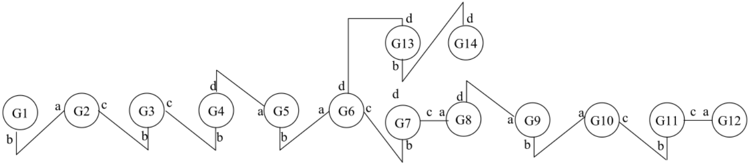
Since node degrees are bound and labelled,
finding MCESC is easy (polynomial).

GreedyCM – example

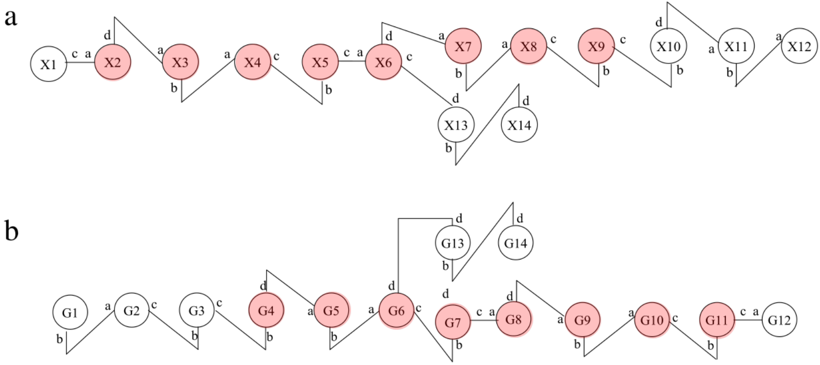
a



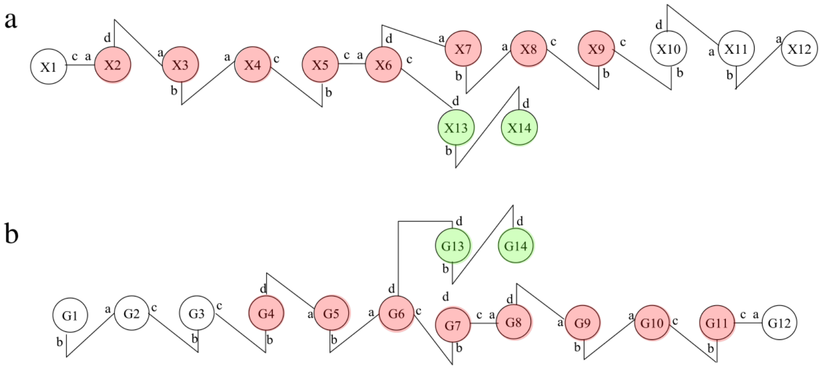
b



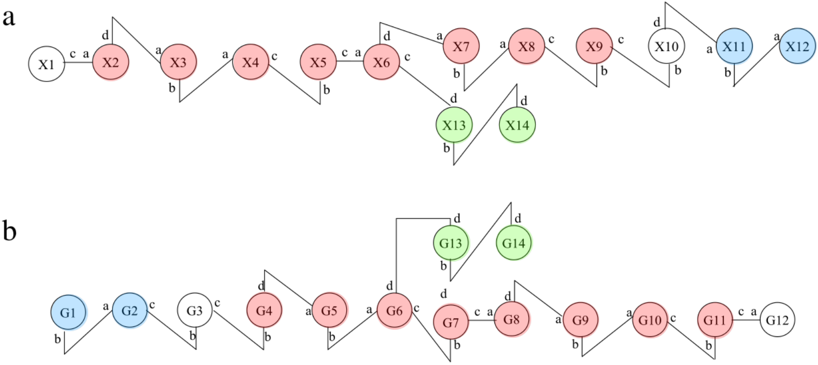
GreedyCM – example



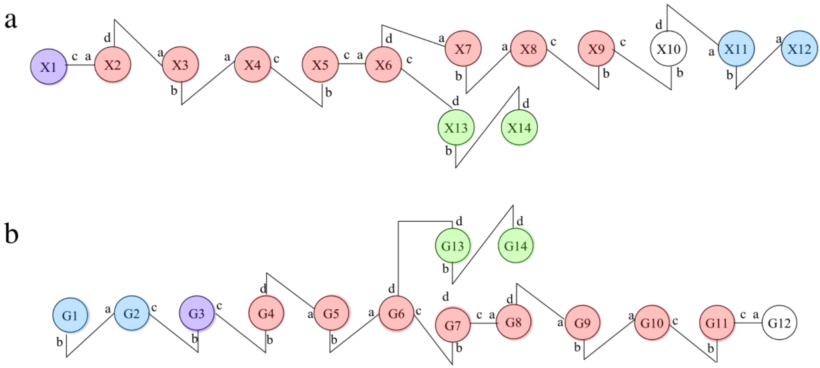
GreedyCM – example



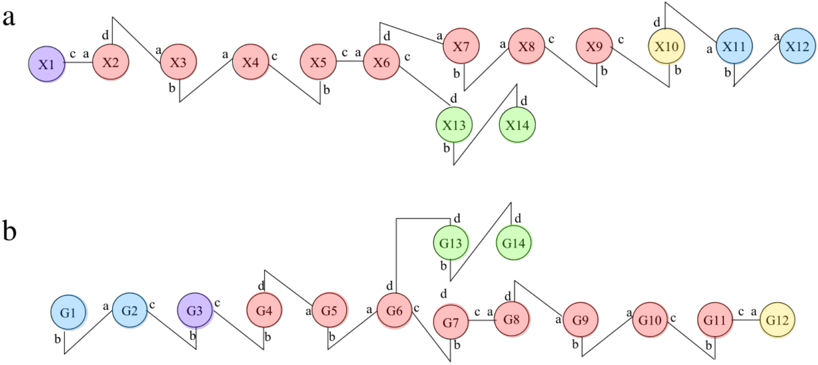
GreedyCM – example



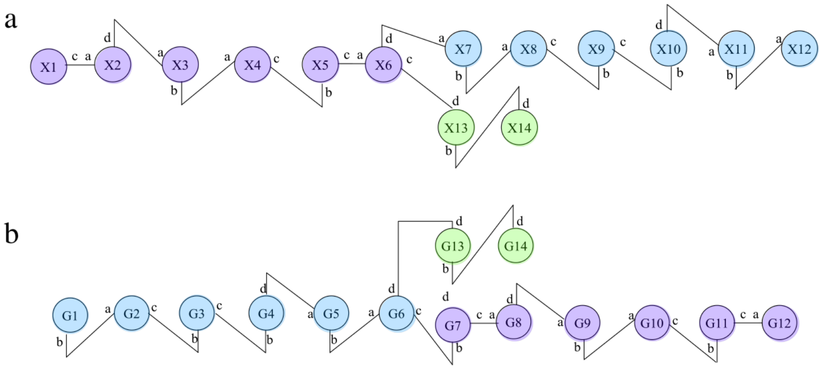
GreedyCM – example



GreedyCM – example



GreedyCM – optimal solution



GreedyCM – distributive

1. Each node $u \in I$ computes for each $v \in G$ MCESC where u is matched to v.
2. All MCESC are sent to the leader.
3. The leader sorts all MCESC in a partial order.
4. Maximal disjoint MCESC are chosen and erased.
5. Repeat until all nodes are matched.

GreedyCM – distributive

1. Each node $u \in I$ computes for each $v \in G$ MCESC where u is matched to v.
2. All MCESC are sent to the leader.
3. The leader sorts all MCESC in a partial order.
4. Maximal disjoint MCESC are chosen and erased.
5. Repeat until all nodes are matched.

Example.

Experimental results

- Videos.
- Paper.

Remaining problems

- Hardware limitation (how to align modules to connect).
- Order of reconfiguration steps.
- Parallel execution.

Remaining problems

- Hardware limitation (how to align modules to connect).
- Order of reconfiguration steps.
- Parallel execution.

. . . and that is what we will try to solve.