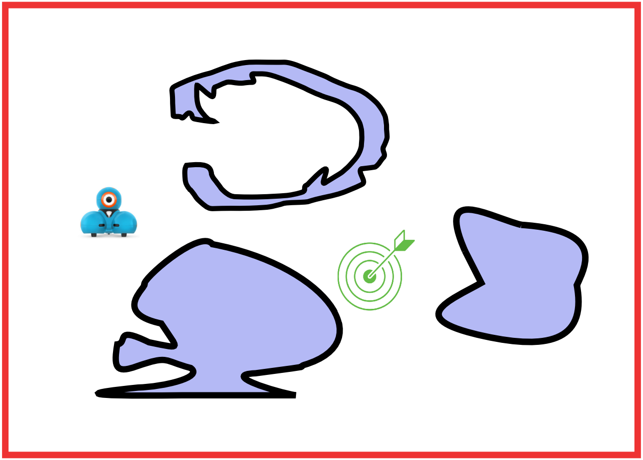
Counterexample Guided Inductive Optimization Applied to Mobile Robot Path Planning

SBR/LARS 2017

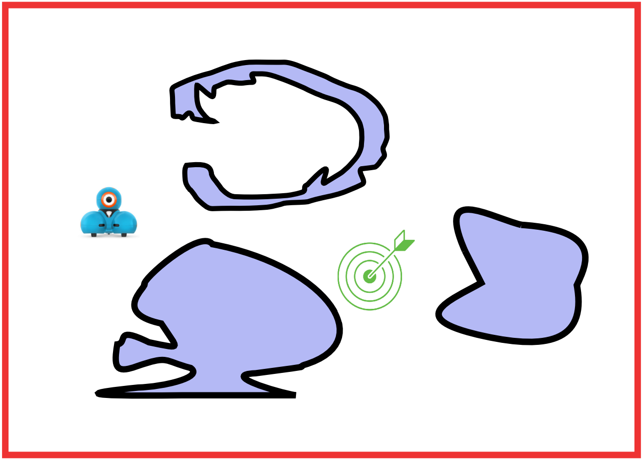
Rodrigo Araújo, Alexandre Ribeiro, **Iury Bessa**,
Lucas Cordeiro, and João Edgar Chaves Filho

Federal University of Amazonas,
University of Oxford,
Federal University of Minas Gerais

Motivation

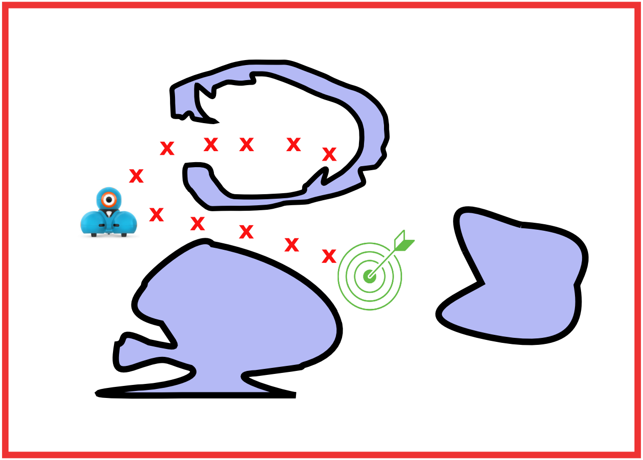


Motivation



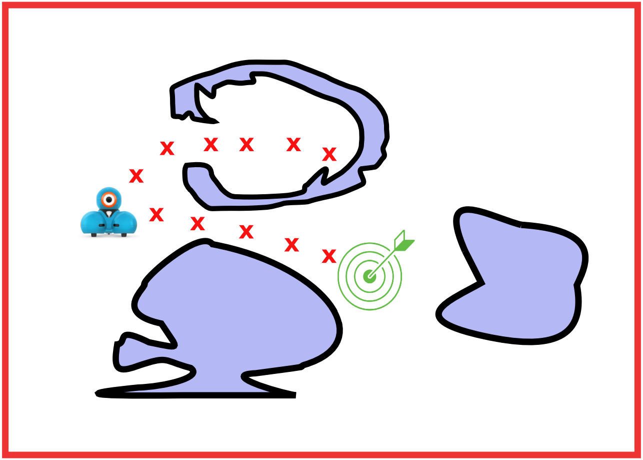
How to find a path from the starting point to the target point?

Motivation



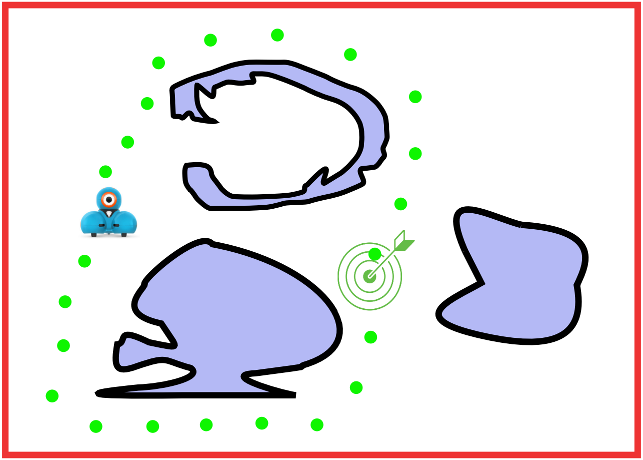
Path must meet safety constraints (e.g., obstacle avoidance)

Motivation



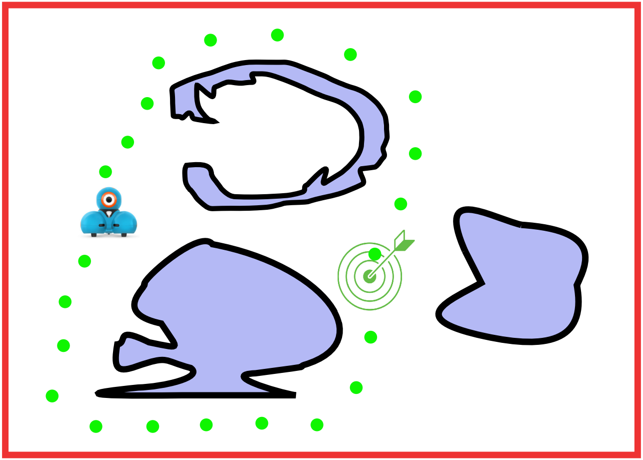
It is the objective of the path planning task

Motivation



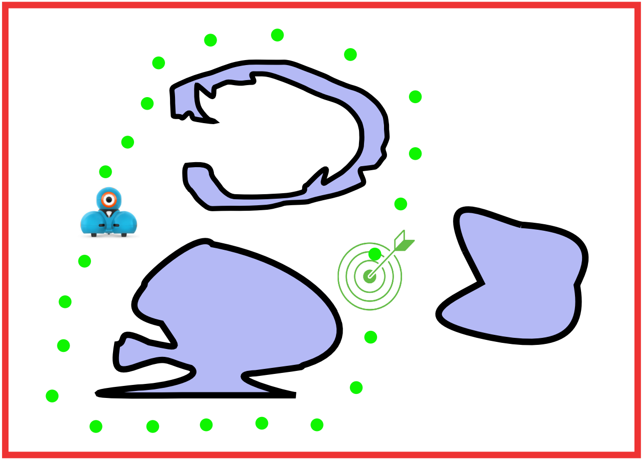
How to find the best path that meets the constraints?

Motivation



The path must be evaluated w.r.t. a cost function (e.g., distance and energy waste)

Motivation



It is the objective of **optimal path planning**

Objectives

Apply Counterexample Guided Inductive Optimization (CEGIO) to mobile robot optimal path planning

Objectives

Apply Counterexample Guided Inductive Optimization (CEGIO) to mobile robot optimal path planning

- Encode the environment, movement space, and static obstacles

Objectives

Apply Counterexample Guided Inductive Optimization (CEGIO) to mobile robot optimal path planning

- Encode the environment, movement space, and static obstacles
- Parametrize the path by using the coordinates of path points and its respective orientations

Objectives

Apply Counterexample Guided Inductive Optimization (CEGIO) to mobile robot optimal path planning

- Encode the environment, movement space, and static obstacles
- Parametrize the path by using the coordinates of path points and its respective orientations
- To find the shortest path that satisfies the constraints given by the problem

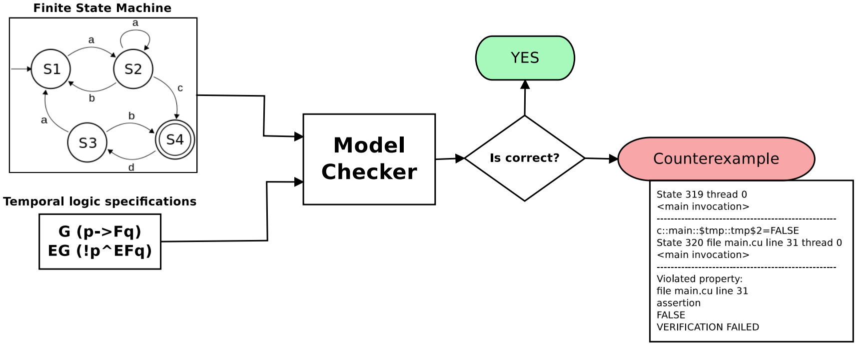
Model checking

Traditional path planning methodologies (e.g. APF- and GA-based algorithms) cannot ensure the global path optimality.

Model checking

CEGIO optimization ensures the global optimization because it is based on model checking procedures.

Model checking



Modeling the optimization problem using a model checker

The directives `ASSUME` and `ASSERT` must be employed for modeling optimization problems

- `ASSUME`:
- `ASSERT`:

Modeling the optimization problem using a model checker

The directives ASSUME and ASSERT must be employed for modeling optimization problems

- ASSUME: is used for modeling the knowledge about the problem and the constraints set
- ASSERT:

Modeling the optimization problem using a model checker

The directives `ASSUME` and `ASSERT` must be employed for modeling optimization problems

- `ASSUME`: is used for modeling the knowledge about the problem and the constraints set
- `ASSERT`:is used for holding the global optimization condition $l_{optimal}$

$$l_{optimal} \leftrightarrow f(x) > f_p \tag{1}$$

Modeling the optimization problem using a model checker

The directives `ASSUME` and `ASSERT` must be employed for modeling optimization problems

- `ASSUME`: is used for modeling the knowledge about the problem and the constraints set
- `ASSERT`:is used for holding the global optimization condition $l_{optimal}$

$$l_{optimal} \leftrightarrow f(x) > f_p \tag{1}$$

- Decision variables are defined as non-deterministic integers that represents rationals with desired precision

Modeling the optimization problem using a model checker

The directives ASSUME and ASSERT must be employed for modeling optimization problems

- `ASSUME`: is used for modeling the knowledge about the problem and the constraints set
- `ASSERT`:is used for holding the global optimization condition $l_{optimal}$

$$l_{optimal} \leftrightarrow f(x) > f_p \tag{1}$$

- Decision variables are defined as non-deterministic integers that represents rationals with desired precision
- The verification engine is executed by iteratively increasing the precision and converging to the optimal solution

CEGIO-based Path Planning

The main steps of CEGIO-based Path Planning are:

CEGIO-based Path Planning

The main steps of CEGIO-based Path Planning are:

Step 1

Parametrize and encode the environment, movement space, and static obstacles

CEGIO-based Path Planning

The main steps of CEGIO-based Path Planning are:

Step 1

Parametrize and encode the environment, movement space, and static obstacles

Step 2

Formulate the cost function

CEGIO-based Path Planning

The main steps of CEGIO-based Path Planning are:

Step 1

Parametrize and encode the environment, movement space, and static obstacles

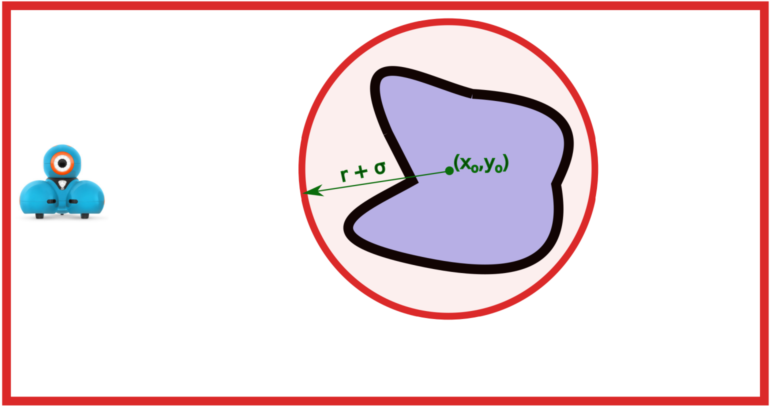
Step 2

Formulate the cost function

Step 3

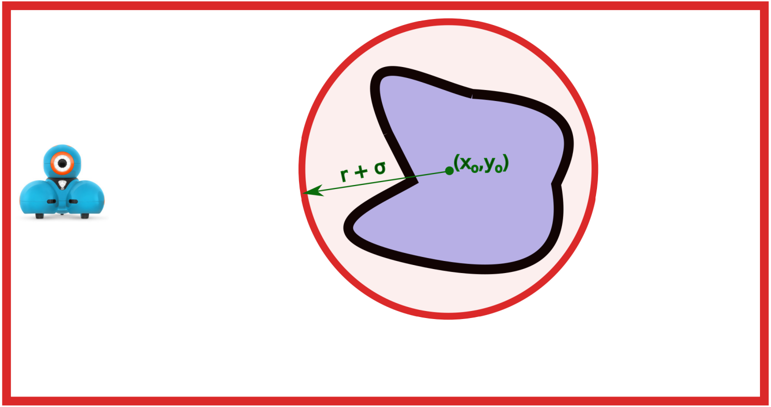
Parametrize the paths and find an optimal path that satisfies the constraints given by the problem

Environment Modeling



The search space is delimited by a rectangle

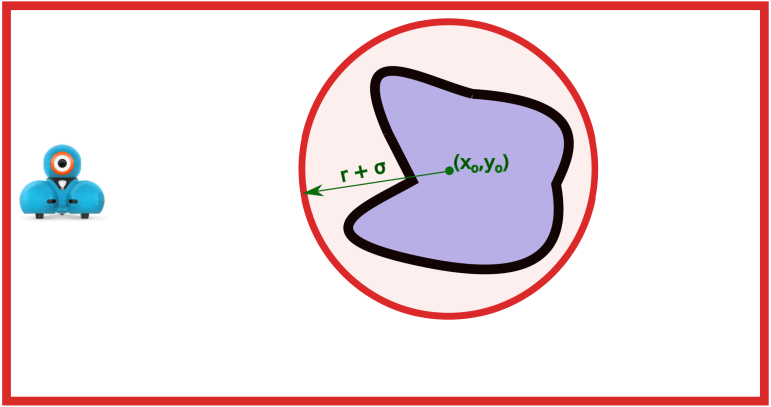
Environment Modeling



The obstacles are modeled as circles

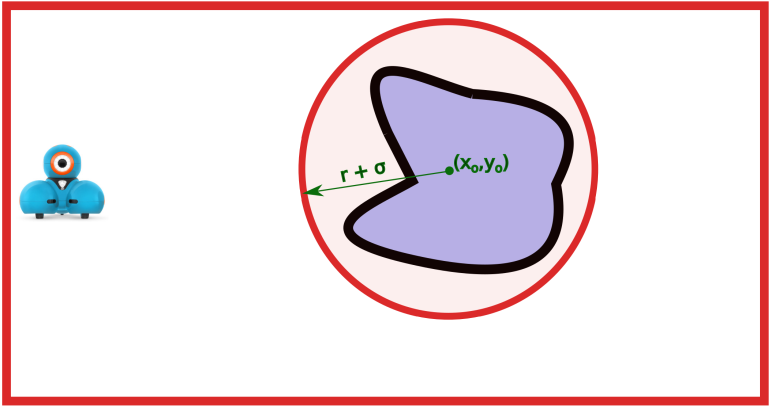
$$(x_{i\lambda} - x_0)^2 + (y_{i\lambda} - y_0)^2 \geq (r + \sigma)^2 \quad (2)$$

Environment Modeling

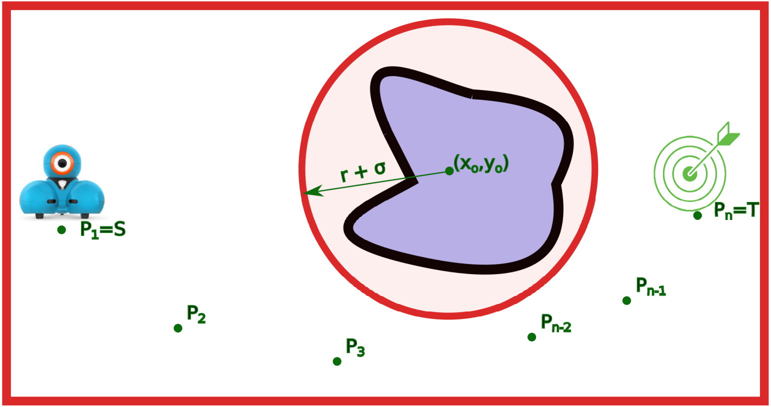


The constraints of the optimization problem must ensure that there is no intersection between the path and the obstacles

Path parametrization

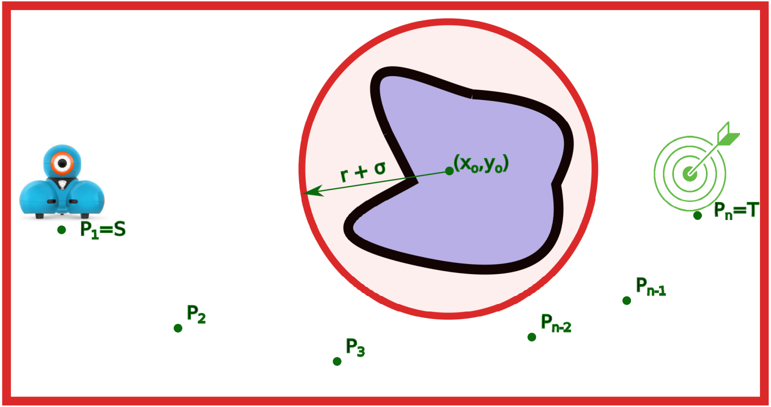


Path parametrization



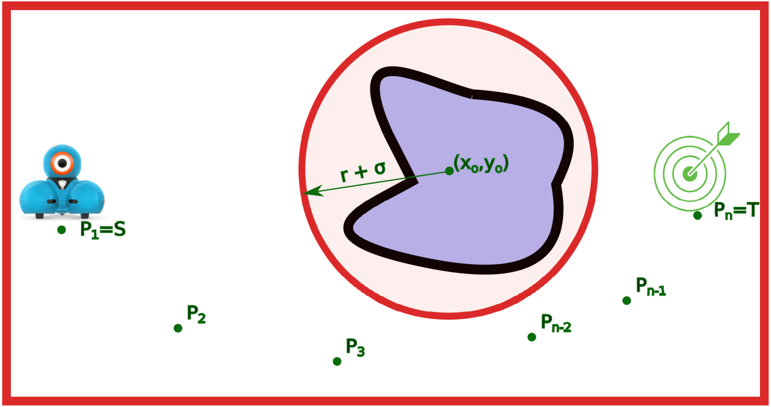
The bi-dimensional path has n **vertices** $(P_1, P_2, ..., P_n)$

Path parametrization



The path must start at the starting point $S = P_1$ and end at target point $T = P_n$

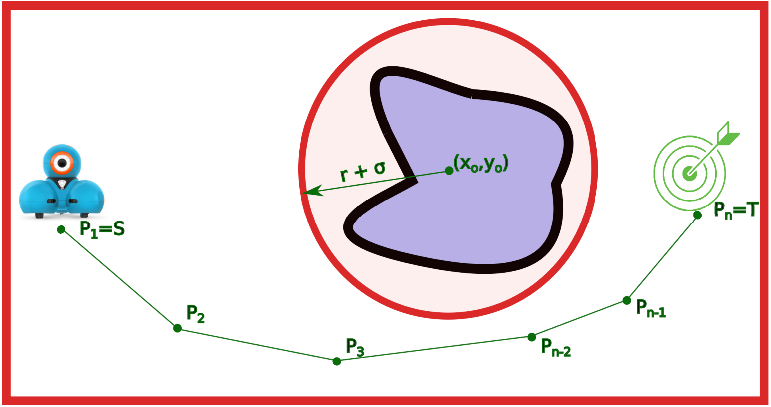
Path parametrization



The vertex matrix **L** is defined as follows.

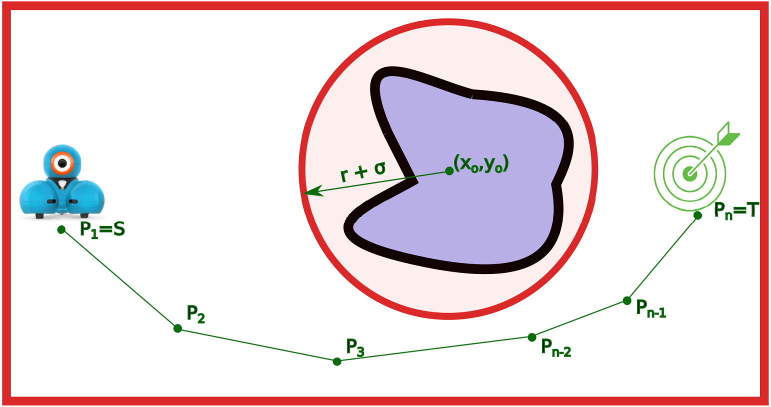
$$\mathbf{L} = [P_1, P_2, ..., P_{n-1}, P_n] \tag{2}$$

Path parametrization



The path is formed by $n-1$ straight segments. The i-th segment is built from P_i to P_{i+1}

Path parametrization



The set of points in the *i*-th segment is parametrized as follows for all $\lambda \in [0,1]$

$$p_{i\lambda}(L) = (1-\lambda)P_i + \lambda P_{i+1} \tag{2}$$

Path Optimization Problem

Path Optimization Problem

The cost function is defined as follows

$$J(L) = \sum_{i=1}^{n-1} \left\| P_{i+1} - P_i \right\|_2 , \tag{3}$$

Path Optimization Problem

The cost function is defined as follows

$$J(L) = \sum_{i=1}^{n-1} \|P_{i+1} - P_i\|_2 \,, \tag{3}$$

The resulting optimization problem is:

$$\begin{aligned} \min_{L} \quad & J(L), \\ \text{s.t.} \quad & p_{i\lambda}(L) \notin \mathbb{O} \\ & p_{i\lambda}(L) \in \mathbb{E} \\ & i = 1, ..., n-1, \end{aligned} \tag{4}$$

The model checking procedure checks the satisfiability of $J_{optimal}$:

$$J_{optimal} \leftrightarrow J(L) > J_c \tag{5}$$

CEGIO-based path planning algorithm

CEGIO-based path planning algorithm

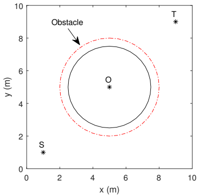
Input: Cost function $J(\mathbf{L})$, set of constraints Ω, desired precision η
Output: The optimal path and length ($\mathbf{L}^*$ and $J(\mathbf{L}^*)$)
Initialize $J(\mathbf{L}^{(0)})$ randomly, precision variable with $p = 1$, $k = 0$ e $i = 1$, number of points with $n = 1$
Declare $\mathbf{L}^i$ as non-deterministic integer vector
while $k \leq \eta$ **do**
    Find the best solution with the precision k
    $k = k + 1$
    Update the set Ω^k and the precision variable k
end while
$\mathbf{L}^* = \mathbf{L}^{(i)}$ and $J(\mathbf{L}^*) = J(\mathbf{L}^{(i)})$

CEGIO-based path planning algorithm

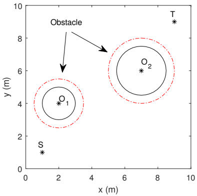
Find the best path with precision k

Define limits of $\mathbf{L}$ with directive `ASSUME`
Describe the objective function model $J(\mathbf{L})$
repeat
    `ASSUME`$(J(\mathbf{L}^{(i)}) < J(\mathbf{L}^{(i-1)}))$
    `ASSERT`$(J_{optimal})$
    Update $\mathbf{L}^* = \mathbf{L}^{(i)}$ and $J(\mathbf{L}^*) = J(\mathbf{L}^{(i)})$ based on the counterexample
    i=i+1;
until TRUE
if $J_{optimal}$ is not consecutively `SAT` **then**
    Break
else
    Update n
end if

Benchmarks and Experimental Settings



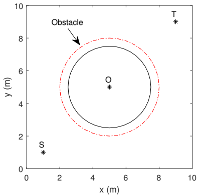
(a) Environment 1



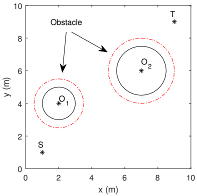
(b) Environment 2

Figure

Benchmarks and Experimental Settings



(a) Environment 1

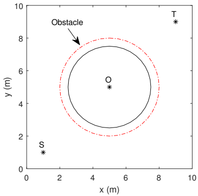


(b) Environment 2

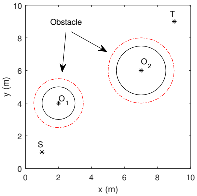
Figure

- The experiments were performed in the two above environments

Benchmarks and Experimental Settings



(a) Environment 1

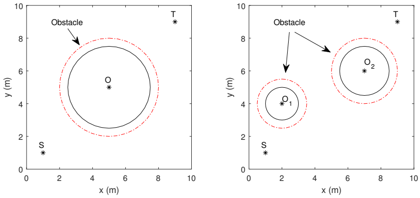


(b) Environment 2

Figure

- The experiments were performed in the two above environments
- All experiments were conducted on an otherwise idle Intel Core i7 4790 3.60 GHz processor, with 16 GB of RAM, running Ubuntu 14.10 64-bits

Benchmarks and Experimental Settings



(a) Environment 1 (b) Environment 2

Figure

- The experiments were performed in the two above environments
- All experiments were conducted on an otherwise idle Intel Core i7 4790 3.60 GHz processor, with 16 GB of RAM, running Ubuntu 14.10 64-bits
- The CBMC v4.5 with support to the MiniSAT v2.2.0 and ESBMC v3.1.0 with support to the MathSAT v5.3.13 were employed

Research Questions

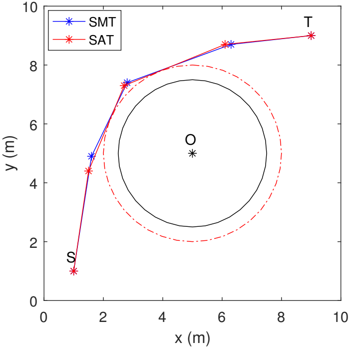
Research Questions

- **RQ1** Is it possible to apply CEGIO for robot mobile path planning?

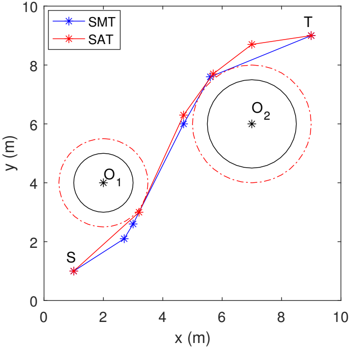
Research Questions

- **RQ1** Is it possible to apply CEGIO for robot mobile path planning?
- **RQ2** Which CEGIO parameters can be adjusted to obtain a good trade off between planning time and cost?

Experimental Results



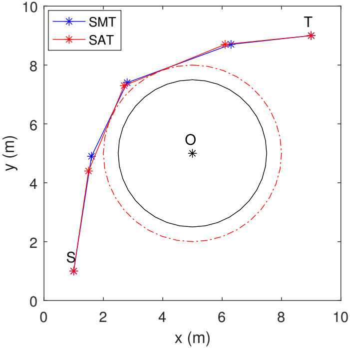
(a) Path for environment 1



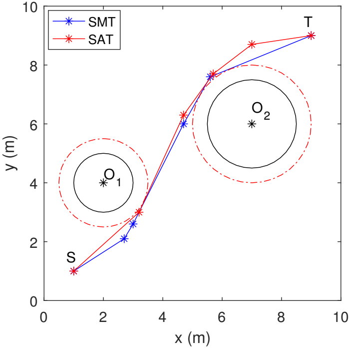
(b) Path for environment 2

Figure

Experimental Results



(a) Path for environment 1

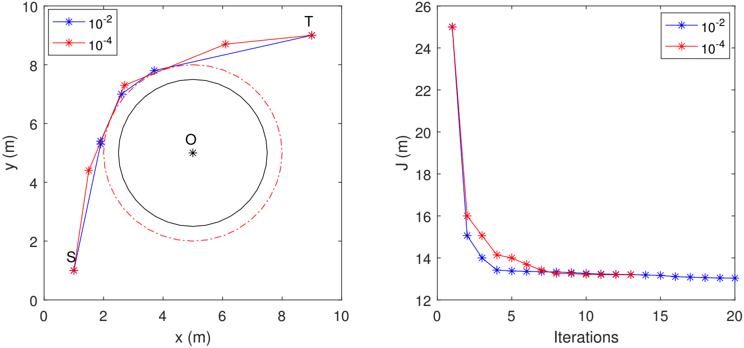


(b) Path for environment 2

Figure

- Optimal paths are obtained for both settings with 5 and 6 points respectively

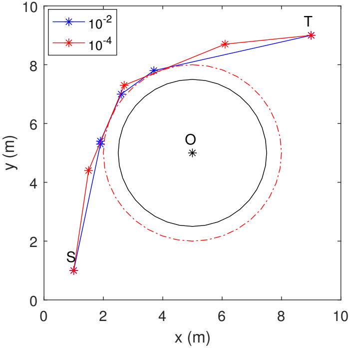
Experimental Results



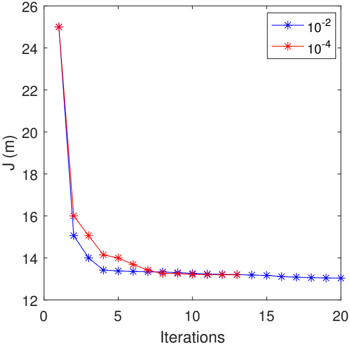
(a) (b)

Figure

Experimental Results



(a)



(b)

Figure

- The time spent is reduced to about 5% by reducing the precision from 10^{-4} to 10^{-2}

Conclusions

- The CEGIO is able to produce optimal paths for mobile robots

Conclusions

- The CEGIO is able to produce optimal paths for mobile robots
- CEGIO-based optimization was applied for optimal path planning in environments with multiple obstacles

Conclusions

- The CEGIO is able to produce optimal paths for mobile robots
- CEGIO-based optimization was applied for optimal path planning in environments with multiple obstacles
- CEGIO-based path planning presents high computational cost, however after few iterations, the cost becomes almost stationary

Conclusions

- The CEGIO is able to produce optimal paths for mobile robots
- CEGIO-based optimization was applied for optimal path planning in environments with multiple obstacles
- CEGIO-based path planning presents high computational cost, however after few iterations, the cost becomes almost stationary
- CEGIO-based path planning cost is highly dependent on precision variable

Conclusions

- The CEGIO is able to produce optimal paths for mobile robots
- CEGIO-based optimization was applied for optimal path planning in environments with multiple obstacles
- CEGIO-based path planning presents high computational cost, however after few iterations, the cost becomes almost stationary
- CEGIO-based path planning cost is highly dependent on precision variable
- The time can be reduced by adjusting the precision and breaking the optimization process when it achieves the steady state

Conclusions

- The CEGIO is able to produce optimal paths for mobile robots
- CEGIO-based optimization was applied for optimal path planning in environments with multiple obstacles
- CEGIO-based path planning presents high computational cost, however after few iterations, the cost becomes almost stationary
- CEGIO-based path planning cost is highly dependent on precision variable
- The time can be reduced by adjusting the precision and breaking the optimization process when it achieves the steady state
- Further studies include the usage of multi-objective optimization and applications to UAVs