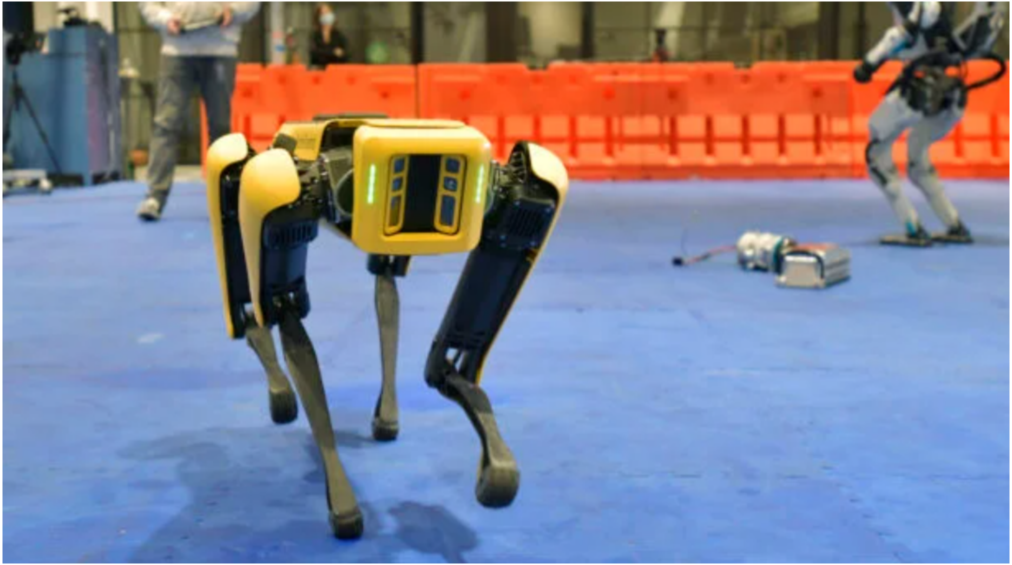
A Graph-Based Search Approach to Planning and Learning

G.S. Groote

Supervisors: Ir. C. Pezzato
Prof. Dr. Ir. M. Wisse
Dr. Ir. C.S. Smith

Delft University of Technology, The Netherlands

July 3, 2023




CHIPS


TIAGo
Doritos
Croky
ZOUTE STICKS

Intro

Table of Content

Table of Content

Intro

- Learn System Models

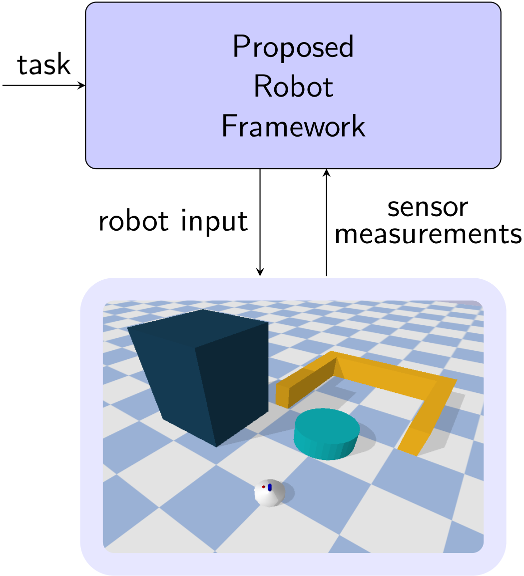
Intro

- Learn System Models
- Navigation Among Movable Objects (NAMO)

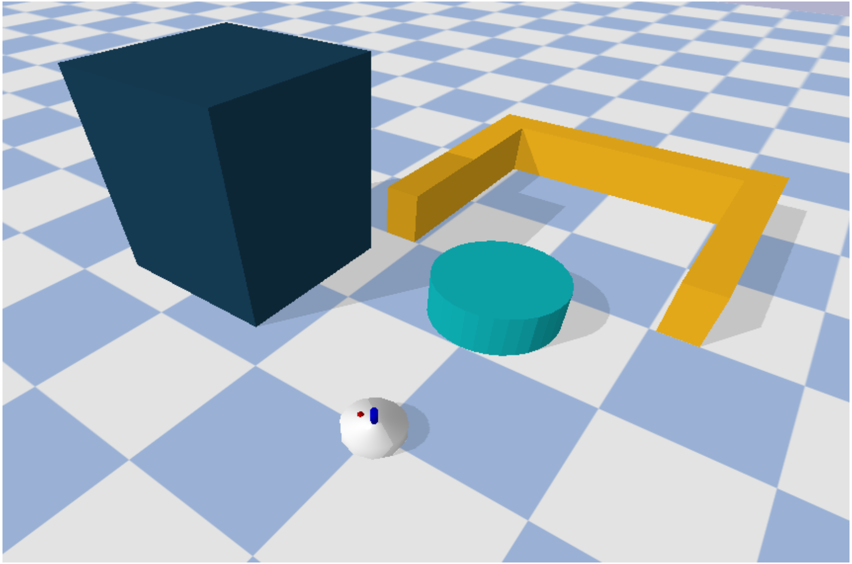
Intro

- Learn System Models
- Navigation Among Movable Objects (NAMO)
- Nonprehensile Pushing

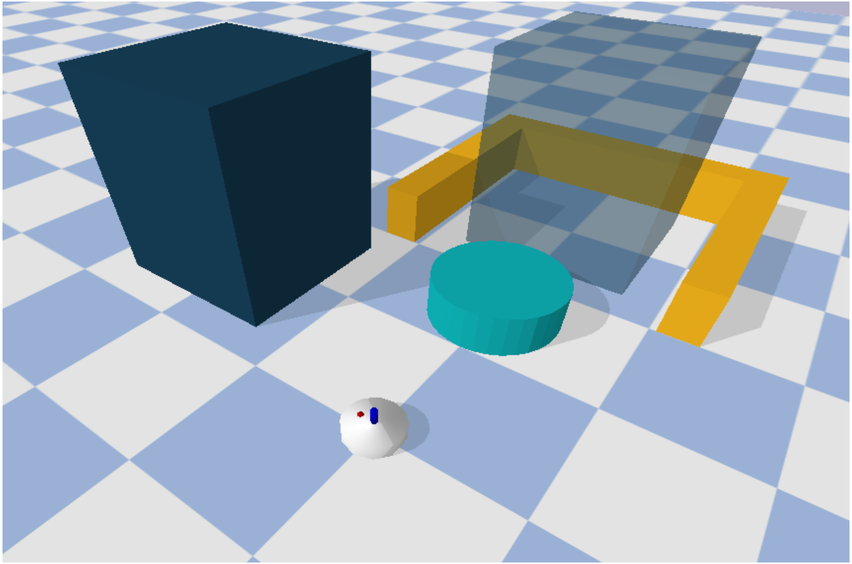
Intro


task
Proposed
Robot
Framework
robot input
sensor
measurements

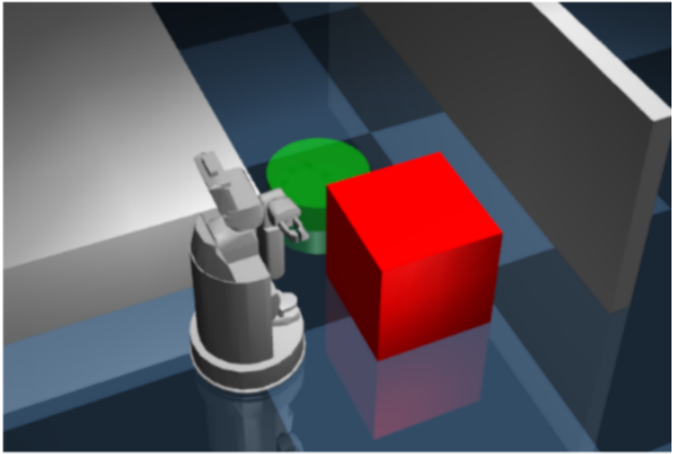
Intro



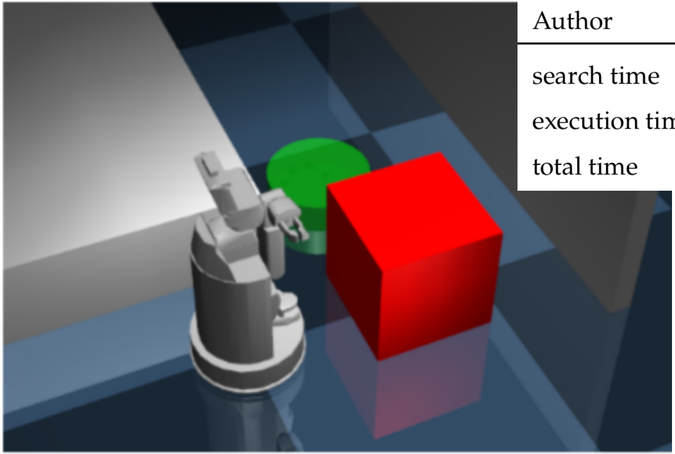
Intro



Intro

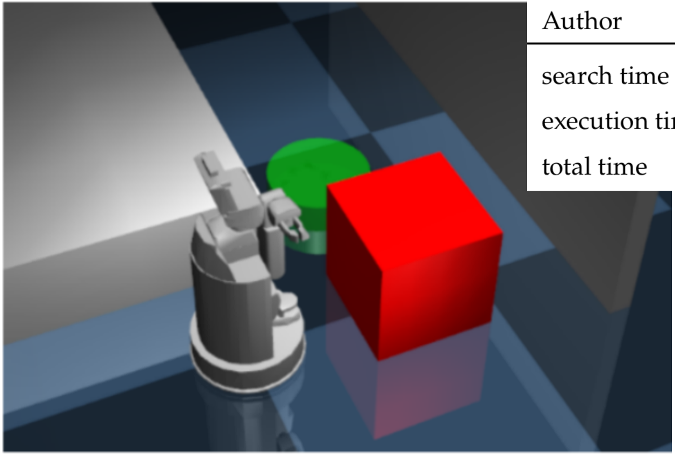


Intro



Author	Wang et al.
search time	109 sec
execution time	67 sec
total time	176 sec

Intro



Author	Wang et al.	Groote
search time	109 sec	?
execution time	67 sec	?
total time	176 sec	?

Intro

How do learned objects' system models improve global task planning for a robot with nonprehensile push manipulation abilities over time?

Research Subquestions:

1. How to combine learning and planning for push and drive applications?
2. How does the proposed framework compare against the state-of-the-art?

Intro

Assumtions

1. **Closed-World**

Intro

Assumtions

1. **Closed-World**
2. **Perfect Object Sensor**

Assumtions

1. **Closed-World**
2. **Perfect Object Sensor**
3. **Tasks are Commutative**

Required Background

System Models

1. LTI drive model

Required Background

System Models

1. LTI drive model
2. LTI push model

Required Background

System Models

1. LTI drive model
2. LTI push model
3. Nonlinear push model

Required Background

System Models

1. LTI drive model
2. LTI push model
3. Nonlinear push model

Example

$$x(k+1) = f(x(k), u(k))$$

Required Background

Control Methods

1. Model Predictive Control (MPC)
2. Model Predictive Path Intergral (MPPI) control

Required Background

Driving

1. (MPC, *lti-drive-model*)
2. (MPPI, *lti-drive-model*)

Required Background

Driving

1. (MPC, *lti-drive-model*)
2. (MPPI, *lti-drive-model*)

Pushing

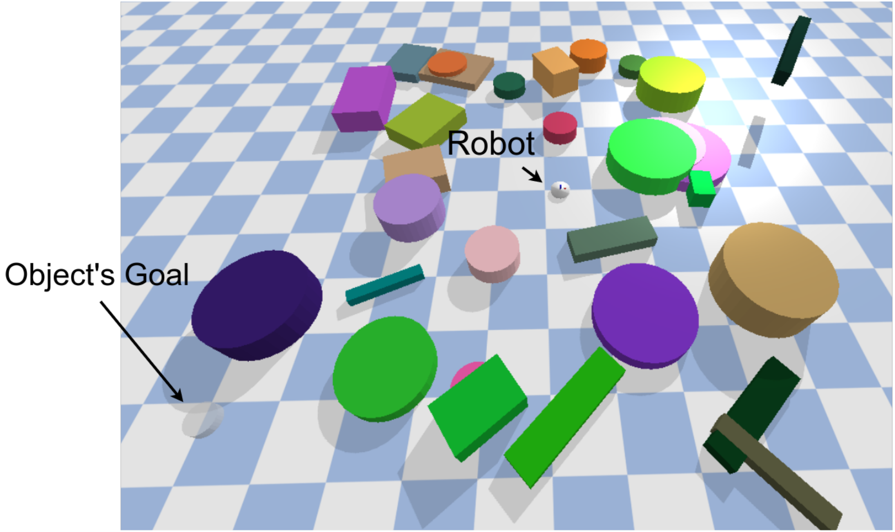
1. (MPPI, *lti-push-model*)
2. (MPPI, *nonlinear-push-model*)

Required Background

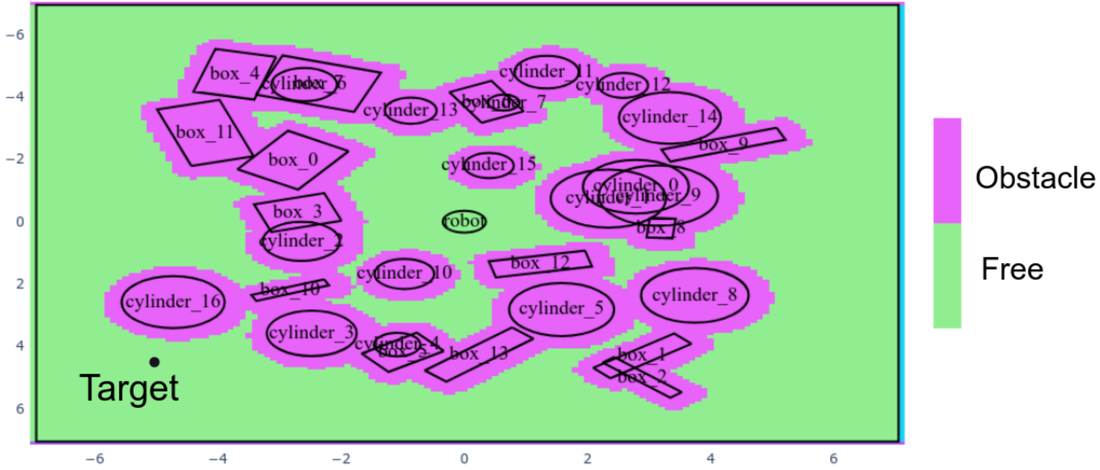
Find a Path

1. Path Estimation
2. Path Planning

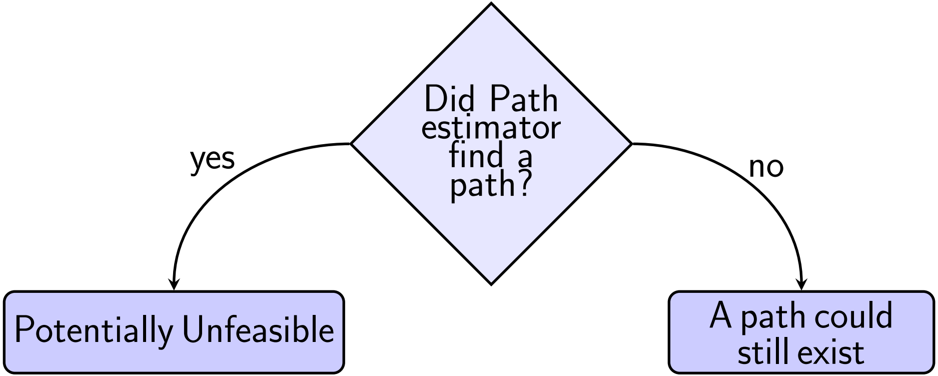
Required Background



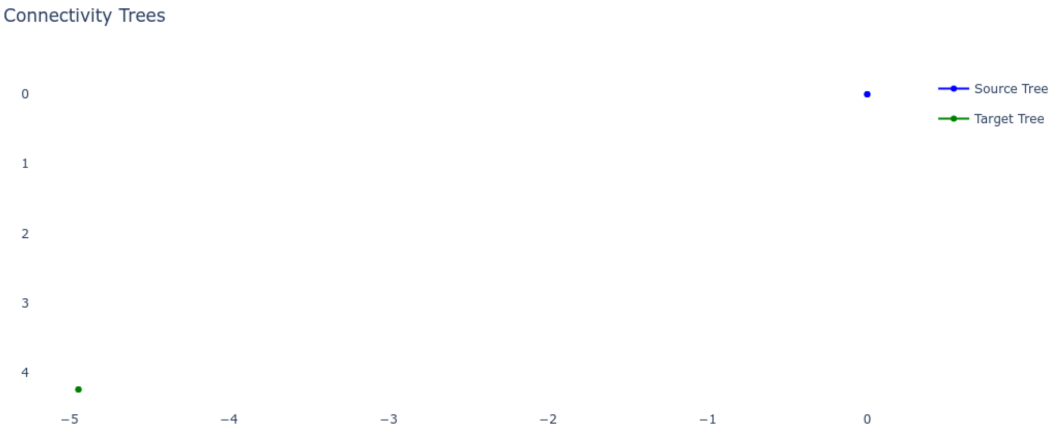
Required Background



Required Background



Required Background



Required Background

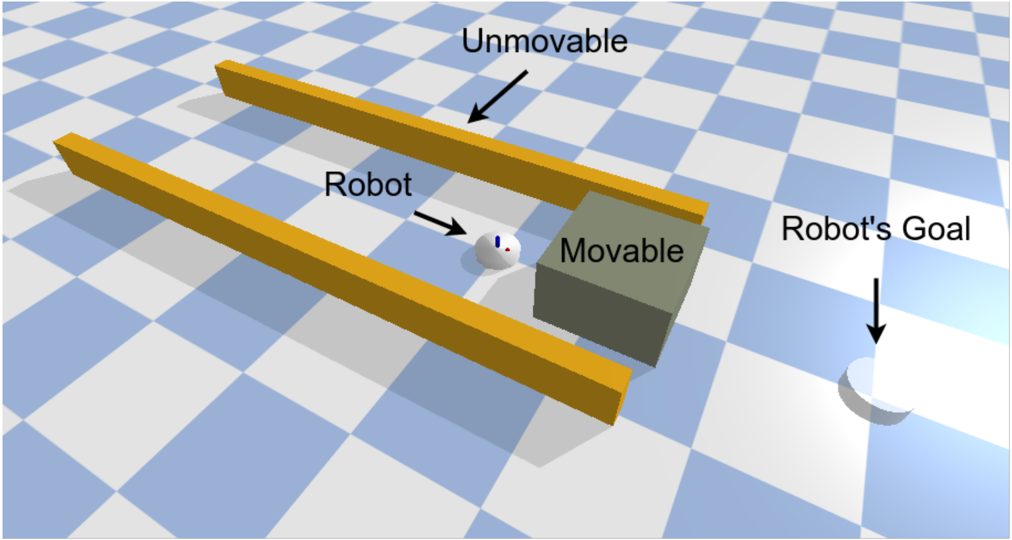
Required Background

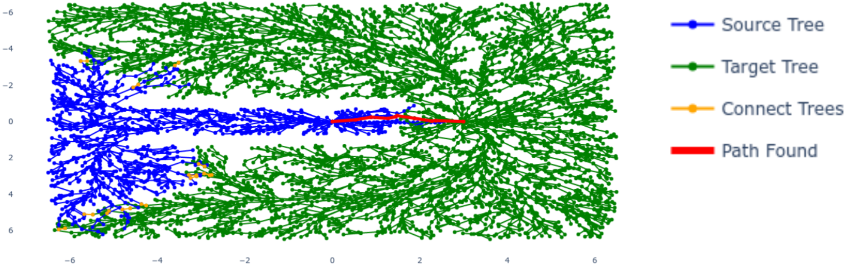
$$Cost_{path} = \sum_{i=1}^{n-1} Distance(c_i, c_{i+1})$$

Proposed method

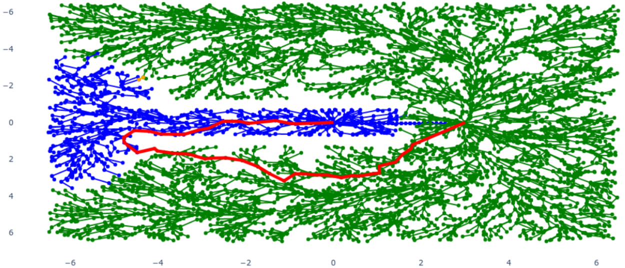
$$Cost_{path} = \sum_{i=1}^{n-1} Distance(c_i, c_{i+1})$$
$$+MovableSpaceCost + UnknownSpaceCost$$

Proposed Method





$$MovableSpaceCost = 0.5m$$



$$MovableSpaceCost = 1.0m$$

Proposed Method

Hypothesis Graph (H-Graph)

- Consists of nodes and edges
- Represent propabilistic search in action space

Proposed Method

Hypothesis Graph (H-Graph)

- Consists of nodes and edges
- Represent propabilistic search in action space

Hypothesis Algorithm (H-Algorithm)

- Search for hypotheses that complete tasks
- Execute hypotheses

Proposed Method

Hypothesis Graph (H-Graph)

- Consists of nodes and edges
- Represent propabilistic search in action space

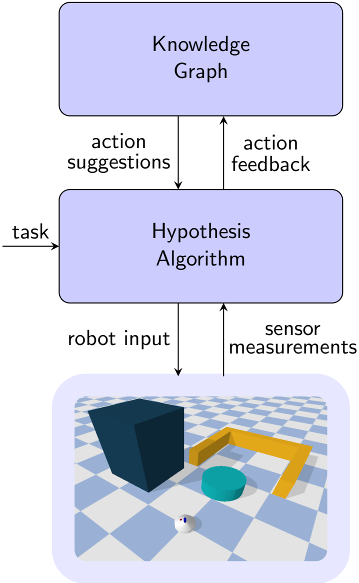
Hypothesis Algorithm (H-Algorithm)

- Search for hypotheses that complete tasks
- Execute hypotheses

Knowledge Graph (K-Graph)

- Store object's class
- Store action feedback
- Suggest actions

Proposed Method



Proposed Method

$$G^{hypothesis} = \langle V_H, E_H \rangle$$

Proposed Method

$$G^{hypothesis} = \langle V_H, E_H \rangle$$

Nodes:

$V_H = \{v_1, v_2, \ldots, v_n\}$

Proposed Method

$$G^{hypothesis} = \langle V_H, E_H \rangle$$

Nodes:

$V_H = \{v_1, v_2, \ldots, v_n\}$

Edges:

$E_H = \{e_1, e_2, \ldots, e_m\}$

$m, n \geq 0$

Proposed Method

$$G^{hypothesis} = \langle V_H, E_H \rangle$$

Nodes:

$V_H = \{v_1, v_2, \ldots, v_n\}$

$$v_{id} = \langle obj, c(k) \rangle$$

Edges:

$E_H = \{e_1, e_2, \ldots, e_m\}$

$m, n \geq 0$

Proposed Method

A **identification edge**:

$$e_{id}^{iden} = \langle id_{from}, id_{to}, \text{ Identification method}, \text{ IO data set}, \text{ controller}, \text{ system model}, \text{ status}\rangle$$

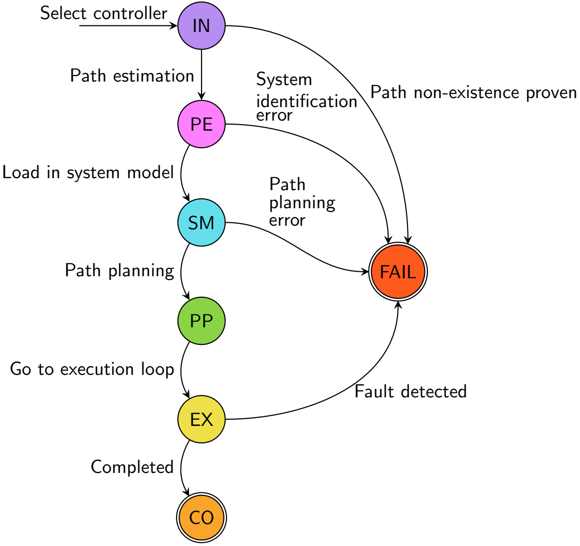
A **action edge**:

$$e_{id}^{action} = \langle id_{from}, id_{to}, \text{verb}, \text{ controller}, \text{ system model}, \text{ path}, \text{ status}\rangle$$

A **empty edge**:

$$e_{id}^{emtpy} = \langle id_{from}, id_{to}, \text{status}\rangle$$

Proposed Method



Proposed Method

Drive edge parameterizations

- (MPC, *lti-drive-model*)
- (MPPI, *lti-drive-model*)

Proposed Method

Drive edge parameterizations

- (MPC, *lti-drive-model*)
- (MPPI, *lti-drive-model*)

Push edge parameterizations

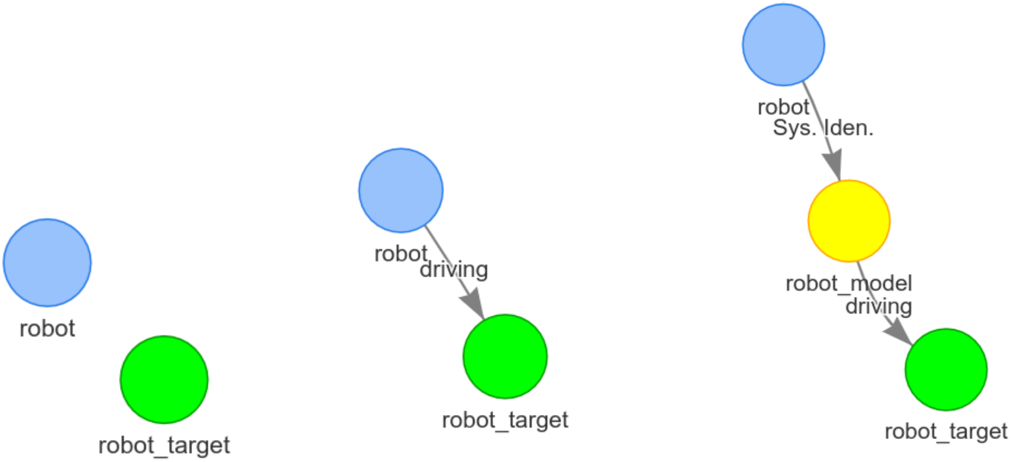
- (MPPI, *lti-push-model*)
- (MPPI, *nonlinear-push-model*)

Proposed Method

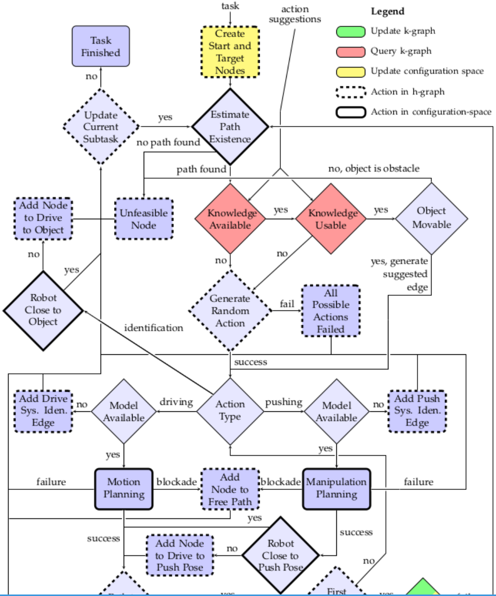
Hypothesis Algorithm (H-Algorithm)

- Search for hypotheses that complete tasks
- Execute hypotheses

Proposed Method

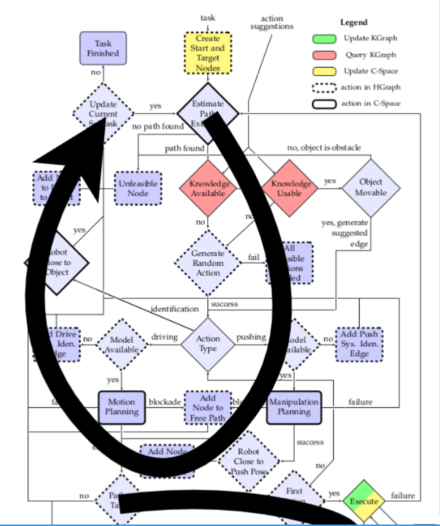


Proposed Method

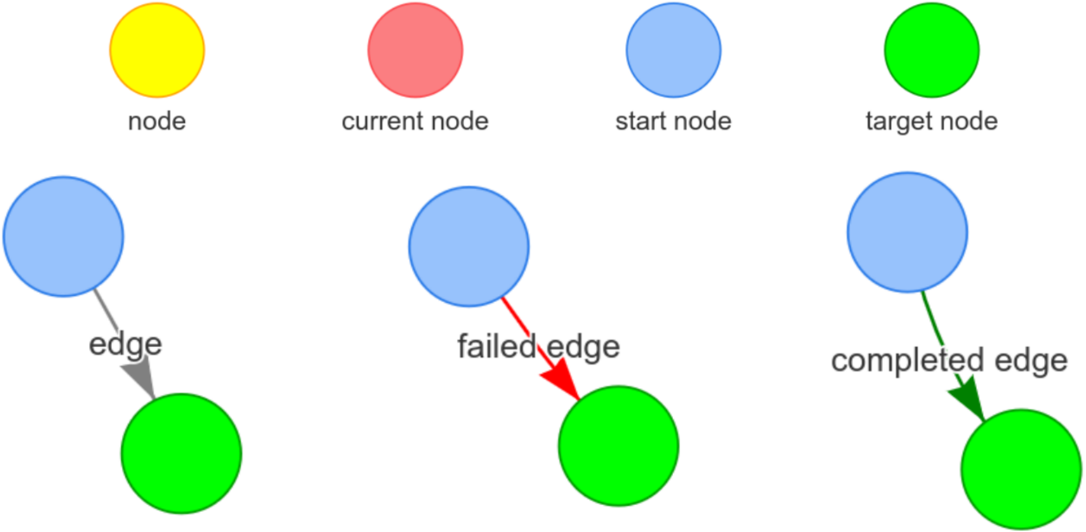


Proposed Method

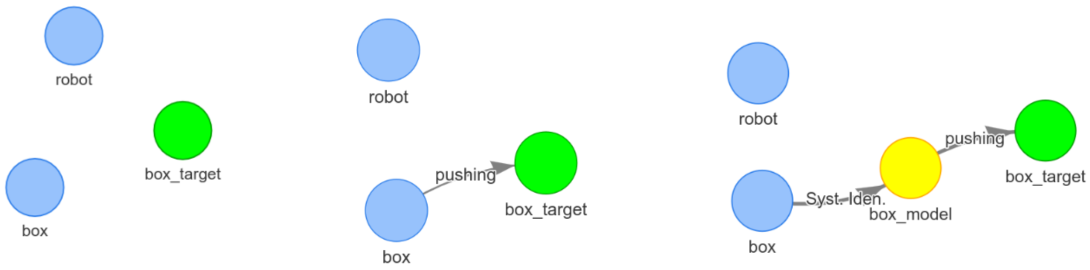
Search Loop



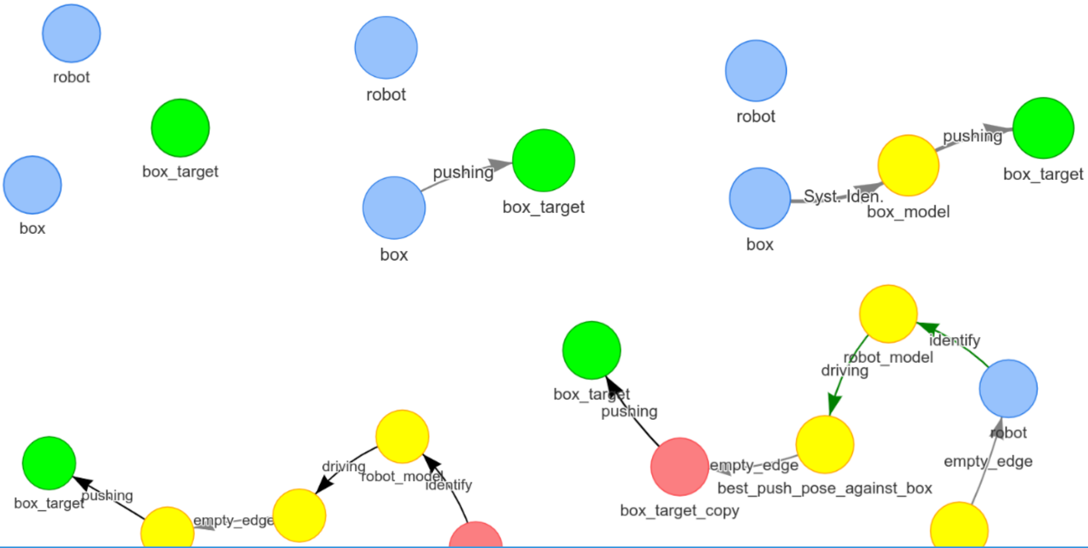
Proposed Method



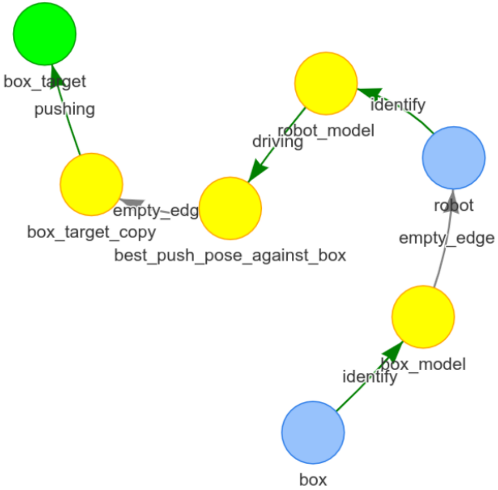
Proposed Method



Proposed Method



Proposed Method



Proposed Method

Halgorithm behaviour

- Fault detection → fail edge
- Blocking obstacle → free path
- Stop regeneration of failed edges → blocklist

Proposed Method

Knowledge Graph (K-Graph)

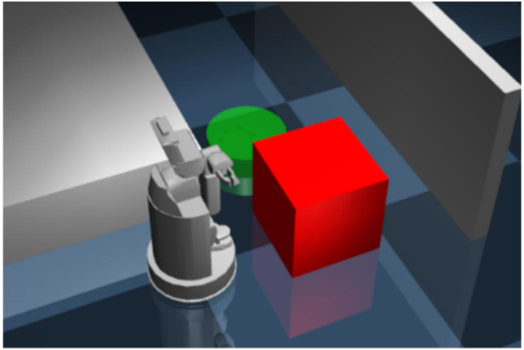
- Store object's class
- Store action feedback
- Suggest actions

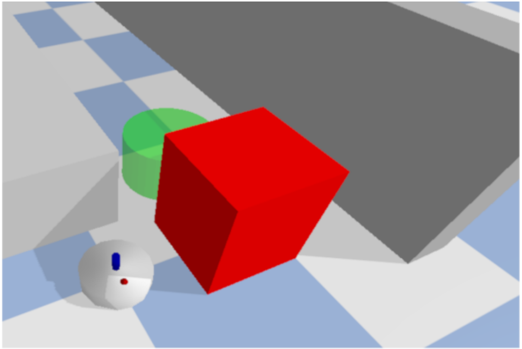
Proposed Method

Success Factor

$$\alpha(a+1) = \begin{cases} 0.1^{\epsilon^{pred}{}_{\text{avg}}} \\ 0.1 + 0.9\alpha(a) \\ 0.9\alpha(a) \end{cases}$$

Results





Results

Author	Wang et al.	Groote
search time	109 sec	?
execution time	67 sec	?
total time	176 sec	?

Results

Author	Wang et al.	Groote
search time	109 sec	26 sec
execution time	67 sec	?
total time	176 sec	?

Results

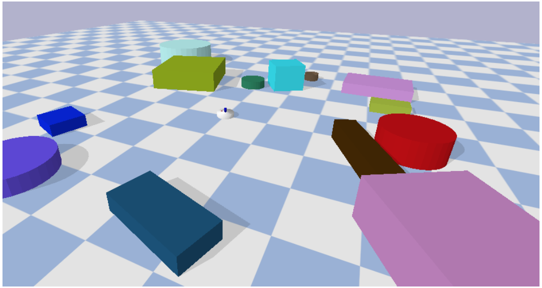
Author	Wang et al.	Groote
search time	109 sec	26 sec
execution time	67 sec	4 sec
total time	176 sec	?

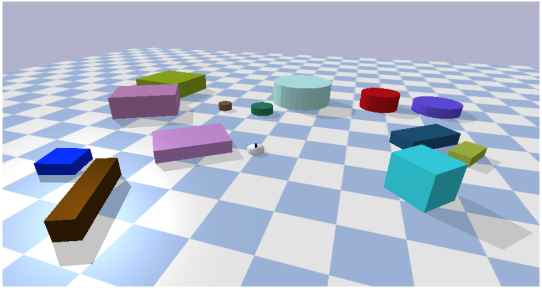
Results

Author	Wang et al.	Groote
search time	109 sec	26 sec
execution time	67 sec	4 sec
total time	176 sec	30 sec

Results

make targets and arrows to targets please



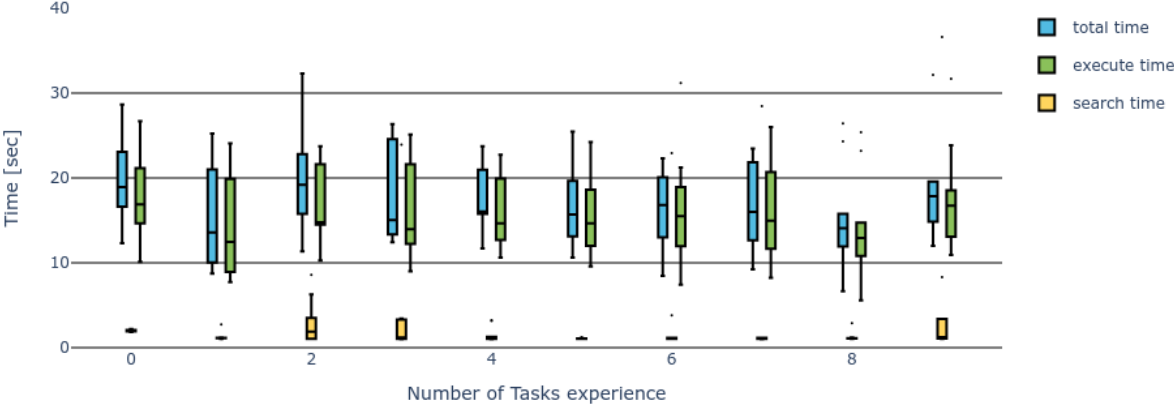


Results

make this visual that there are 300 subtasks to solve

$$\text{number of subtasks} = 10 * 10 * 3 = 300$$

Results

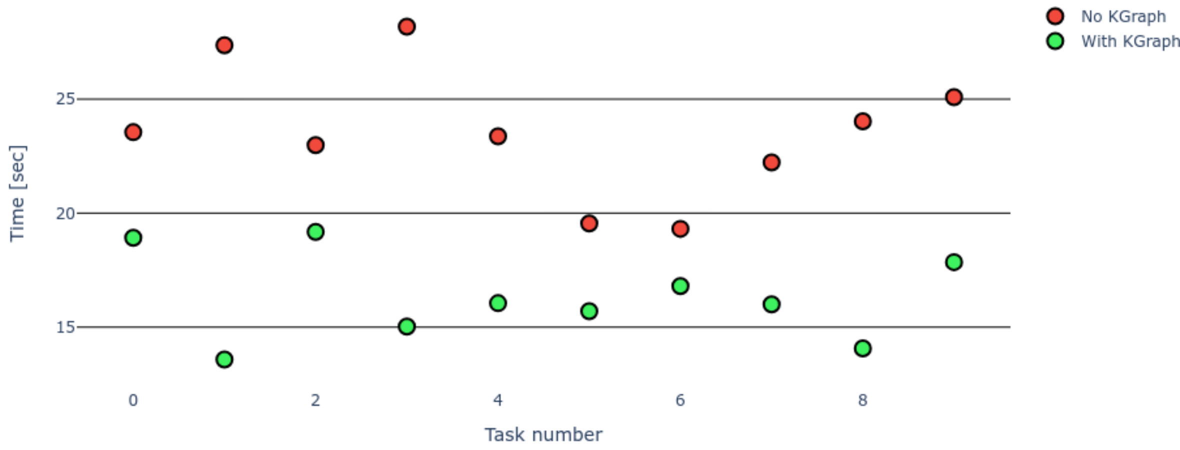


40
30
20
10
0
Time [sec]
0
2
4
6
8
Number of Tasks experience
total time
execute time
search time

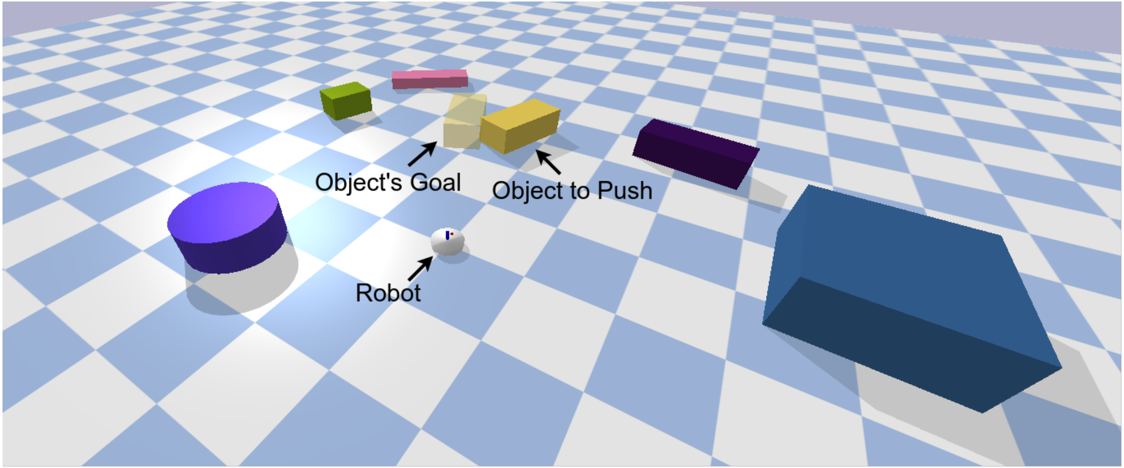
Results

Number of Tasks in experience		0	1	2	3	4	5	6	7	8	9
With k-graph suggestions	Number of MPC parameterizations	20	30	31	31	30	30	31	30	30	33
	Number of MPPI parameterizations	10	0	0	0	0	0	0	0	0	0
Without k-graph suggestions	Number of MPC parameterizations	12	14	13	10	15	16	13	17	16	9
	Number of MPPI parameterizations	18	17	18	20	17	15	17	13	15	22

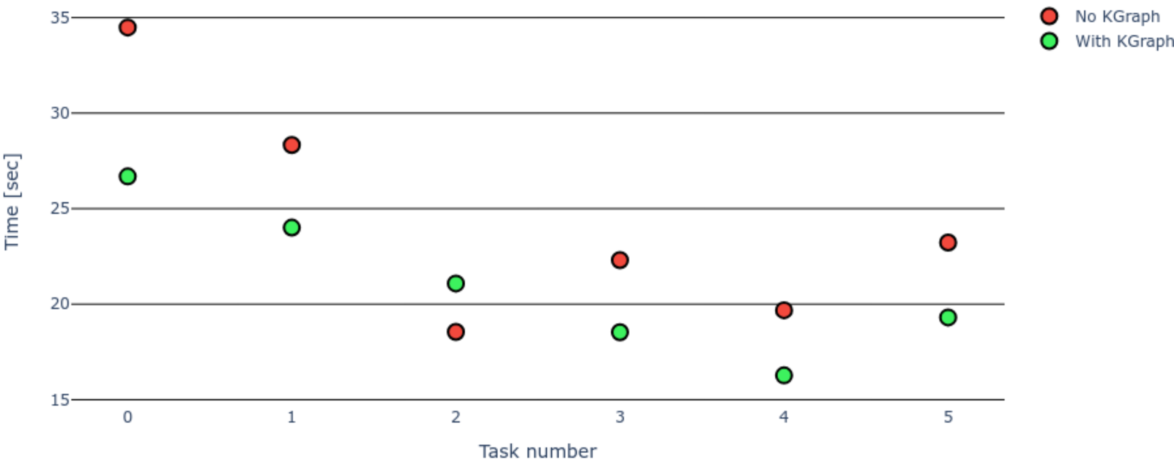
Results



Results

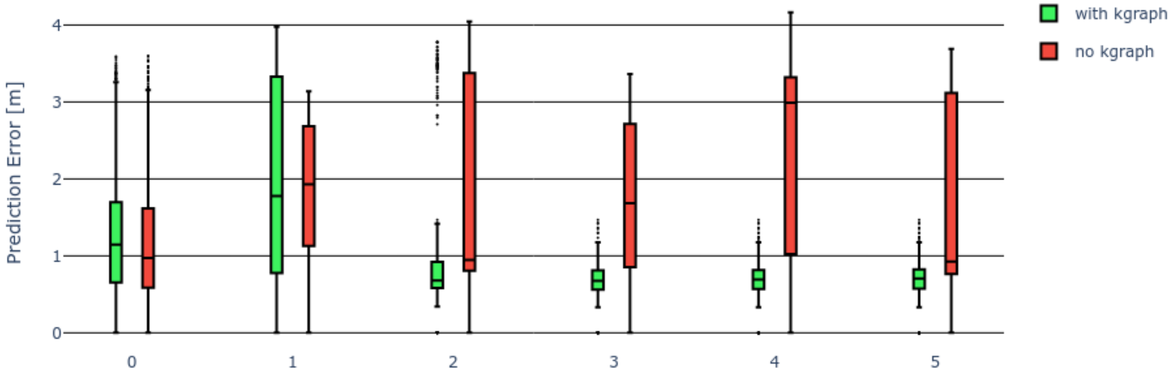


Results



No KGraph
With KGraph
35
30
25
20
15
Time [sec]
0
1
2
3
4
5
Task number

Results



Conclusion

System Models

1. Party combine three topics

Conclusion

System Models

1. Party combine three topics
2. Edge parameterizations

Conclusion

System Models

1. Party combine three topics
2. Edge parameterizations
3. Nonlinear push model

Conclusion

How do learned objects' system models improve global task planning for a robot with nonprehensile push manipulation abilities over time? **Research Subquestions:**

1. How to combine learning and planning for push and drive applications?
2. How does the proposed framework compare against the state-of-the-art?

Conclusion

Questions?