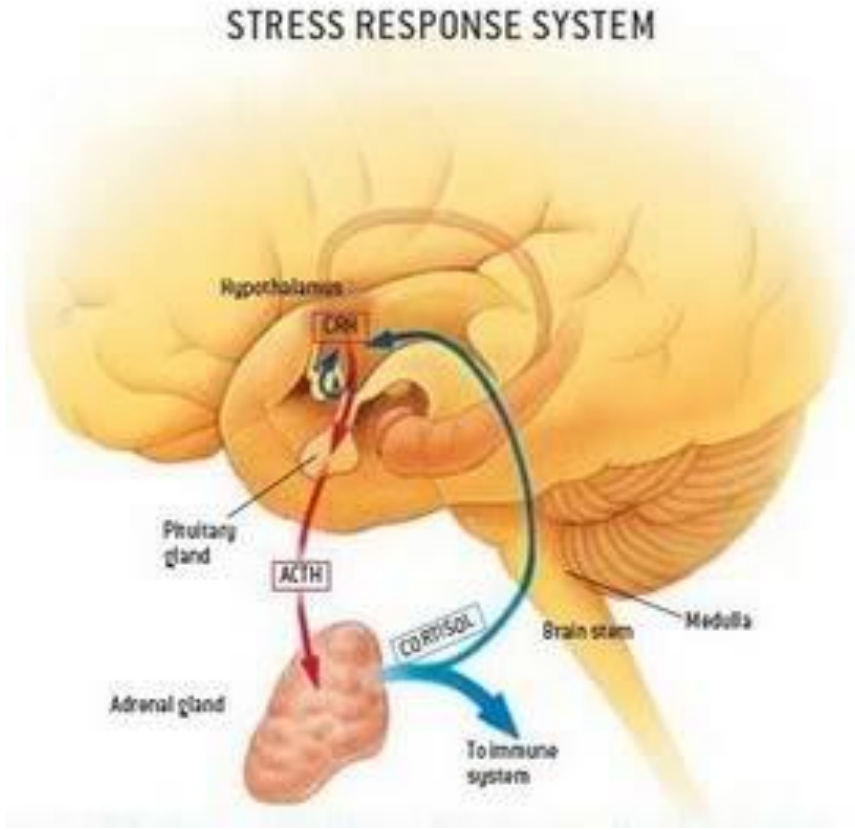
Stress-inspired working memory

Biological Stress-Inspired Cognitive Intelligence

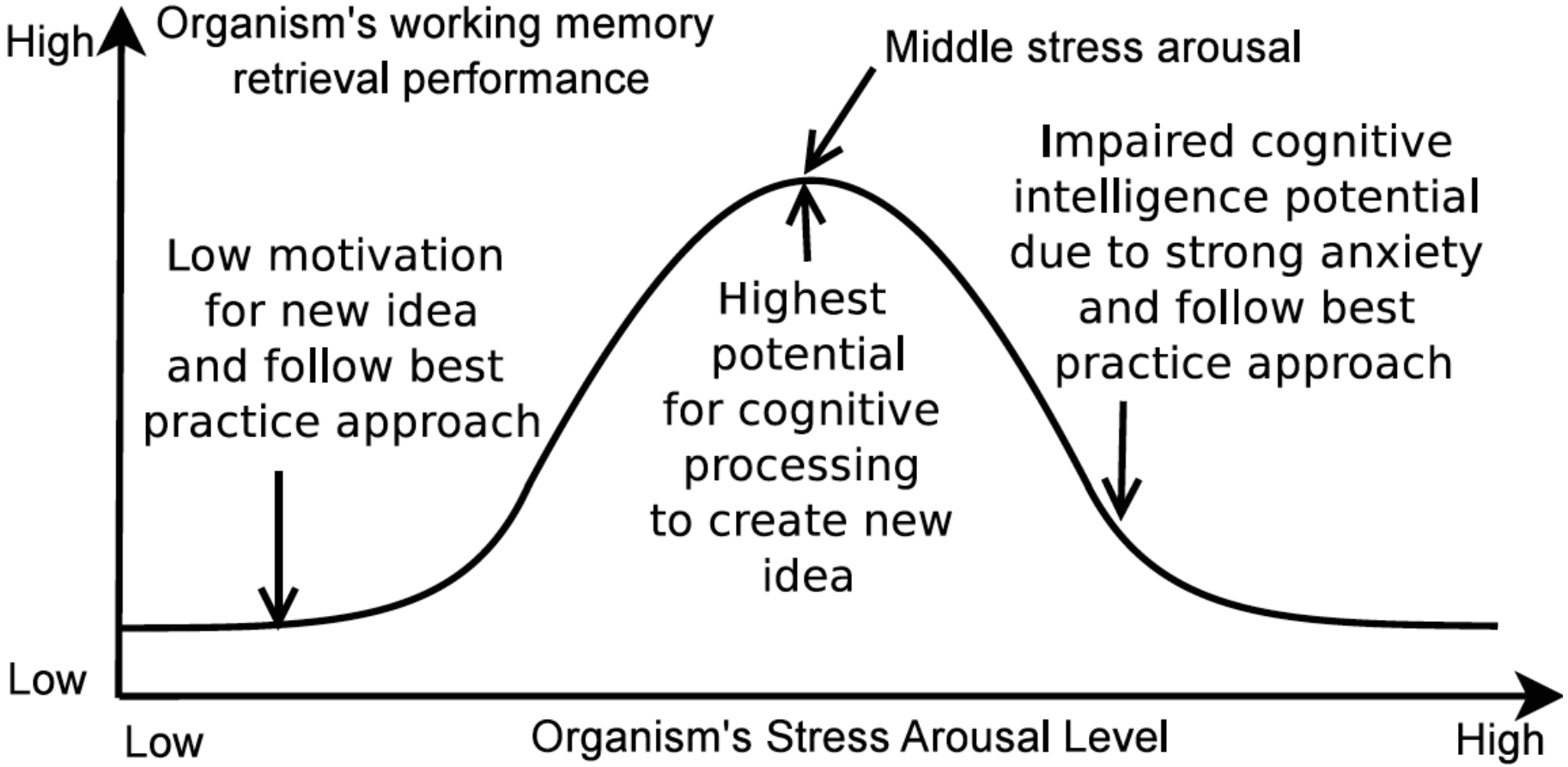
- In order to simulate artificial cognitive intelligence, we investigate human biological stress models.
- Why stress model?
  - Stress model is directly related to cognitive performance according to Yerkes Dodson law of stress.
  - During middle stress condition, human exhibit full cognitive intelligence to solve problems.
  - It is also related to organism's memory retrieval performance according to recent biology research.

Biological Stress Model

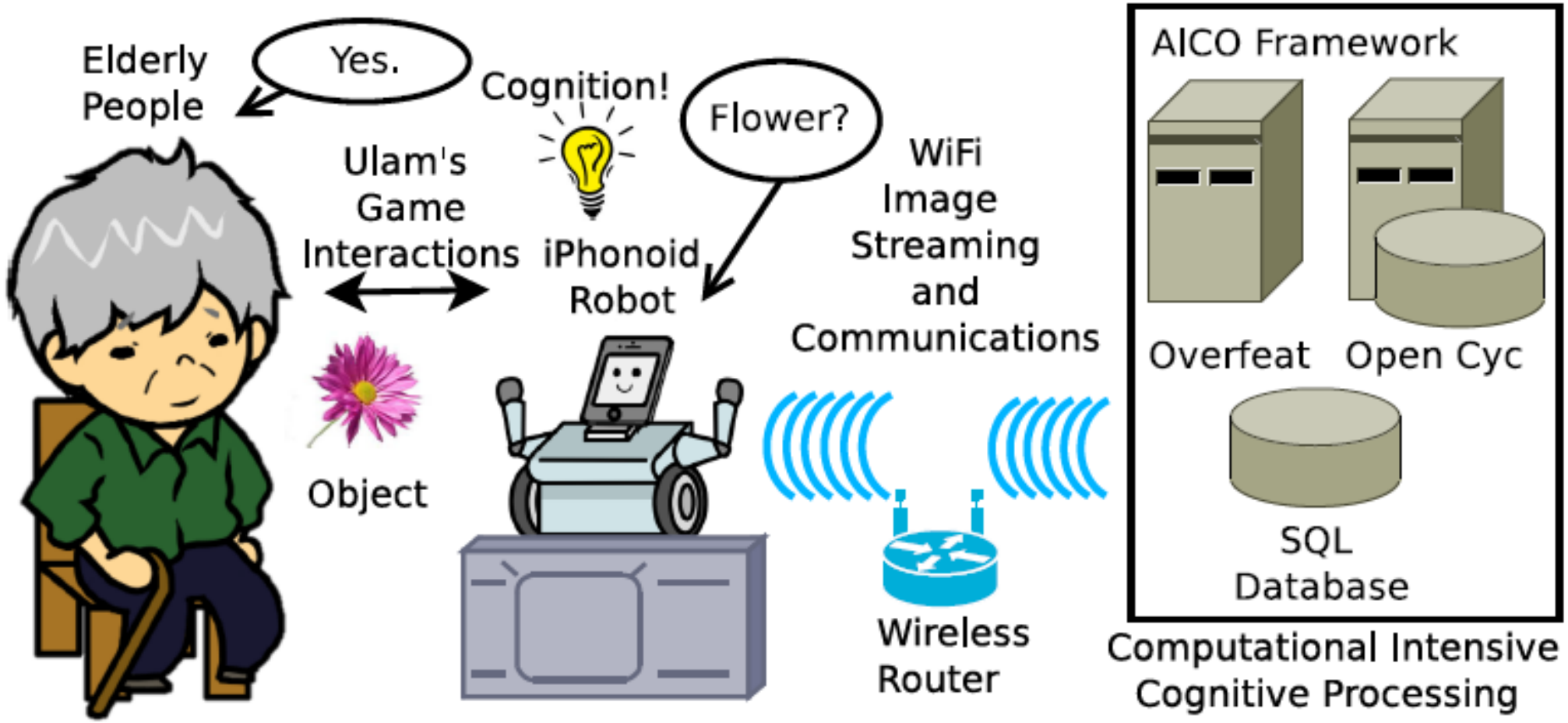
- Hypothalamus Pituitary Adrenal (HPA) axis has many features
- Stress hormones production
- Overdrive the body for higher metabolism (escape from danger)
- Forms new long-term memory
- Biological stress system is a problem-solving reactions to unknown situations.
- Changes the organism's working memory retrieval performance (Focus of this research)



Biological Stress-Inspired Cognitive Intelligence



Biological Stress-Inspired Working Memory Optimization



Biological Stress-Inspired Working Memory Optimization

- The scope is the optimization of robot partner working memory.
- The working memory retrieval performance (c_{wm}) based on the stress arousal level (h_{gc}):

$$c_{wm} = \exp\left(-\frac{(h_{gc} - \mu)^2}{2\sigma^2}\right)$$

  where μ is the mean and σ is the standard deviation of the robot partner's stress model configurations.

- The h_{gc} value is proportional to the number of wrong answers (n_{wa}) answered by the human user ($h_{gc}=n_{wa}/20$)

Biological Stress-Inspired Working Memory Optimization

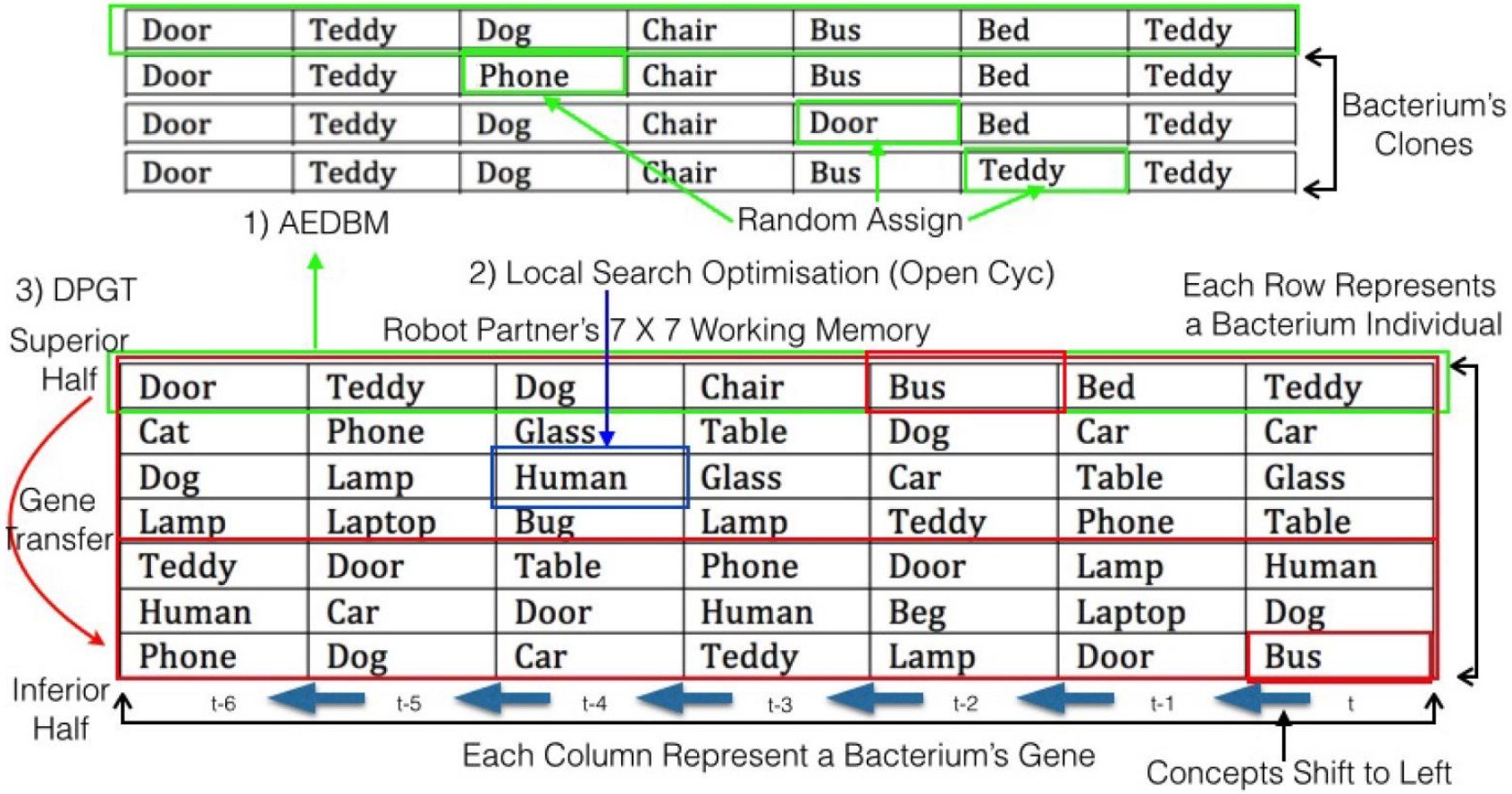
- During low stress condition $n_{wa} \approx 0$, the robot partner has confidence in the game with minimum optimization on its working memory (less thinking)
- During middle stress condition $n_{wa} \approx 10$, the robot partner is alerted of the game difficulties and started to optimize its working memory in a holistic way
- However during high stress situation, if the robot partner knows it is going to lose the game (wrong answer is about to reach $n_{wa} \approx 20$), the robot shows lack of interest to guess for correct answers act as its frustration behavior.

Dynamic Bacterial Memetic Algorithm

1: **procedure** DBMA
2: Update Working Memory X as bacterial population
3: $generation \leftarrow 0$
4: **while** $generation \neq N_{gen}$ **do**
5: **for** $row = 1 \ldots rounded(N_{bac} \cdot c_{wm})$ **do**
6: $AEDBM(X_{row})$
7: **for** $row = 1 \ldots rounded(N_{bac} \cdot c_{wm})$ **do**
8: $LocalSearch(X_{row})$
9: $DPGT(X)$
10: $generation \leftarrow generation + 1$

- Working memory: population
- Row: a bacterium
- Column: a gene of a bacterium
- Newly detected concept from Overfeat will be assigned to time column t

Dynamic Bacterial Memetic Algorithm



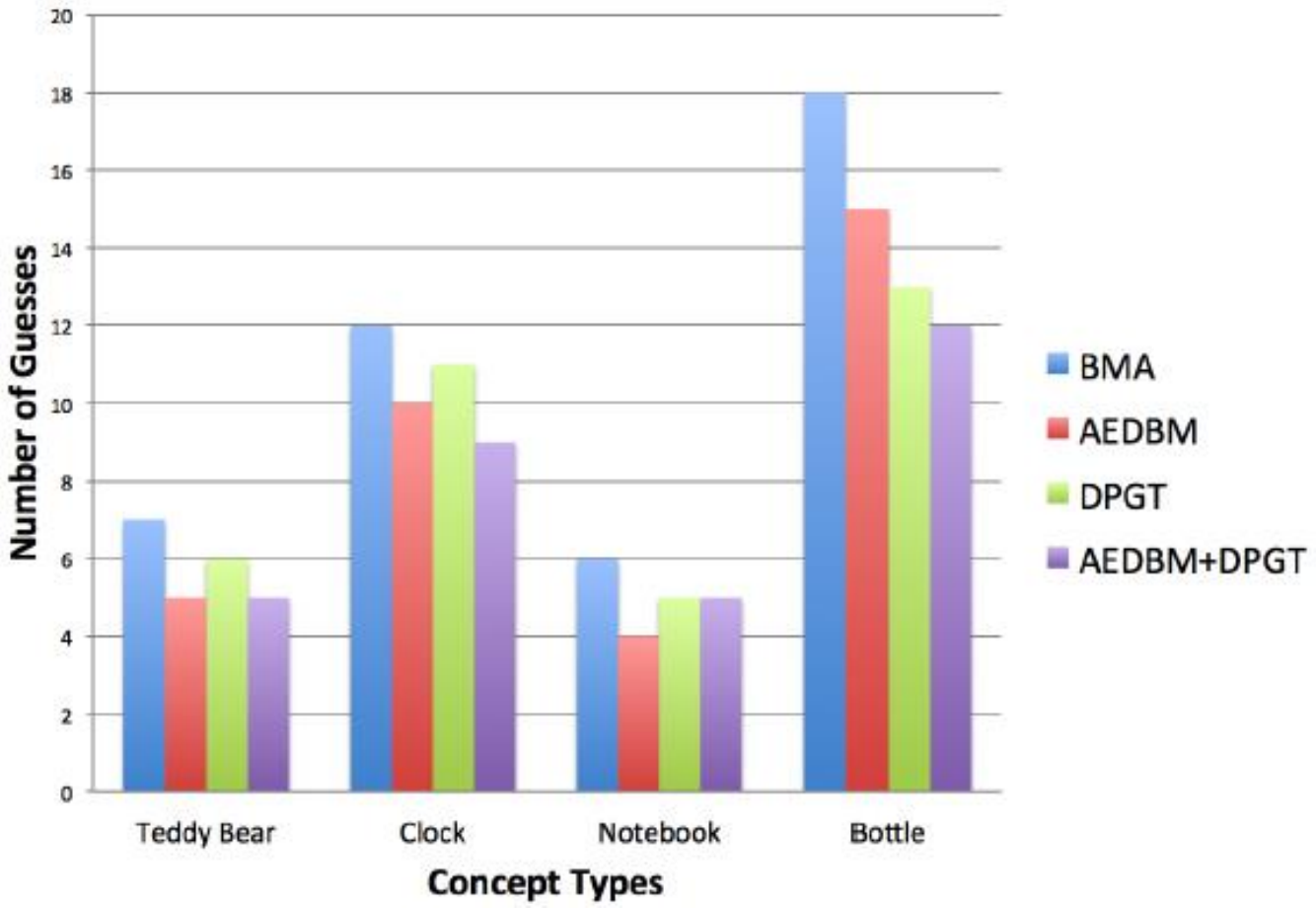
Experiment Setup

- Modified Rényi-Ulam guessing game.
- The robot partner iPhonoid is given image concept detection clues.
- The robot partner needs to guess the object's concept within 20 guesses only.
- The iPhonoid speaks the guessed concept with the speaker.
- Four object concepts were tested in the experiment.
- Tested with background as image concept detection noise.
- User needs to answer 'correct' or 'wrong' replies.

TABLE I. PARAMETER SETTING FOR THE PROPOSED ALGORITHM

N_{gen}	N_{clones}	N_{inf}	α	N_{search}	μ	σ
10	3	3	0.7	5	0.5	4

Experiment Result



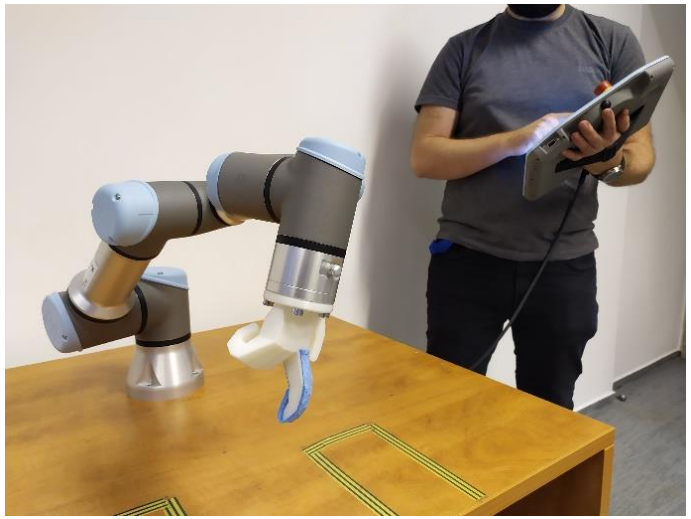
Discussions

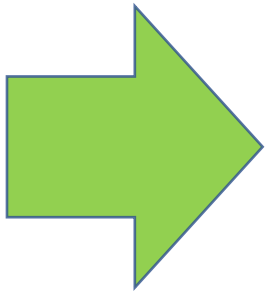
- Dynamic working memory optimization is important to robot partner's guessing ability.
- The background noise increases the image concept detection difficulty. It is necessary to include image concept detection in the game.
- The robot partner's working memory is useful to optimize the answers of the Rényi-Ulam guessing game.
- However, in order to have real time human-robot interactions the working memory cannot be too large.
- Hence, the working memory retrieval performance needed to be adjusted according the difficulty of the image concept detection. The robot partner's stress is the guessing game difficulty gauge meter.
- Increased working memory optimization can lead to higher human-robot interactions.

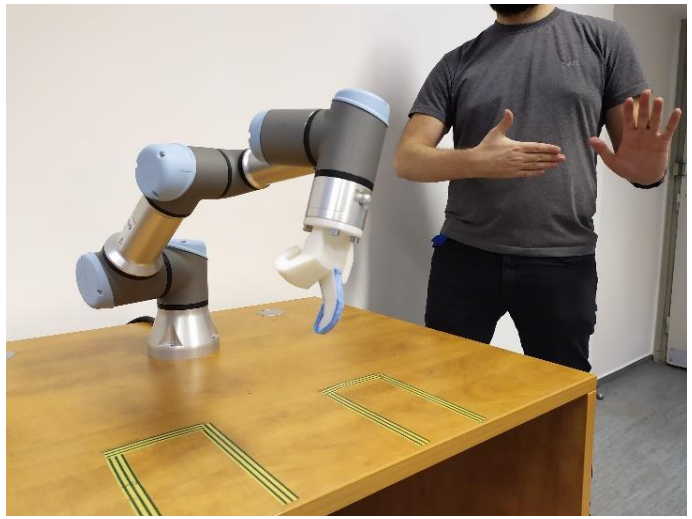
Robot pianist

Human-Robot Interaction

- Recent interests in HCI and HRI research
- More „human” like communication is the better
- Moved from info-communicational to socio-communicational channel

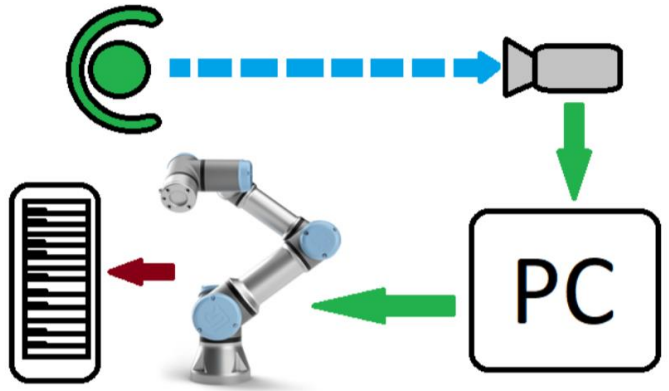




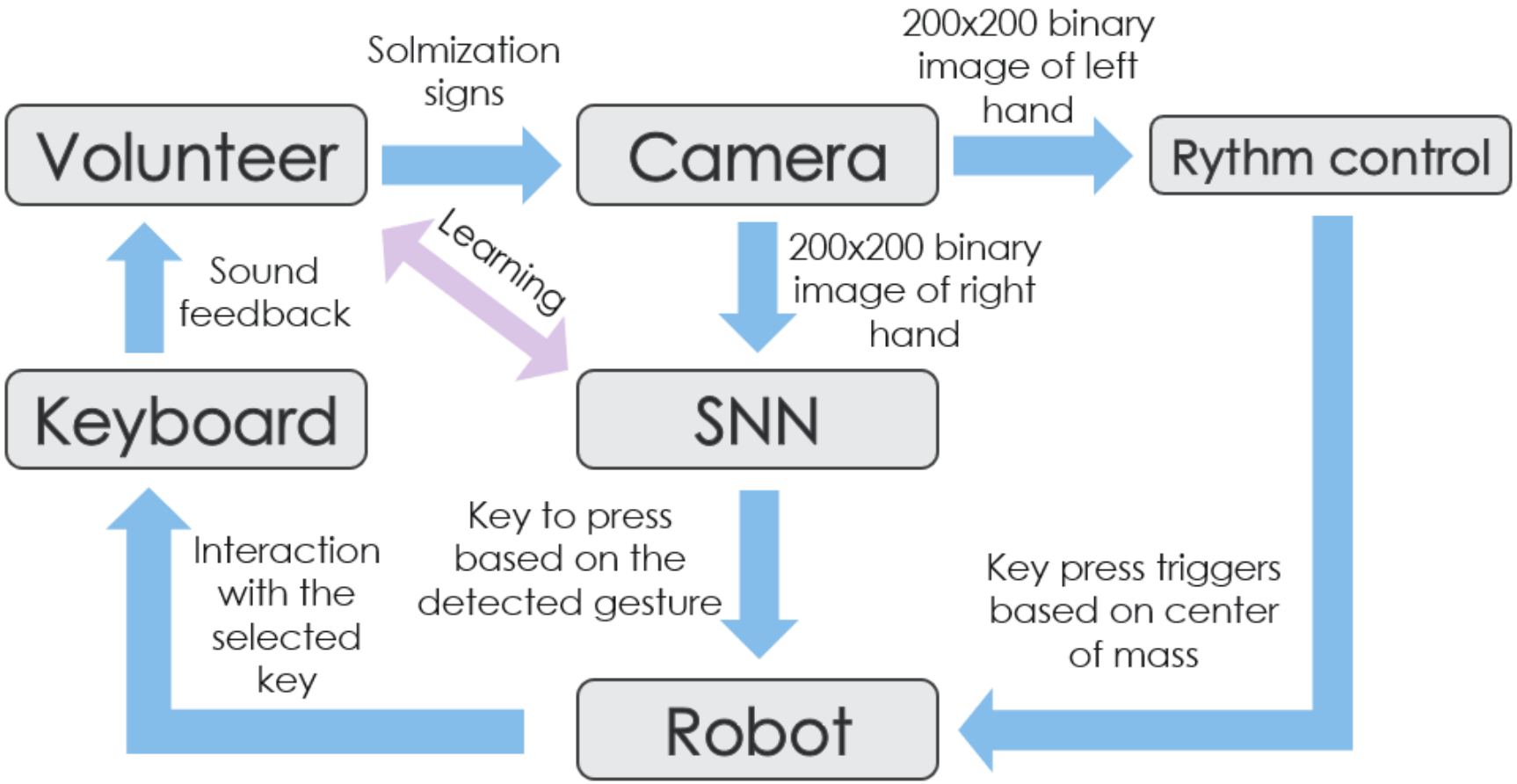


Application – UR3e

- Industrial cobot arm
  - Standards have safety features
- Low power, blocks immediately on resistance
- Can be moved to either end of the piano's playing space, can be taught, intermediate keys interpolated linearly
- Inputs from camera, one hand for rhythm, one for soloing signals

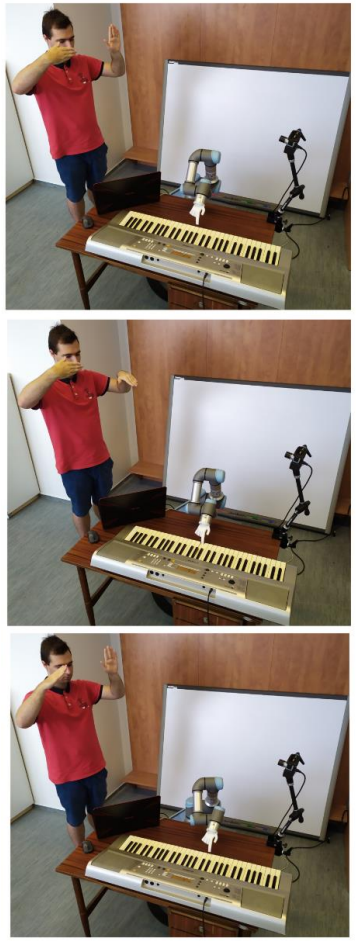


Application system design



Solmization based application

- The network is trained on 20 images per gesture
- Binary images generated via color segmentation and thresholding from a region of interest
- The rhythm is generated via the vertical movement of the left hand
- Testing on 100 images from the same volunteer the network's average accuracy reached 93%
- Fast retraining (3 epochs, 1 minute) made it possible to use it in education and aiding singers



Solmization based application

- Discrete simulation of Non-Leaky I&F SNN
- Hebbian-LMS learning without gradients
  - Classic learning rule of Perceptrons with Hebbian extension
- Recognized category indicated by first spike (the least delay)
- Fast convergence → real-time retraining

$$h^M[t] = h^M[t-1] + h^{SYN}[t] + h^{EXT}[t]$$

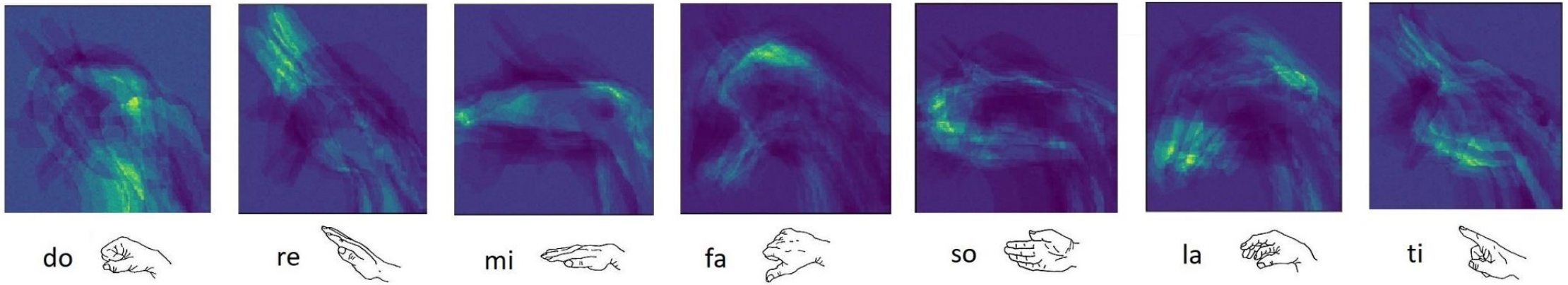
$$\Delta w_{nm} = \gamma \cdot e_m \cdot x_{nm}$$

$$e_m = \hat{t}_m - \hat{t}_{target}$$

$$x_{nm} = \begin{cases} \sum_{t=0}^{t_{max}} s_n(t) \cdot w_{nm}(t) = \sum_{t=0}^{t_{max}} \delta(t, \hat{t}_n) \cdot w_{nm}(t) = \\ = w_{nm}(\hat{t}_n) & \text{if } \hat{t}_n < \hat{t}_m \\ a^- & \text{if } \hat{t}_m < \hat{t}_n < t_{max} + 1 \\ 0 & \text{otherwise} \end{cases}$$

Solmization based application

- Precise fitting to the training set works in our favor
- Average recognition time is 0.06s
- Learned weights for each output neuron are visualized with the assigned hand gestures
- The network determines the category of the shown images based on differences



Playing a Hungarian Folk Song

Evolutionary robotics

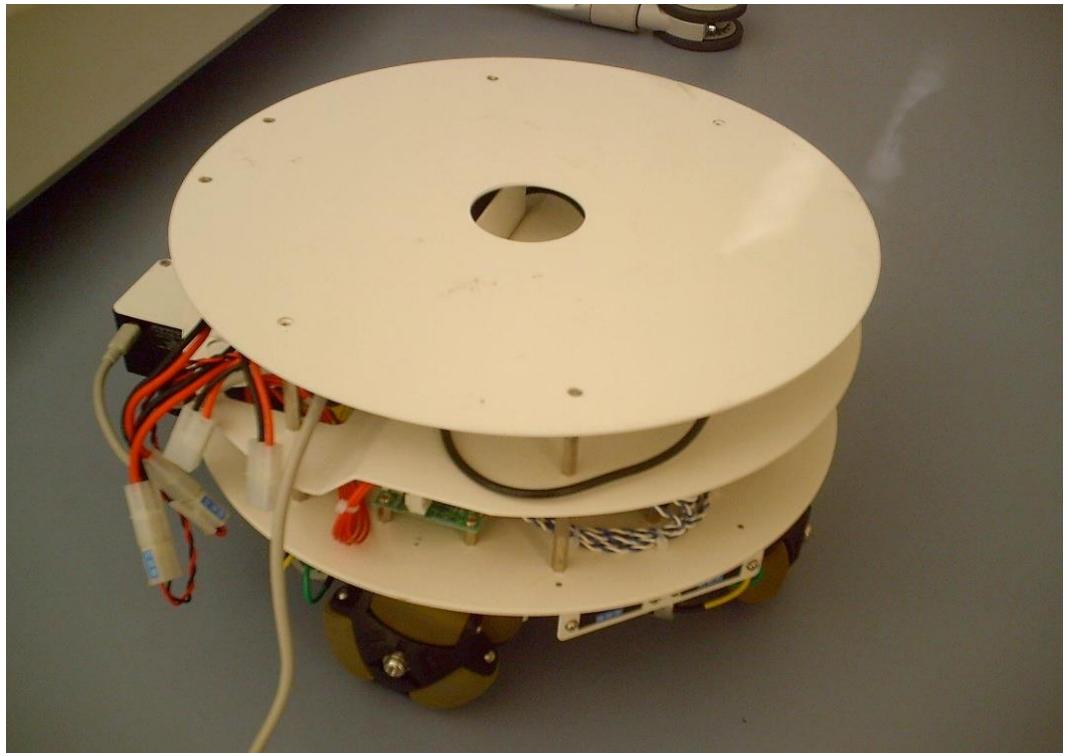
Robot path planning

Introduction

- The goal of path planning is to find an optimal, collision-free path between two points in a static environment composed of walls and obstacles
  - It is assumed that the environment is known and given in advance
  - The planning can be carried out offline
- The optimality of the path is usually measured by the distance covered by the robot
- This problem belongs to the group of hard optimization problems
- Bacterial memetic algorithm is proposed for solving the problem

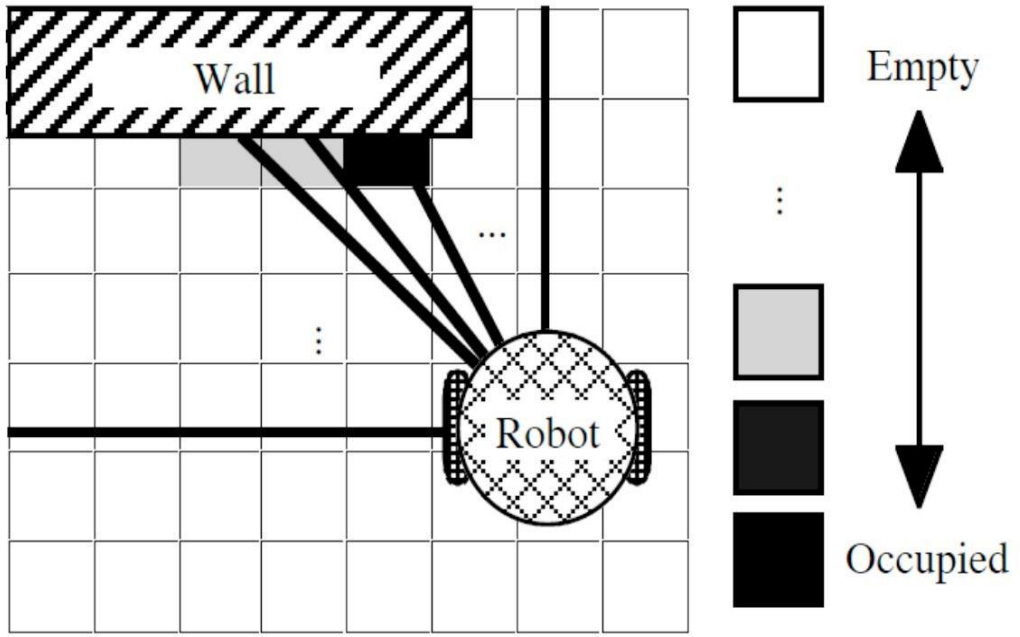
The applied robot

- An omni-directional mobile robot is used
- It has four omni-wheels and DC motors
- A laser range finder (URG04-LN) is used for the localization
  - It has a maximum of 682 rays and can measure the distance 4095 [mm]

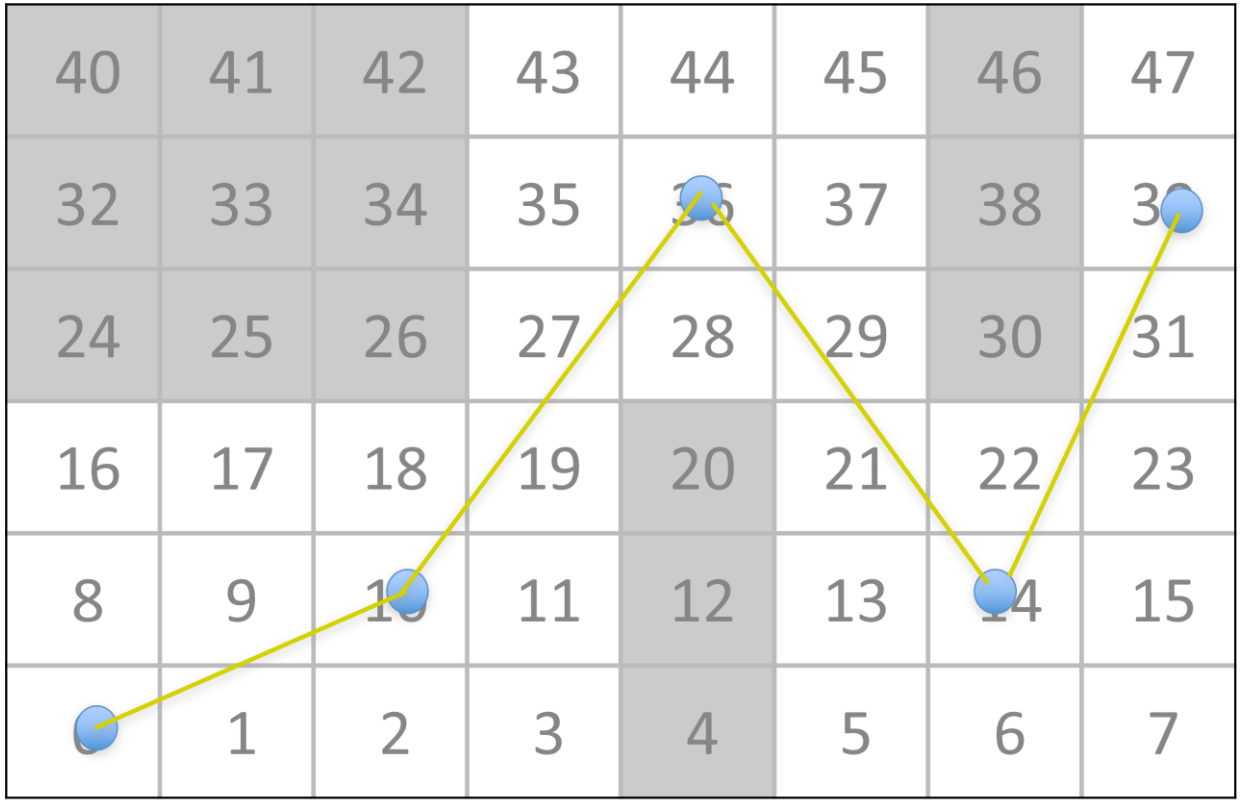


Environment and mapping

- Simultaneous localization and map building (SLAM)
- Two different types of maps are used for SLAM based on a cell decomposition method
  - An environmental map based on the occupation of obstacles
  - The other is used for recording the times of measurement by the laser range finder in order to improve the reliability of the map
- The value of the cell corresponding to the measured values in the absolute coordinate space is decreased, and the values of other cells are increased



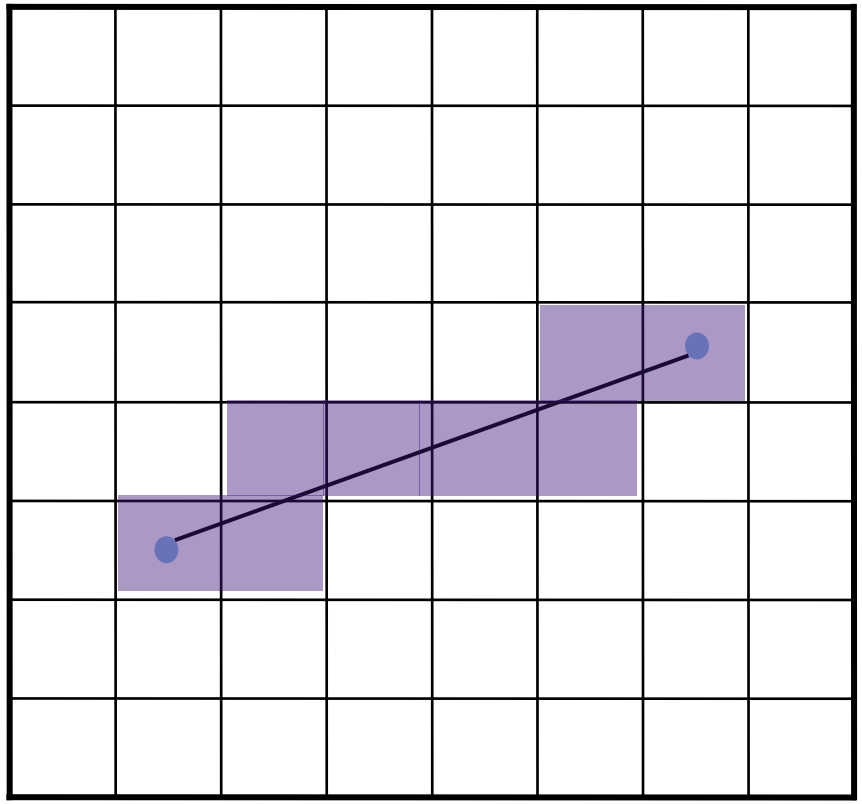
Encoding method



10	36	14

Evaluation of individuals

- In case of path planning a path is evaluated based on new intermediate points which cover the way from the start point to the target point by only neighboring points with 4 possible directions
- This can be done by using “straight” lines between the points encoded in the individual’s chromosome



Evaluation of individuals

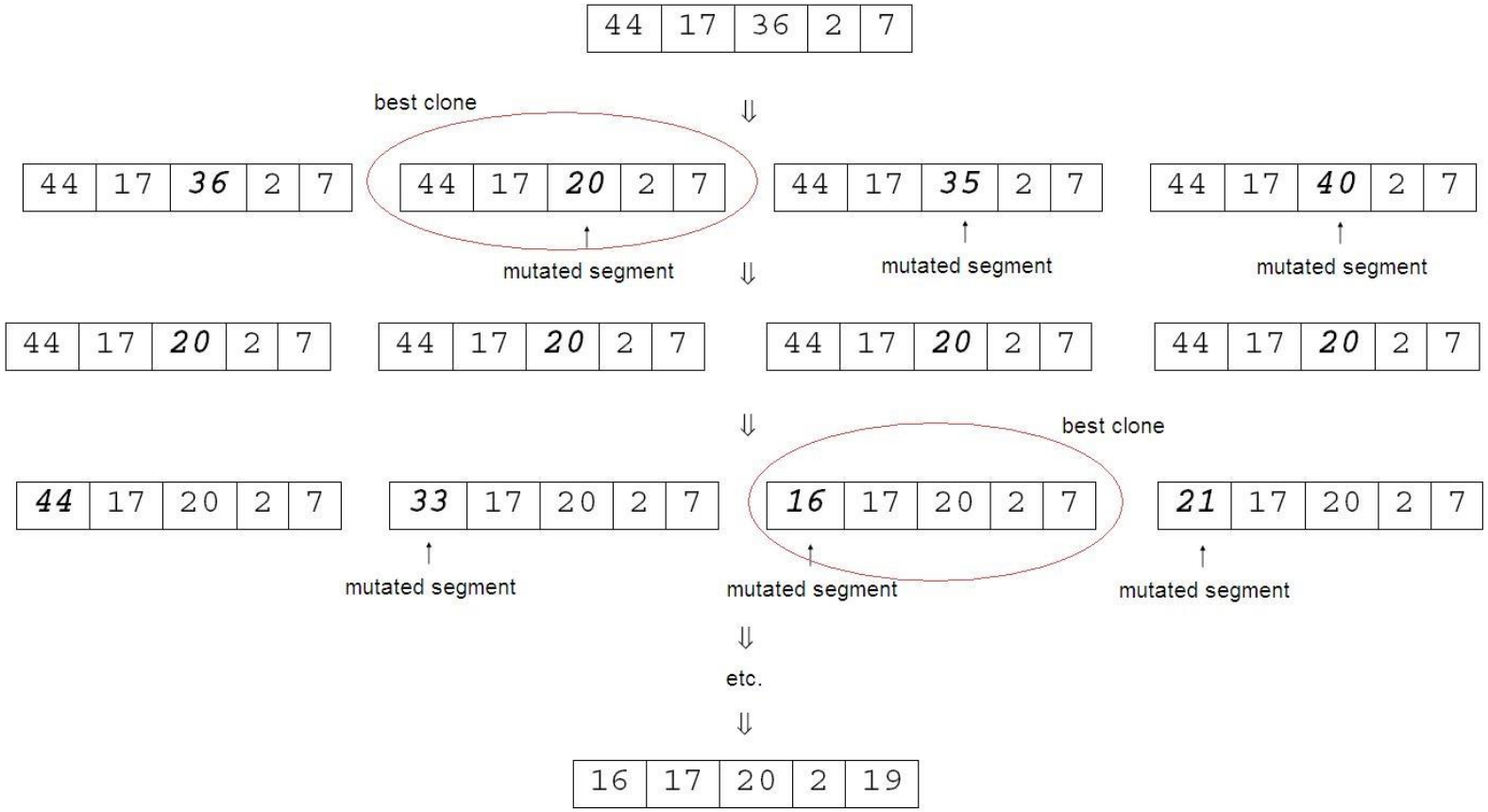
$$Eval_i = L_i + Penalty \cdot \sum_{j=1}^{L_i} CP_j + S \cdot Turns_i$$

- $Eval_i$: the i^{th} individual's evaluation value
- L_i : number of neighboring points in the extended individual
- CP_j : collision probability of the j-th cell in the extended individual's sequence
- Penalty: a parameter reflecting the strength of punishment for crossing occupied cells
- $Turns_i$: the number of robot's turns
- S: a parameter reflecting the smoothness of the path

Initial population

- The initial population is created randomly
- The length of the bacteria is also random between 0 and the maximum allowed bacterium length
- If the length is 0 then the solution is a “straight” line between the start and target points
- Parameters:
  - number of bacteria
  - maximum allowed bacterium length

Bacterial mutation



Bacterial mutation

- The length of the segment is also a parameter of the operator (l_{bm})
- The mutation may not only change the content, but also the length
- The length of the new elements is chosen randomly as:

$$l_{bm} \in \{l_{bm} - l^{\star}, \ldots, l_{bm} + l^{\star}\}$$

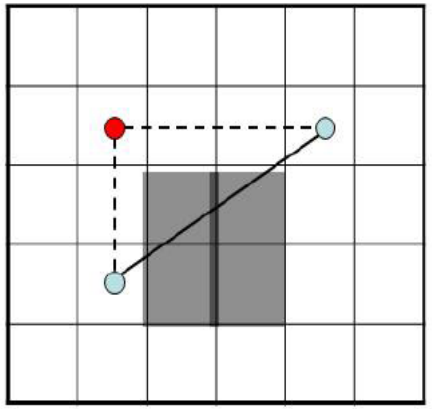
  - where $l^{*} \geq 0$ is a parameter specifying the maximal change in length
- When changing a segment of a bacterium, we must take care that the new segment is unique within the selected bacterium

Local search

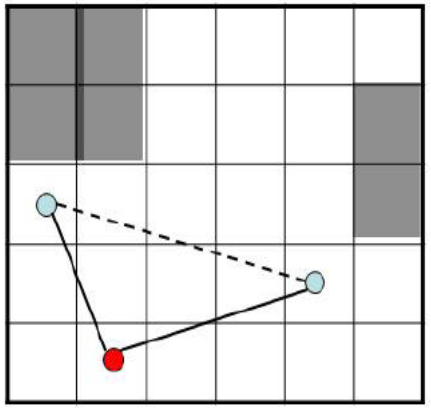
- Between bacterial mutation and gene transfer
- Local search probability: parameter of local search reflecting the chance for a bacterium to be local searched in a given generation
- Four kinds of local search operator are proposed
  - Insertion
  - Deletion
  - Swap
  - Local improvement
- The type of local search being performed on a given bacterium is randomly chosen with equal probability

Local search

- Insertion
  - It inserts a new point between those two intermediate points where the insertion has the biggest benefit
  - The operation is performed only when after insertion the bacterium will be better than before
- Deletion
  - It deletes that point from the individual whose removal has the biggest benefit
  - If no such point is found in the sense that after removal the bacterium will be worse then the operation will not be performed



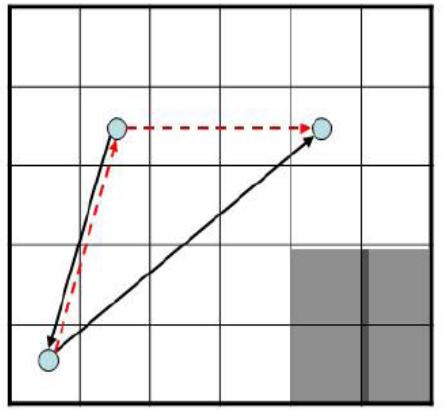
Insertion



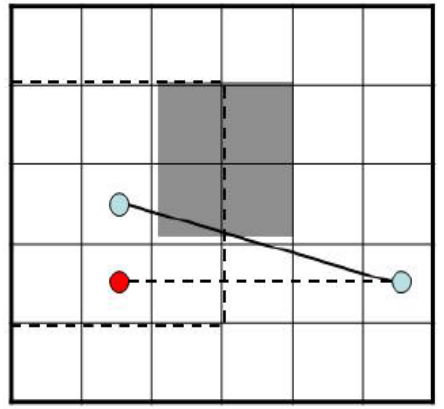
Deletion

Local search

- Swap
  - It tries to swap each pair of consecutive points in the individual's chromosome
  - The operation is performed when after swap the bacterium will be better
- Local improvement
  - It tries to improve each point in the chromosome in a given radius, which is a parameter of the algorithm
  - The local environment of the points is investigated and the best new position for the point is selected



Swap



Local improvement

Gene transfer

- The length of the segment is also a parameter of the operator (l_{gt})
- The gene transfer may change the length of the destination bacterium
- When transferring the segment from the source bacterium, we delete those elements from the destination bacterium which are also in the transferred segment
- Beside these elements more element can be deleted depending on the infection segment length change, which is a parameter of the operator

Parameters

- Number of generations
- Initial population creation:
  - number of bacteria
  - maximum allowed bacterium length
- Evaluation:
  - Penalty
  - S
- Bacterial mutation:
  - number of clones
  - mutation segment length
  - mutation segment length change
- Gene transfer:
  - number of infections
  - infection segment length
  - infection segment length change
- Local search:
  - local search probability
  - radius

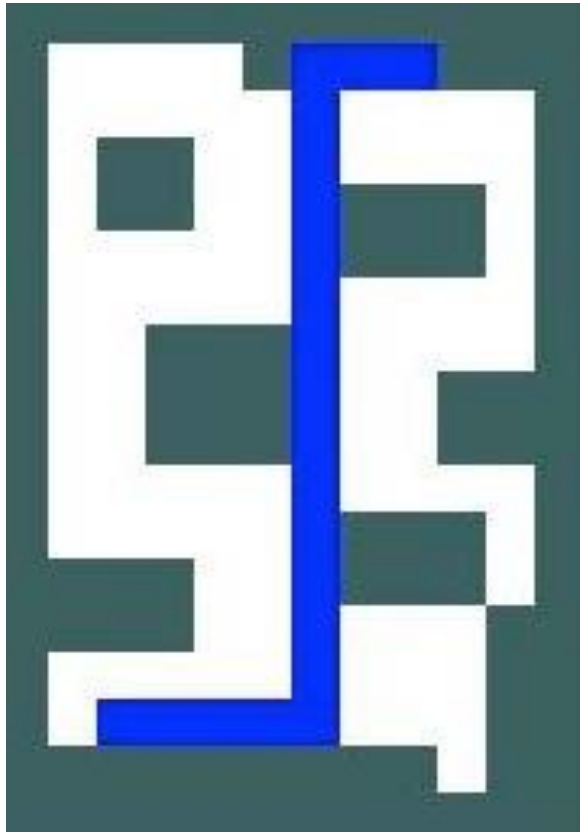
Experimental results

- Two environments were used

Parameter settings

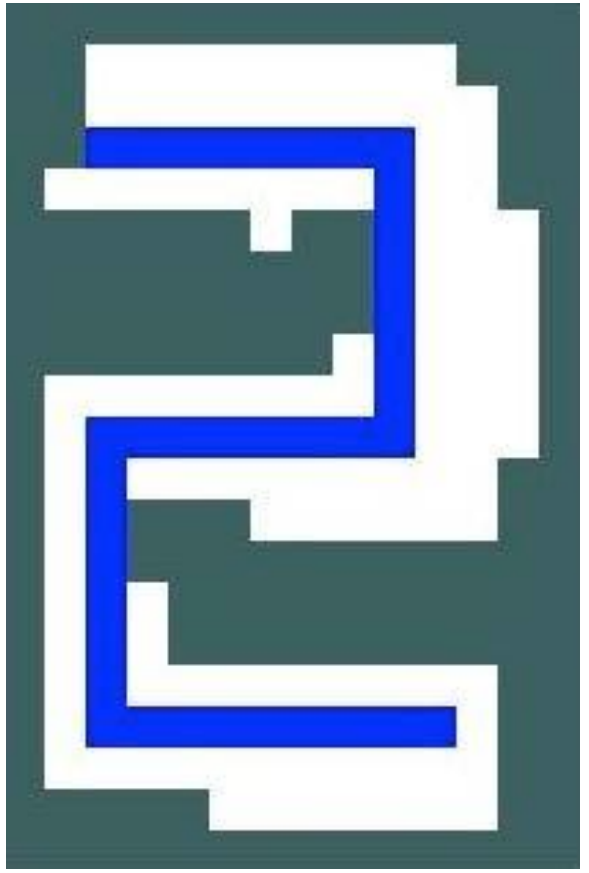
Operation	Parameter	Test 1	Test 2
	Number of generations	30	70
	Number of bacteria	15	70
	Max. bacterium length	10	10
	Penalty	5000	5000
	S	0.5	0.5
Bacterial mutation	Number of clones	5	25
	Segment length	2	2
	Segment length change	1	1
Gene transfer	Number of infections	5	25
	Segment length	2	2
	Segment length change	1	1
Local search	Probability	20%	20%
	Radius	2	3

Results



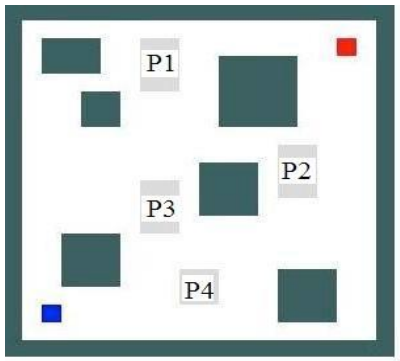
First test
The optimal solution was found ten times from ten simulations

Second test
The optimal solution was found eight times from ten simulations

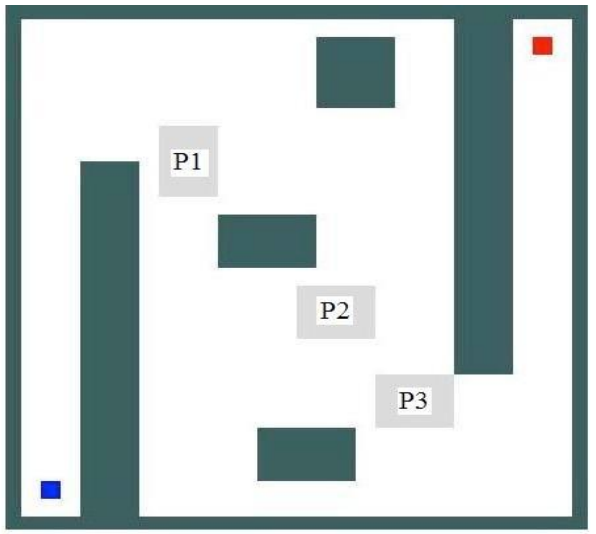


The optimal path of the robot

Simulation results - probabilistic case



Task 1



Task 2

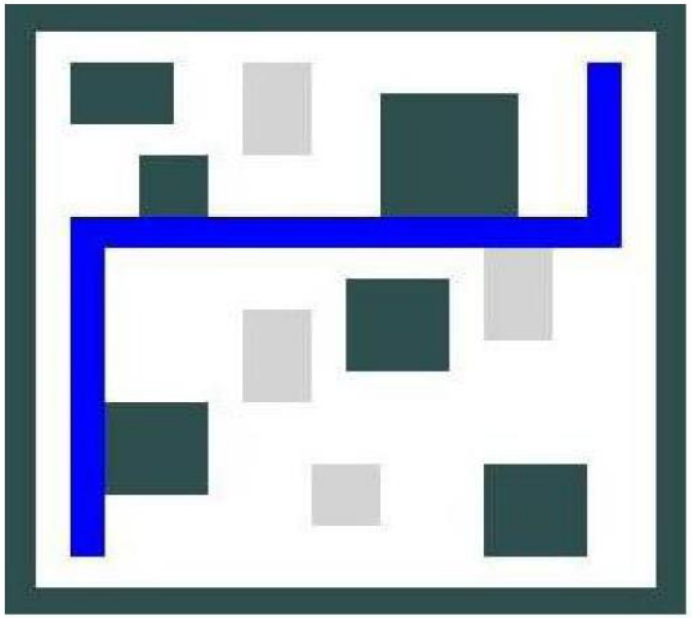
map size: 20 × 20 30 × 30

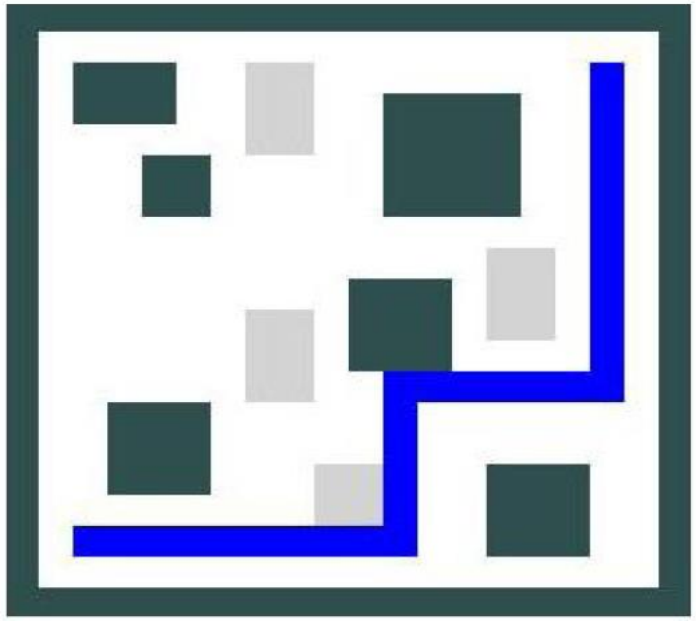
probabilities: P1=0.4, P2=0.8, P3=0.2, P4=0.6 P1=0.8, P2=0.3, P3=0.7

Parameter settings

Operation	Parameter	Task 1	Task 2
	Number of generations	30	200
	Number of bacteria	30	200
	Max. bacterium length	10	20
	Penalty	1000	1000
	S	0.1	0.1
Bacterial mutation	Number of clones	5	25
	Segment length	2	2
	Segment length change	1	1
Gene transfer	Number of infections	5	25
	Segment length	2	2
	Segment length change	1	1
Local search	Probability	20%	20%
	Radius	2	3

Simulation results

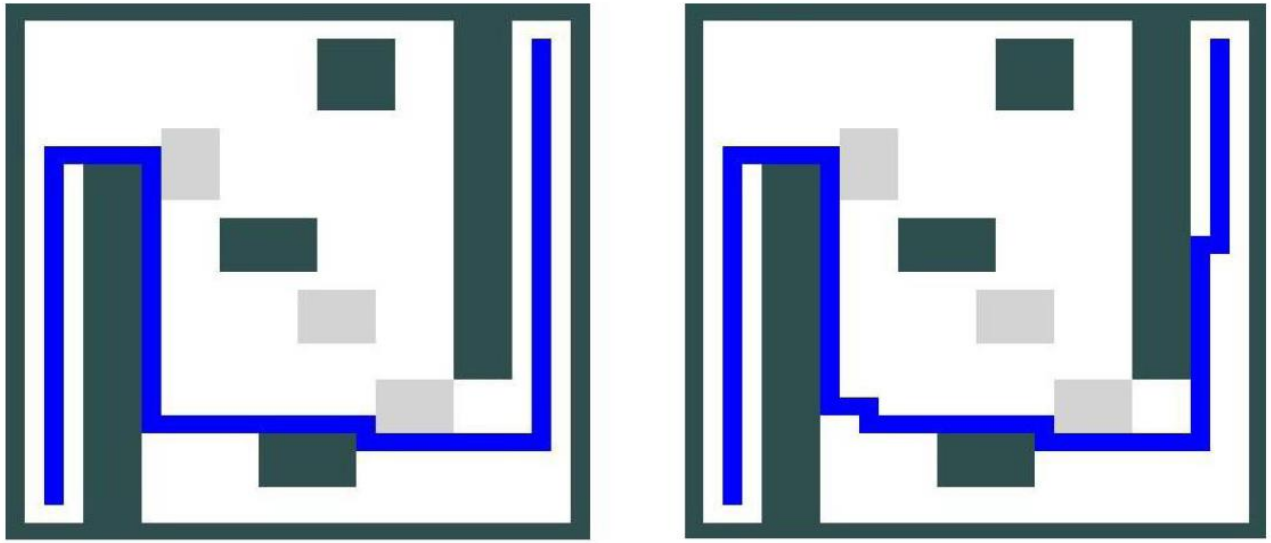




The optimal solution was obtained 7 times from 10 simulations

In the remaining 3 cases only the number of turns was not optimal

Simulation results



The optimal solution was obtained 3 times from 10 simulations

In the remaining cases the obtained paths were also collision-free, only the path length and the number of turns were not optimal

Conclusions

- Bacterial memetic algorithm was proposed for solving the path planning problem in probabilistic environment
- Classical approaches for path planning problems rely on the exact estimation of the cost which cannot be performed in probabilistic cases
- Evolutionary techniques can be useful in these cases, however, their convergence speed to the optimum is pretty slow
- BMA effectively combines the bacterial operators with local search heuristics and can speed up the evolutionary process in this way
- The algorithm can handle different individual length

Path planning, flow shop scheduling, task assignment

Introduction

- Path planning is an essential task in mobile robotics
  - The goal is to find an optimal, collision-free path between a start and a target point in an environment surrounded by walls and obstacles
  - The optimality of the path is usually measured by the distance covered by the robot
- In shop scheduling problems there are *n* jobs to be processed on *m* machines
  - There is a matrix specifying how much time needed for a job to be processed on a given machine
  - The goal is to minimize the overall time demand of the whole process
- Two robots are used for carrying items between the machines
  - The transportation tasks should be assigned to the robots optimally
- The above problems belong to the group of hard combinatorial optimization problems
- Bacterial memetic algorithm is proposed for solving them

Problem statement

- Flow shop problems
  - There is more than one machine and each job must be processed on each of the machines
  - The number of operations for each job is equal to the number of machines
  - The processing order of the operations of the job is predefined and it is the same for each job
- Permutation flow shop problems
  - The machines process the jobs in the same order, so the order of the jobs on each machine is the same
  - The task here is to find a permutation of the jobs which permutation will fit on every machine
- There are *n* jobs, *m* machines and two robots
- Inputs
  - The map with obstacles and machines
  - The matrix containing the processing time for each job on each machine

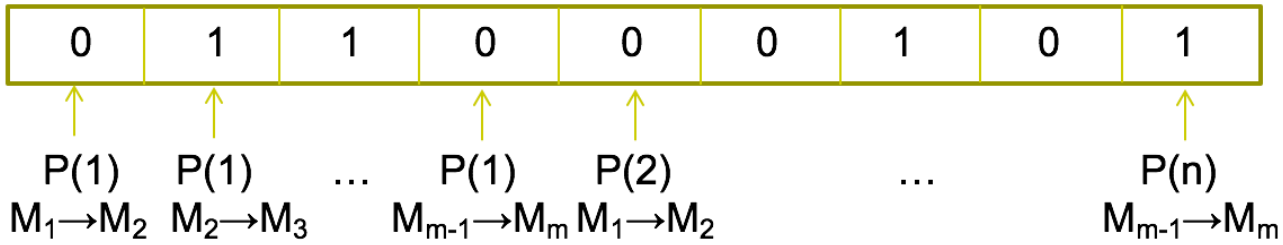
Problem statement

- There are three subproblems
  - Finding the permutation of jobs
  - Task assignment for the robots
  - Path planning for the robots with collision avoidance
- The machines are processing the jobs, while the robots are carrying the items between the machines. After a job is finished on a machine the job will be carried to the next machine by one of the two robots. The job can be processed on this next machine only after the robot is reached the machine with the job-related items and the previous job according to the schedule on the machine is already finished
- The goal is to minimize the completion time of the whole process (makespan)
- Calculation of the makespan
  - Difficult, thus a visualization software is developed to illustrate the scenario
  - We need state matrices describing the states of the machines, the current job times on the machines and the states of the robots

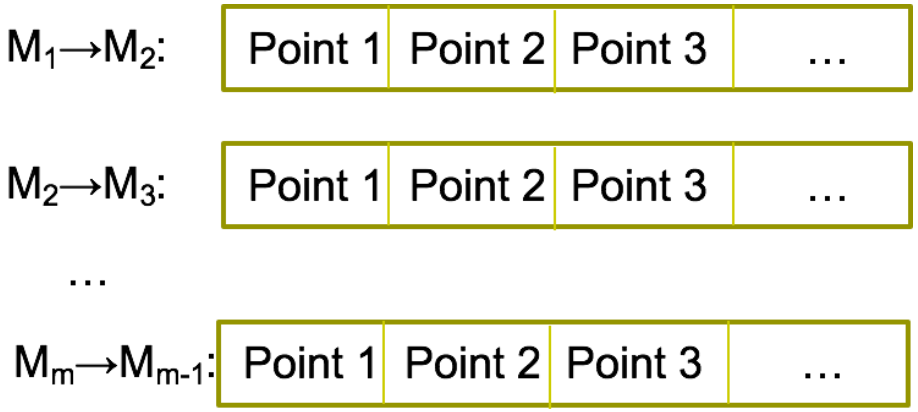
Encoding method

- Job scheduling:

- Task assignment:



- Path planning:

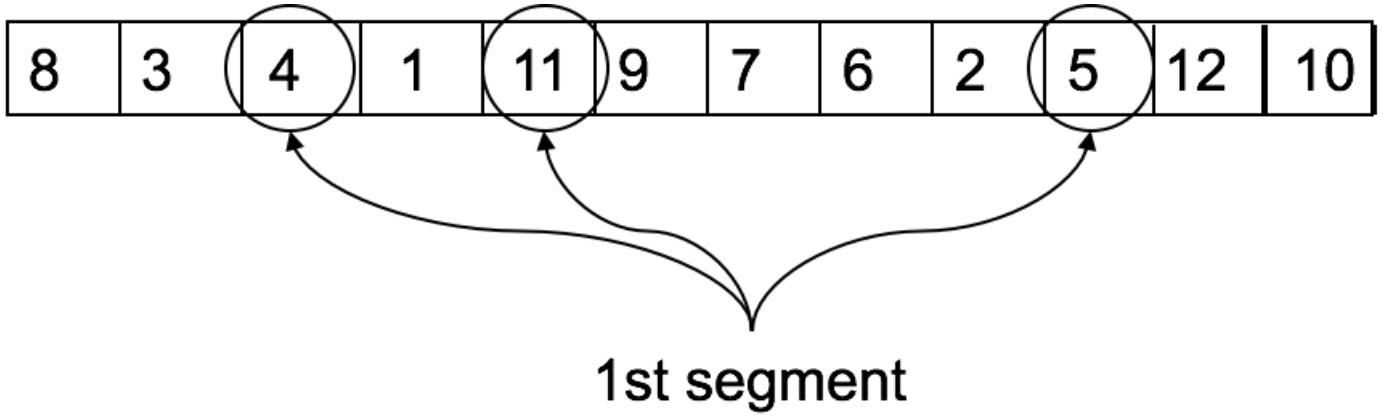


Evaluation of individuals

- In the case of path planning as seen before
- In the case of the whole problem
  - The whole scenario is executed
  - We use state matrices describing the states of the machines, the current job times on the machines and the states of the robots

Bacterial mutation

- The length of the segment is also a parameter of the operator
- In the case of flow shop scheduling the segments of the chromosome do not need to consist of consecutive elements



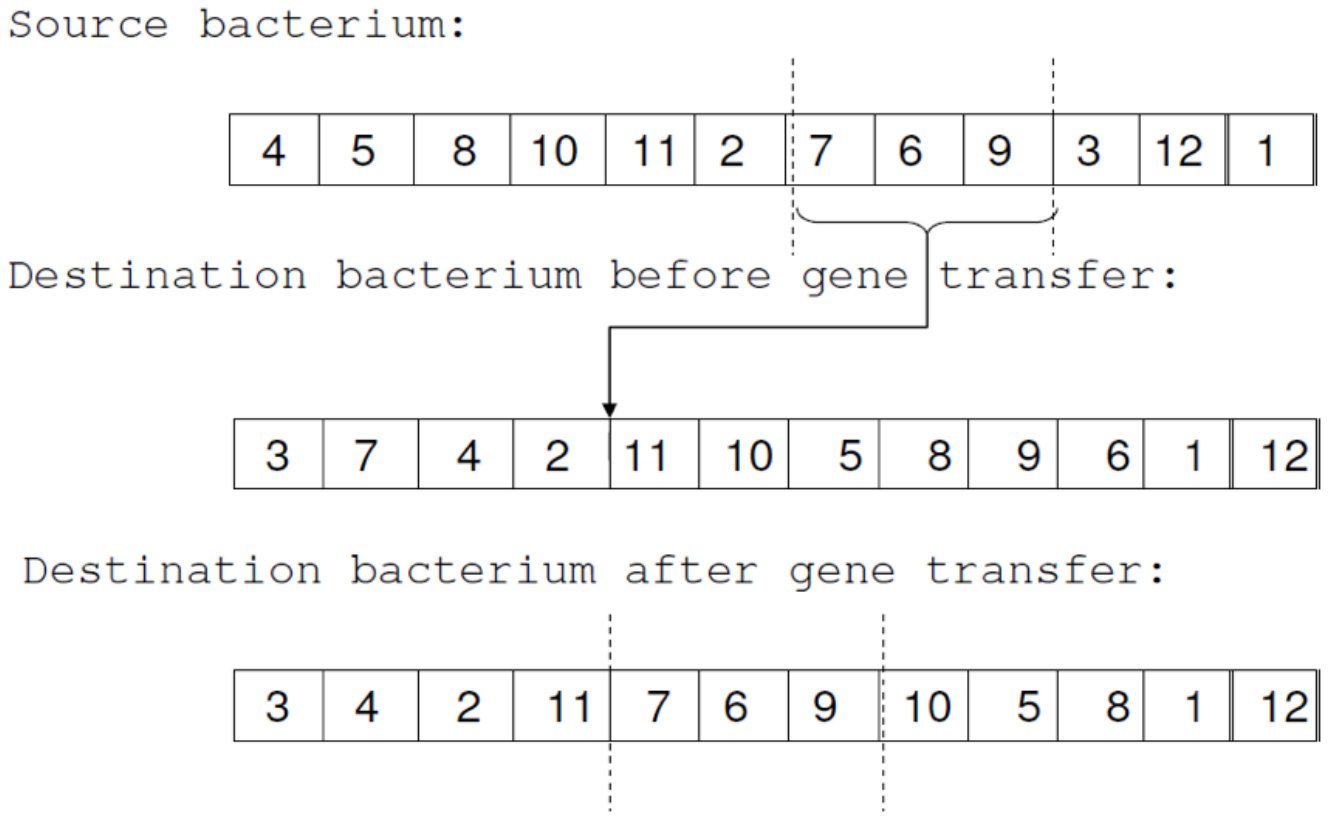
- In the case of path planning chromosome the mutation may not only change the content, but also the length

Local search

- Local search probability: parameter of local search reflecting the chance for a bacterium to be local searched in a given generation
- In the case of the job permutation and task assignment type chromosomes the neighborhood values are exchanged in the chromosome if it leads to better solution
- In the case of path planning four kinds of local search operator are proposed as seen before
- The type of local search being performed on a given bacterium is randomly chosen with equal probability

Gene transfer

- The length of the segment is also a parameter of the operator
- In the case of path planning chromosome the gene transfer may change the length of the destination bacterium
- In the case of flow shop chromosome, the segment can contain only consecutive elements within the bacterium



Simulation results

- 10 jobs, 3 machines, map size: 20×20
- Processing times:

	M1	M2	M3
J1	17	23	18
J2	16	18	25
J3	21	19	23
J4	19	27	26
J5	28	17	21
J6	18	26	19
J7	26	20	22
J8	24	24	24
J9	20	21	22
J10	19	22	20

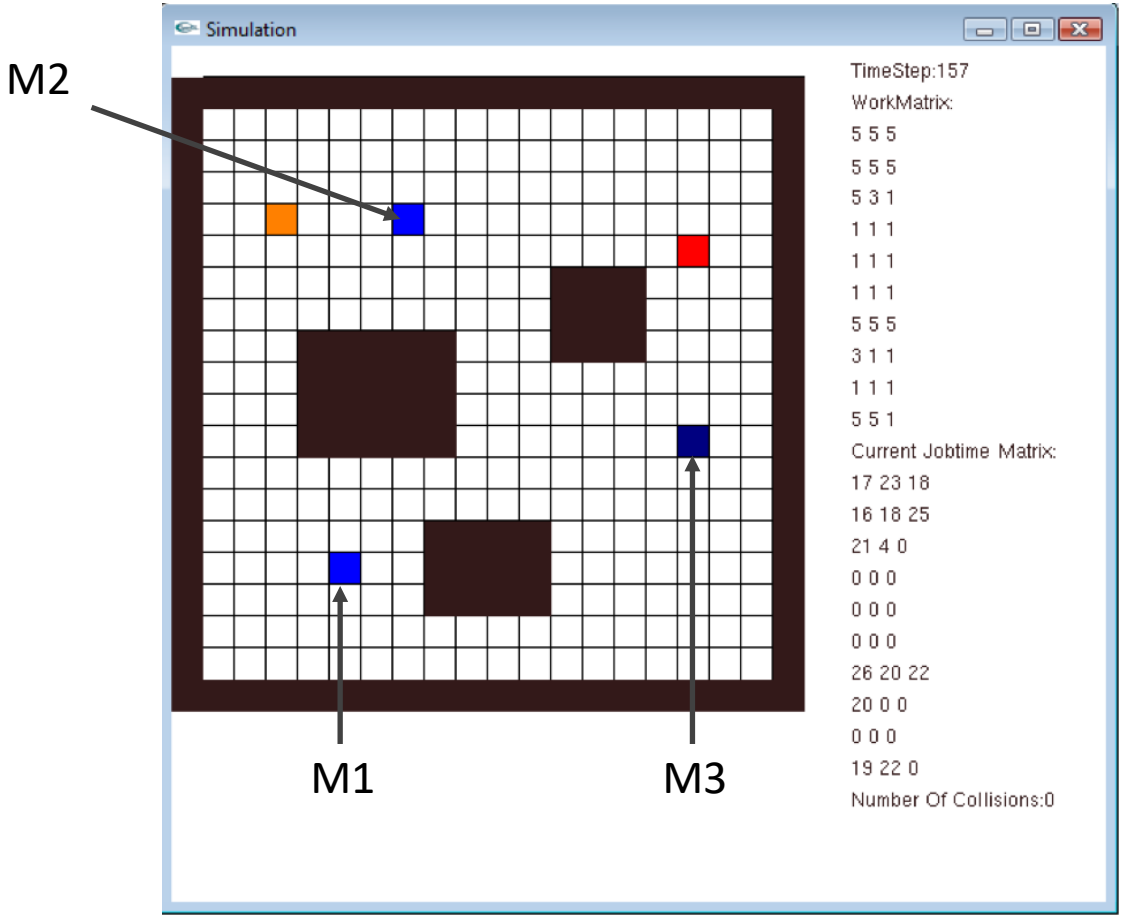
Simulation results

black: obstacle

blue: machine

orange: robot

red: robot



Simulation results

- N_{gen}=25, N_{ind}=25
- Path planning: N_{clones}=5, N_{inf}=5, mut.length=2, inf.length=2
- Flow shop and task assignment: N_{clones}=3, N_{inf}=5, mut.length=3, inf.length=3
- Local search propability=20%
- Collision penalty between robots=10
- The average of best solutions for the makespan based on 10 simulations: 379.7
- Best solution based on 10 simulations: 376
- Worst solution: 383
- In the best solution the job permutation is: (2,1,6,8,4,3,10,9,7,5), the task assignment is: 00101010101100101011, the path lengths are: 19,18,19,17,18

Conclusions

- Bacterial memetic algorithm was proposed for simultaneous optimization of path planning, flow shop scheduling, and task assignment in case of two robots
- The algorithm effectively combines the bacterial operators with local search heuristics and can speed up the evolutionary process in this way
- The operators can handle different type of encodings at the same time, like permutation type chromosome, binary, and integer encoding

A Rapidly-Exploring Random Tree Algorithm by Reducing Random Map Size

Introduction

- Mobile robots have been widely used in automated factory applications such as raw material delivery and product storage transportation
- Path planning algorithms have been proposed to generate a feasible global approach. The path result must be free from obstacle regions and shortest.
- We propose an Improved Rapidly-Exploring Random Tree (Improved RRT*) algorithm.
- The algorithm consists of the pre-processing step for feasible mapping, primary processing with RRT* for path generating, and post-processing with Bacterial Mutation and Node Deletion operators.

Introduction

- We improve further the capability of the Improved RRT* algorithm by reducing the overall computation time.
- The proposed method reduces the size of the random map after each iteration by deleting the used nodes

Rapidly Exploring Random Tree*

- RRT* is a random sampling tree structure search algorithm.

Algorithm 1 RRT* Algorithm

Initialize q_{start} and q_{goal}
for $i < MaxIteration$ **do**
  $q_{rand} \leftarrow$ random node $(0 - X_{max}, 0 - Y_{max})$
  $q_{near} \leftarrow$ find nearest node from Tree
  if obstacle free between q_{near} and q_{new} **then**
    $q_{new} \leftarrow$ steer from q_{near}
    Find minimum cost from q_{min} and q_{new} in radius of R
    Add q_{new} to Tree
  if distance between q_{new} and $q_{goal} \leq D$ **then**
    Stop iteration
Return Tree

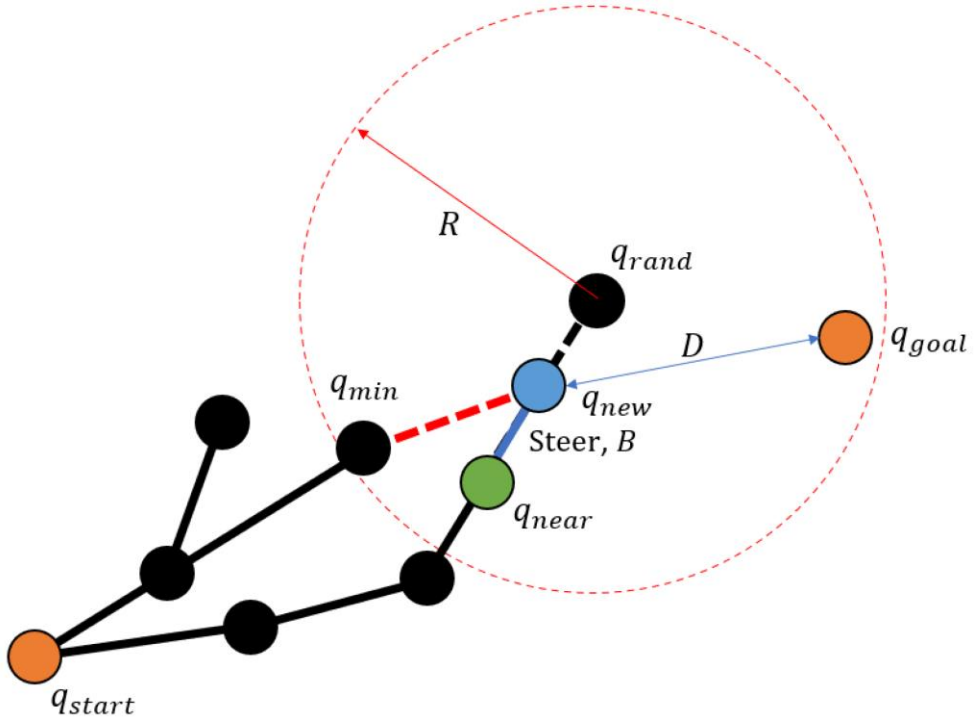
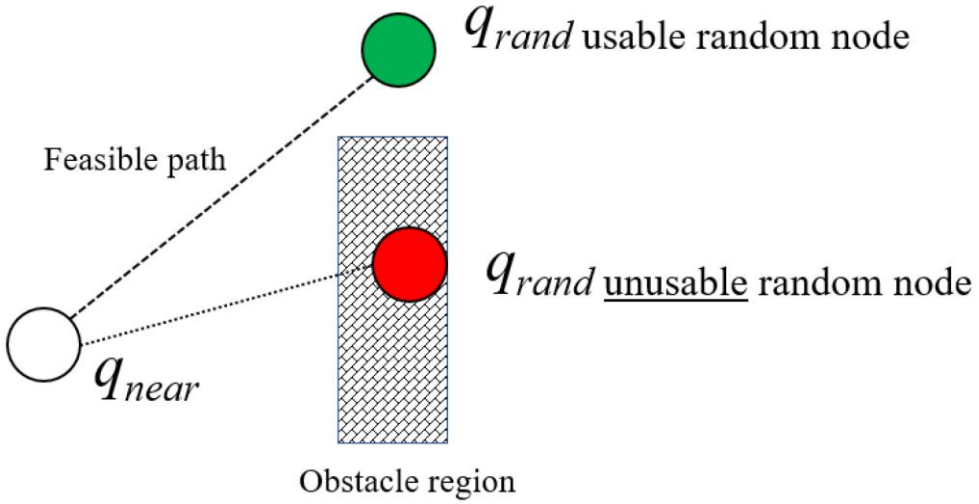


Illustration of RRT* Algorithm

Unusable Nodes

- The unusable nodes are the nodes that fall into the obstacle regions.
- The number of generated unusable nodes is over 40% of RRT*.
- The main idea to improve the efficiency of the RRT* algorithm is to reduce the unusable nodes from the iteration and let the iteration flow continue with usable nodes.



Unusable Node

Improved RRT*

- RRT* is a random sampling tree structure search algorithm.
- The number of generated unusable nodes is over 40% of RRT*
- We proposed to deal with the situation when the random nodes fell into the obstacle regions

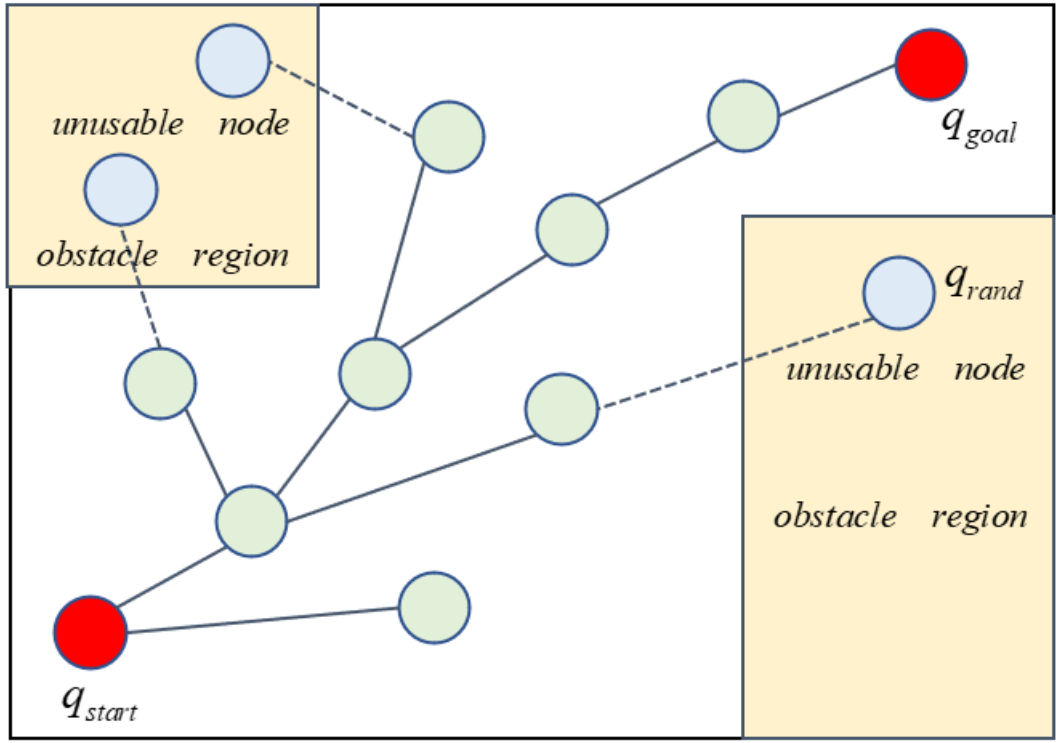
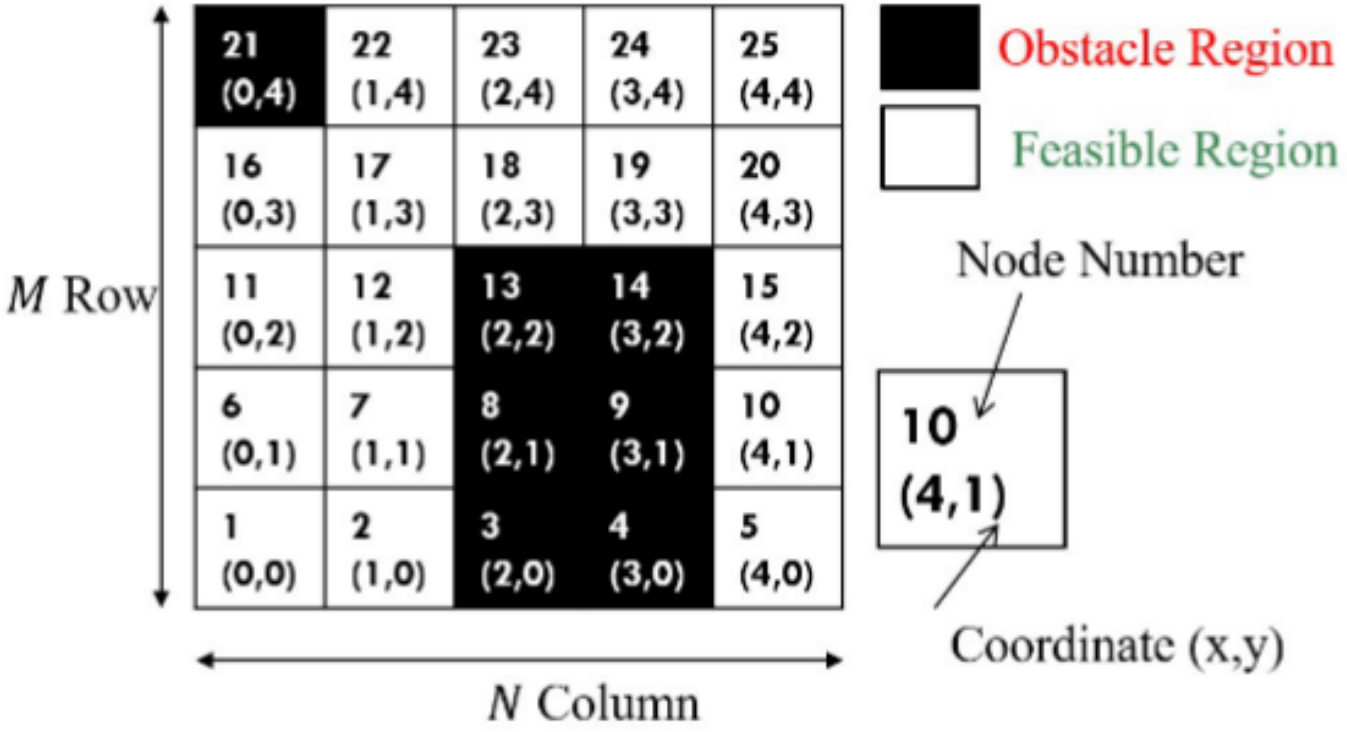


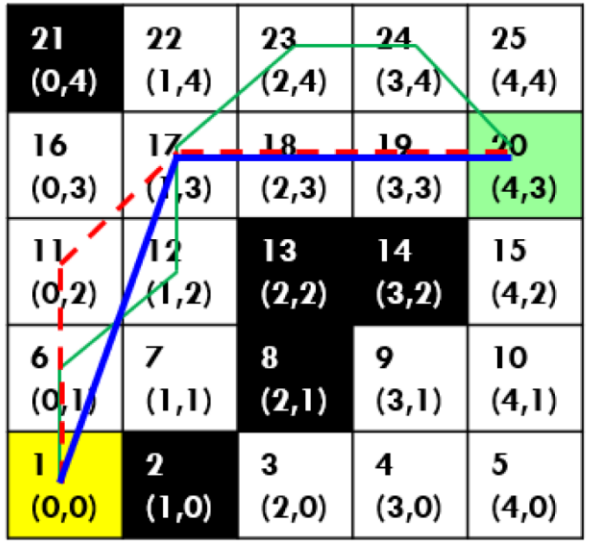
Illustration of RRT-based Algorithm

Environment and Mapping



The global environment

Post-processing Algorithms



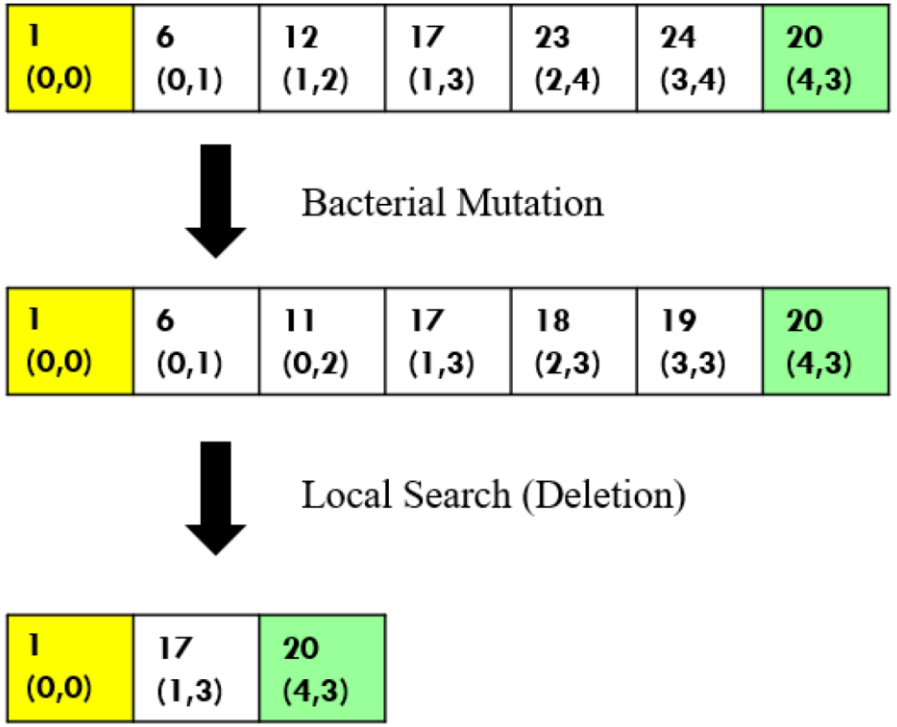
Starting Node

Goal Node

The solution from Improved RRT*
Define as Bacterium
(Green Path)

After bacterial mutation process
(Red Path)

Final Solution
(Blue Path)



Improved RRT*

- The global environment is stretched from the $M \times N$ matrix to a row vector
- The yellow-covered nodes represent obstacle region nodes.
- The resulting vector ($randMap$) contains only the feasible region after the obstacle regions are removed.
- The advantage of the proposed mapping algorithm is that it can also deal with irregular-shaped obstacle regions

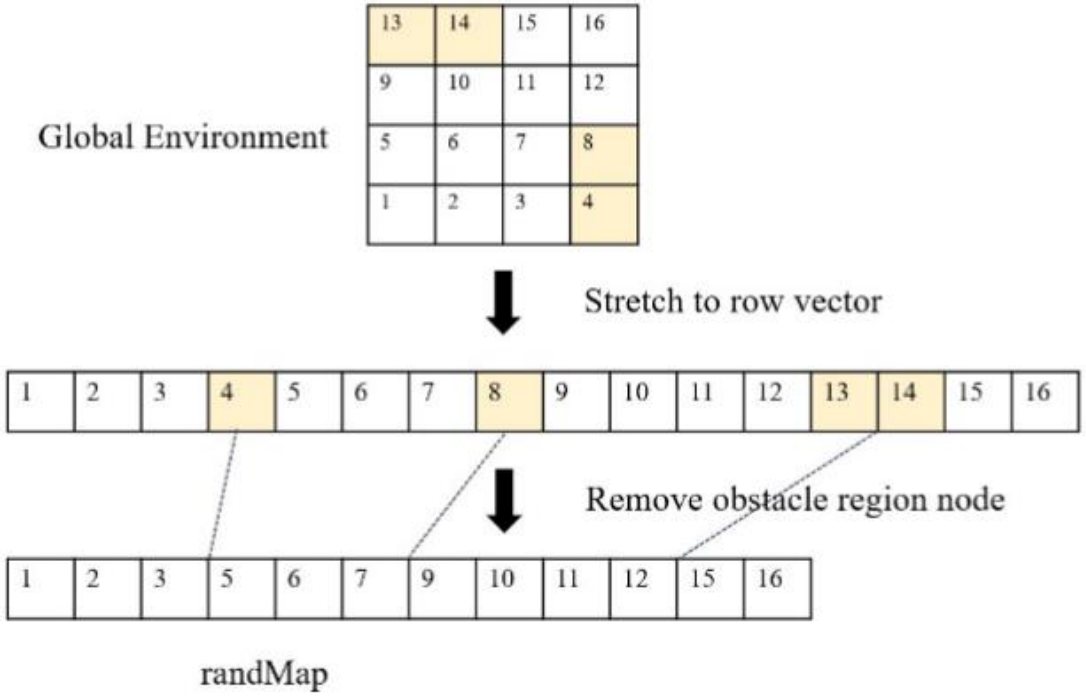
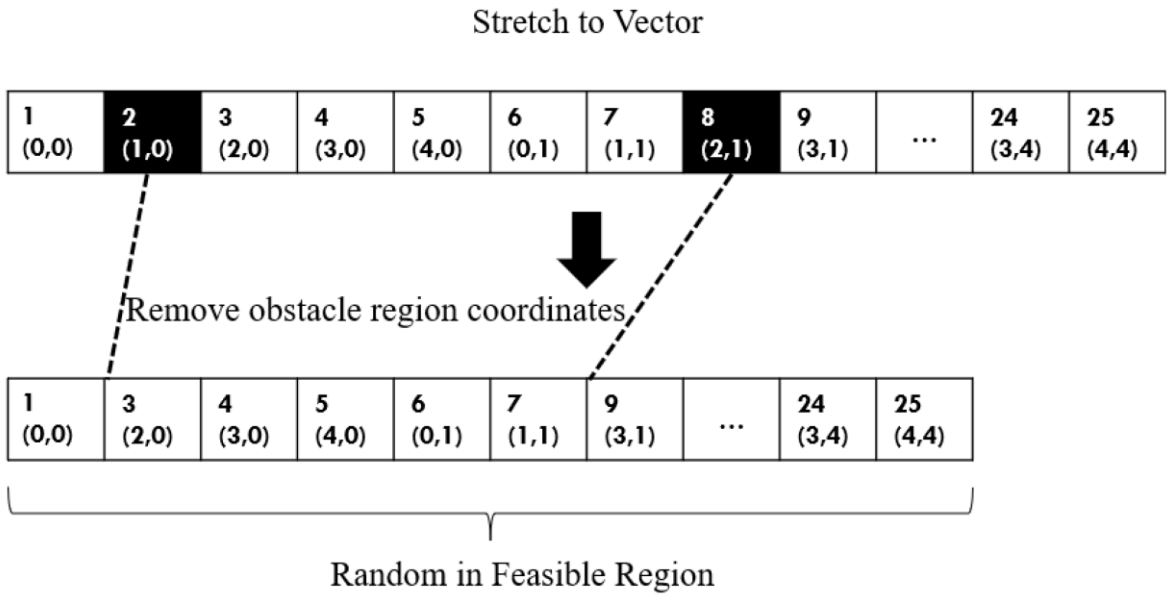


Fig. 3: Improved RRT*

Improved Algorithm

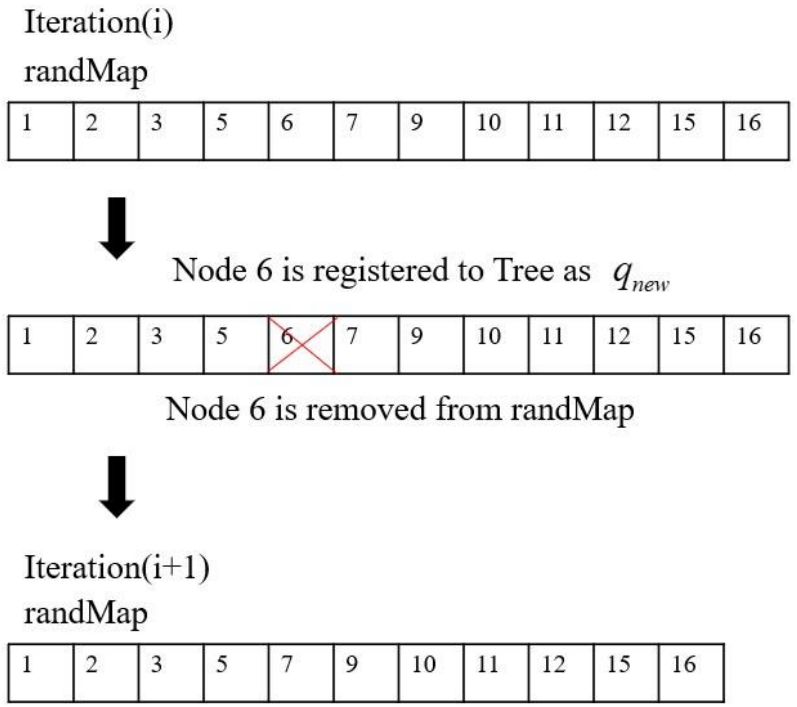
Algorithm 2 Improved RRT* with Bacterial Mutation and Node Deletion Algorithms

```
Map = ReadMap from file (.bmp)
randMap = StretchMap from maxtrix to row vector
for i < Length(randMap) do
    if randMap(i) is an obstacle region then
        Delete randMap(i) from randMap vector
Initialize q_start and q_goal
for i < MaxIteration do
    q_rand ← random node randMap
    q_near ← find nearest node from Tree
    if obstacle free between q_near and q_new then
        q_new ← steer from q_near
        Find minimum cost from q_min and q_new in radius of R
        Add q_new to Tree
    if distance between q_new and q_goal ≤ D then
        Stop iteration
Return Tree
Define Path as a Bacterium
for i < size of bacterium do
    Bacterial Mutation
Return Fine-tuned Bacterium
for i < size of bacterium do
    Node Deletion
Return Final Path
End
```



Further Improvement: Reducing Random Map Size

- An Improvement for Improved Algorithm for Path Planning Task of Mobile Robot
- To avoid the density of random nodes and improve the exploration of the algorithm



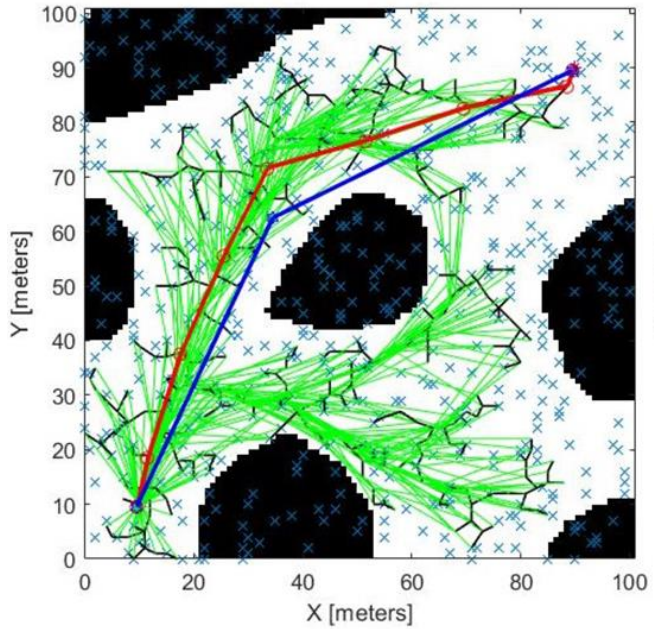
Algorithm 3 Improvement Algorithm

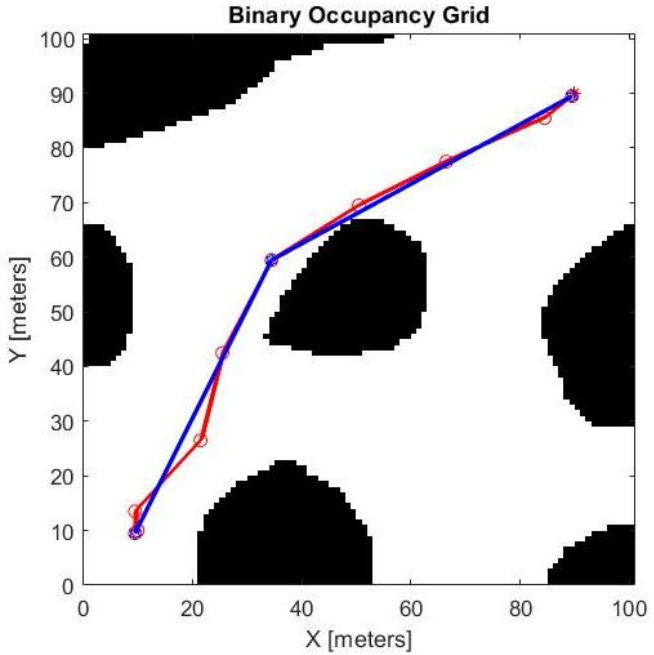
```
Map = ReadMap from file (.bmp)
randMap = StretchMap from maxtrix to row vector
for i < Length(randMap) do
    if randMap(i) is an obstacle region then
        Delete randMap(i) from randMap vector
Initialize q_start and q_goal
for i < MaxIteration do
    q_rand ← random node randMap
    q_near ← find nearest node from Tree
    if obstacle free between q_near and q_new then
        q_new ← steer from q_near
        Find minimum cost from q_min and q_new in radius of R
        Add q_new to Tree
        Remove q_new from randMap
    if distance between q_new and q_goal ≤ D then
        Stop iteration
Return Tree
Define Path as a Bacterium
for i < size of bacterium do
    Bacterial Mutation
Return Fine-tuned Bacterium
for i < size of bacterium do
    Node Deletion
Return Final Path
End
```

Experimental Results

- We combined the improved algorithm with the Bacterial Mutation and Node Deletion algorithms

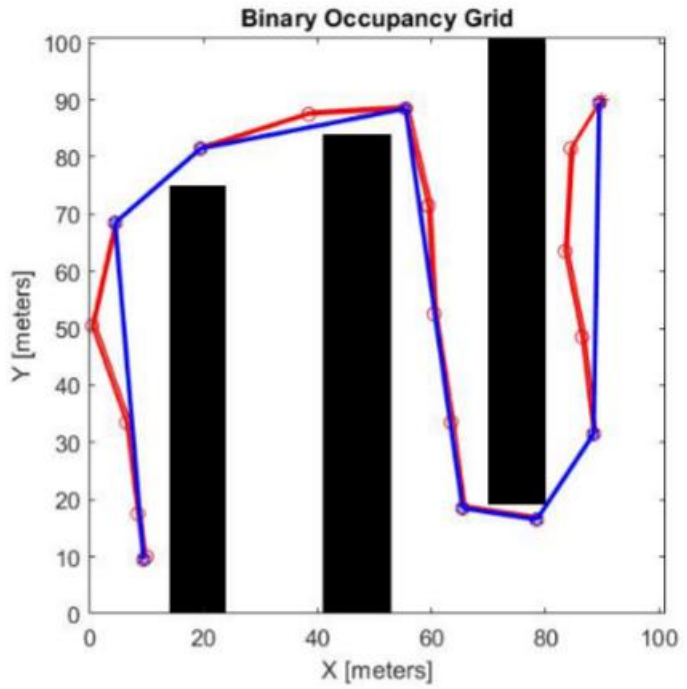
Improved RRT* (red);





Proposed RRT* (red)
Proposed RRT* with Bacterial Mutation and Node Deletion Algorithms (blue);

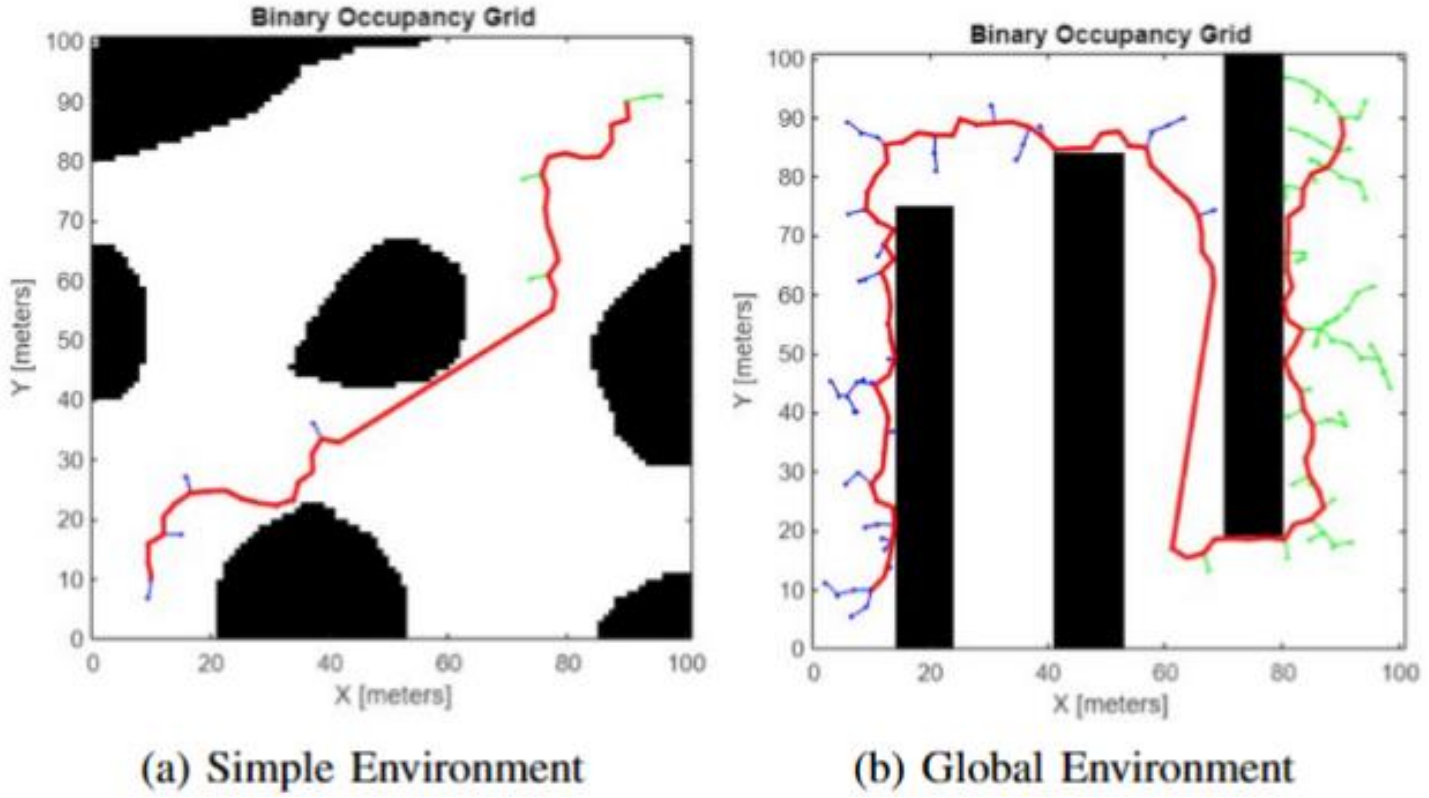
Experimental Results of Complex Environment



(b) Proposed RRT* with Reduced Random Map Size

Proposed RRT* (red)
Proposed RRT* with Bacterial Mutation and Node Deletion Algorithms (blue);

Compared with On-shelf Bi-RRT Algorithm



(a) Simple Environment (b) Global Environment

BiRRT Results

Computational Results

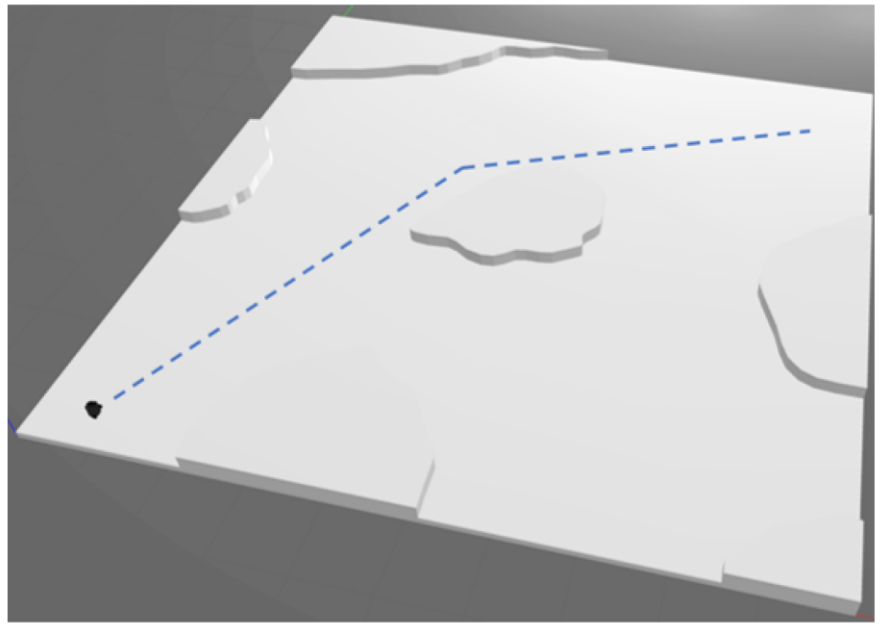
Previous results

New results

Results	Traditional RRT*	Improved RRT*	Proposed RRT*	BiRRT
Simple Environment				
Iterations	609	353	152	41
Path Length	120	120	120	141
Computational Time (s)	2.520	1.808	1.455	0.578
Complex Environment				
Iterations	2791	1835	1423	614
Path Length	274	270	264	315
Computational Time (s)	16.152	13.539	10.395	5.960

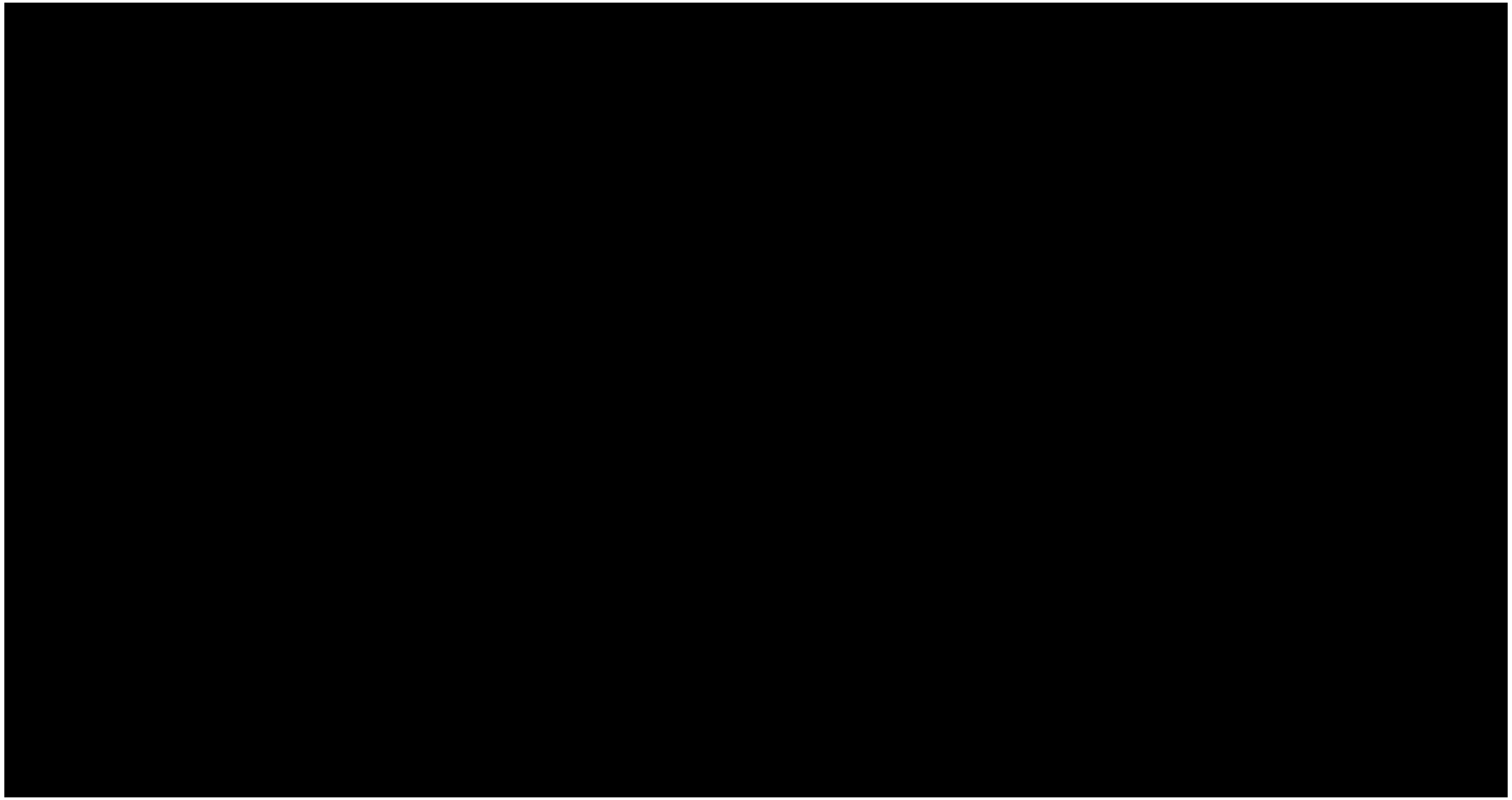
Simulation on the Robot

- The path planning result from post-processing was sent to TurtleBot3-Burger in Gazebo world by MATLAB programming via Robot Operating System.



Example of simulation path in Gazebo using TurtleBot3-Burger

Video of the Simulation



Unknown static obstacle avoidance

- Unknown static obstacles are a challenging task for mobile robot path planning
- When using a global path planning algorithm the resulting path can collide with unknown static obstacles
- An Improved RRT* with Reduced Random Map Size was previously proposed for global path planning
- Implement unknown static obstacles avoidance technique

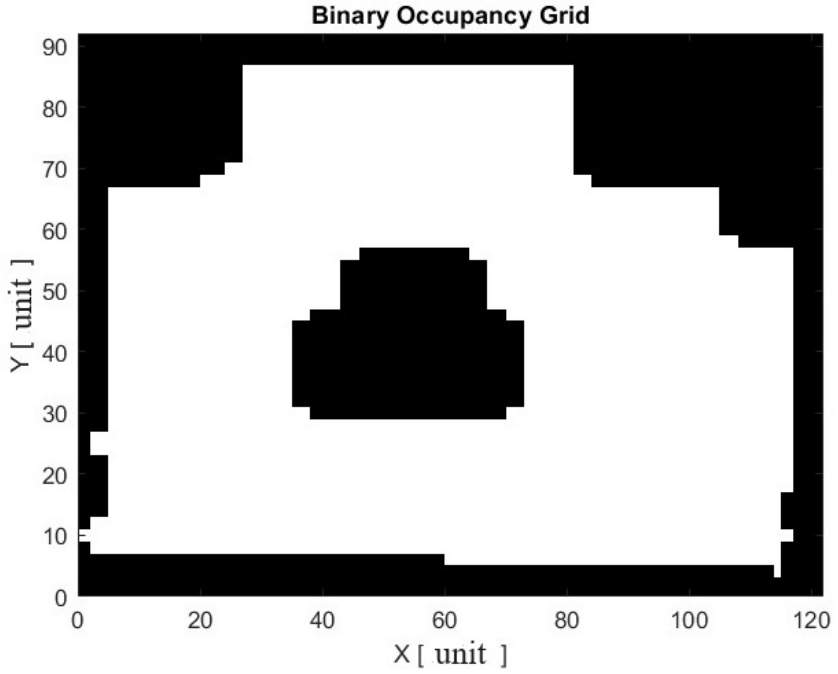
Waffle Pi robot
with OpenManipulator-X

Environment Setup

Real-world Environment

Real-world environment

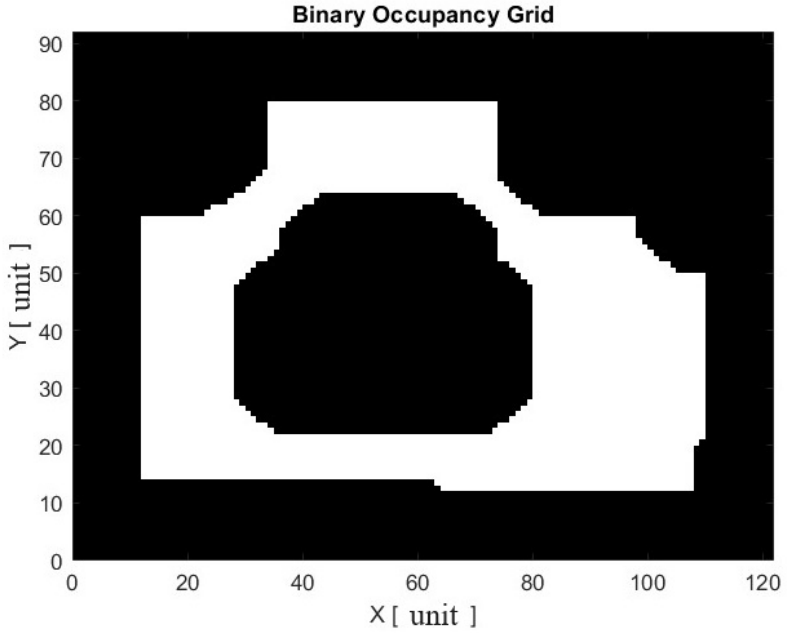
LiDAR Scanned Map (ROS)



Environment Setup

Pre-processing Map

Extended Version



Global environment and obstacle area extended version

Object and Unknown Static Obstacle

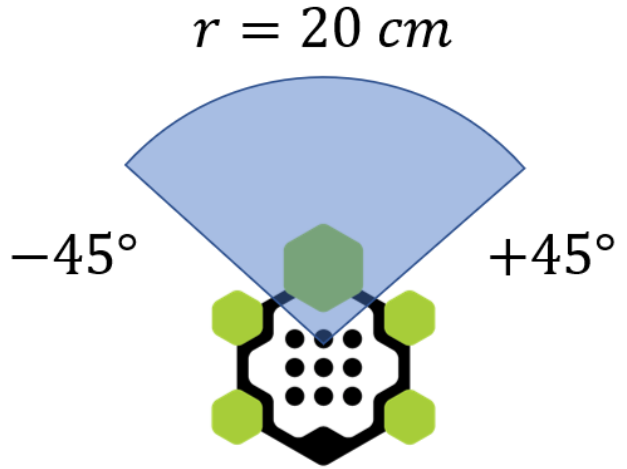
Object for Pick and Place

- 3 x 3 x 3 cm^3

Unknown Static Obstacle

- Cylinder with Radius 3 cm

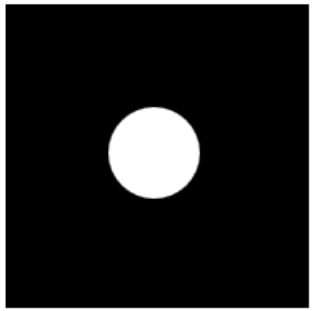
Real world object in environment



Determining an unknown static obstacle

Centering Robot to Station

- When the robot reaches the station, the algorithm detects the white circle and tries to adjust the detected circle's radius size and position



Sign for centering the robot to the station

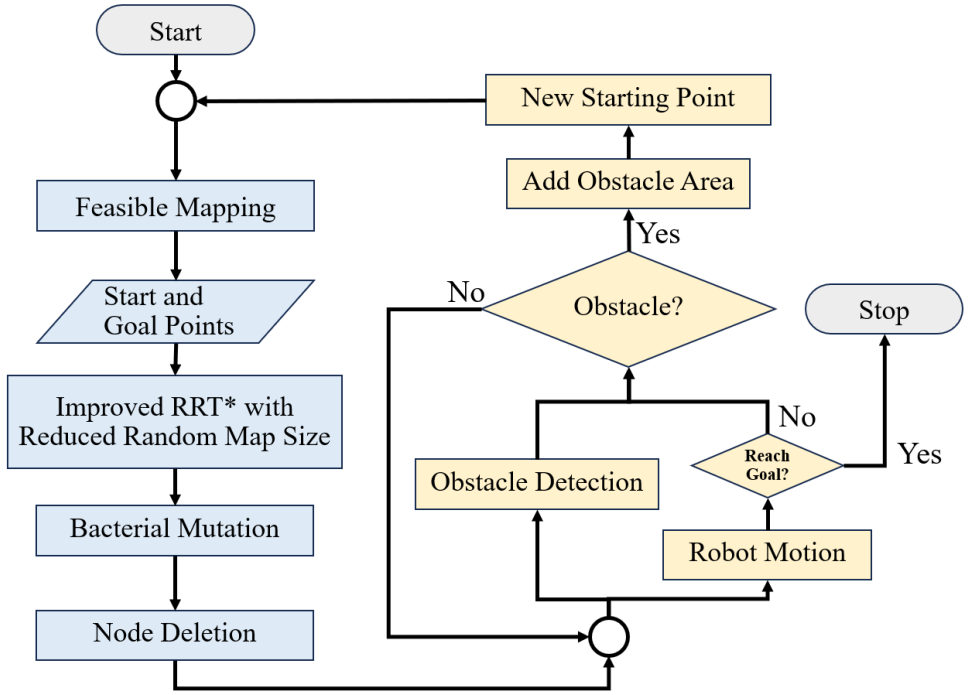
Centering the robot to the station

Experiment : Unknown Static Obstacle

Real-world Capture

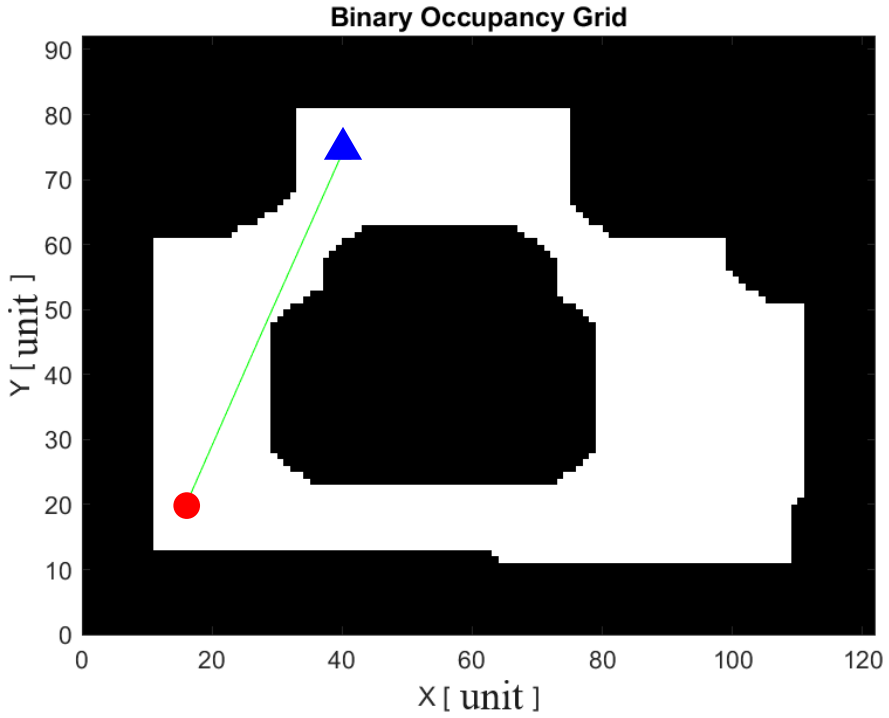
Captured robot motion
when facing with an unknown static obstacle

Flowchart

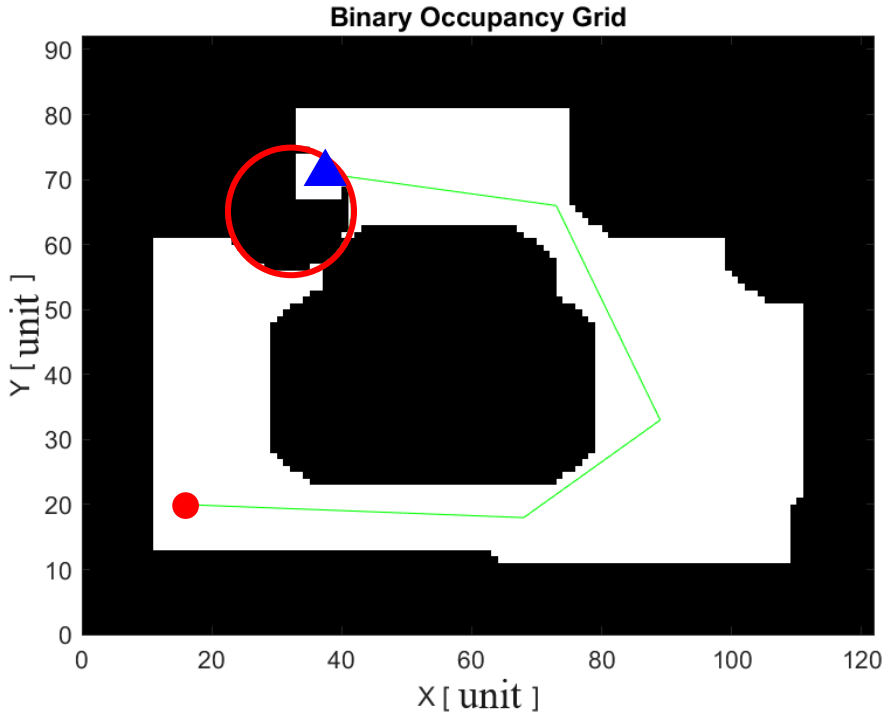


Experiment Result: Unknown Static Obstacle IRRT*-RRMS

Global path planning from Station 2 to Station 1



Local path planning with unknown static obstacle



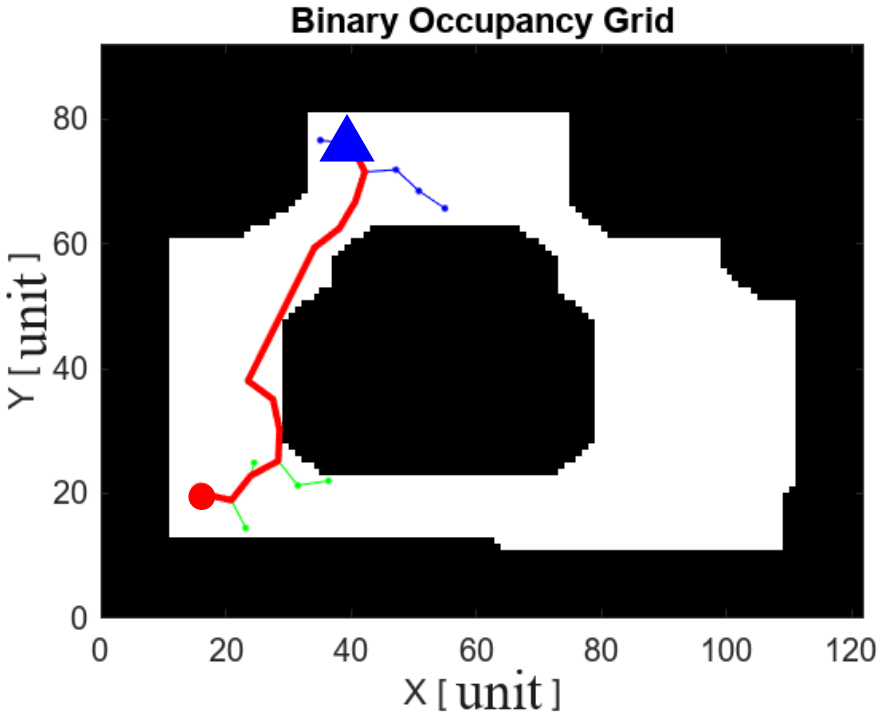
Starting Point

Goal Point

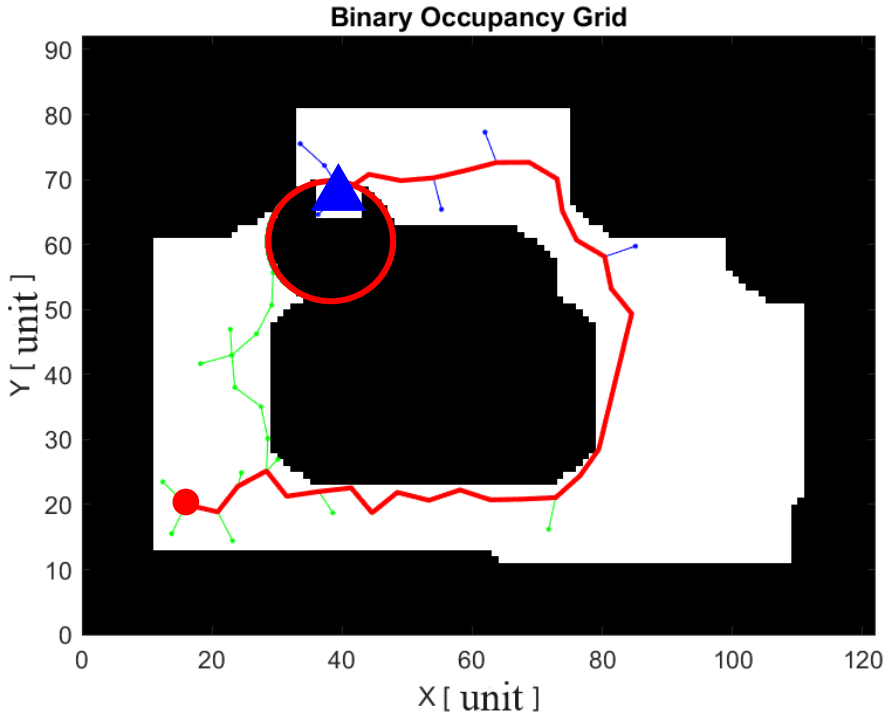
Added Obstacle Area

Experiment Result: Unknown Static Obstacle BiRRT (On-the-shelf Algorithm)

Global path planning from Station 2 to Station 1



Local path planning with unknown static obstacle



Starting Point

Goal Point

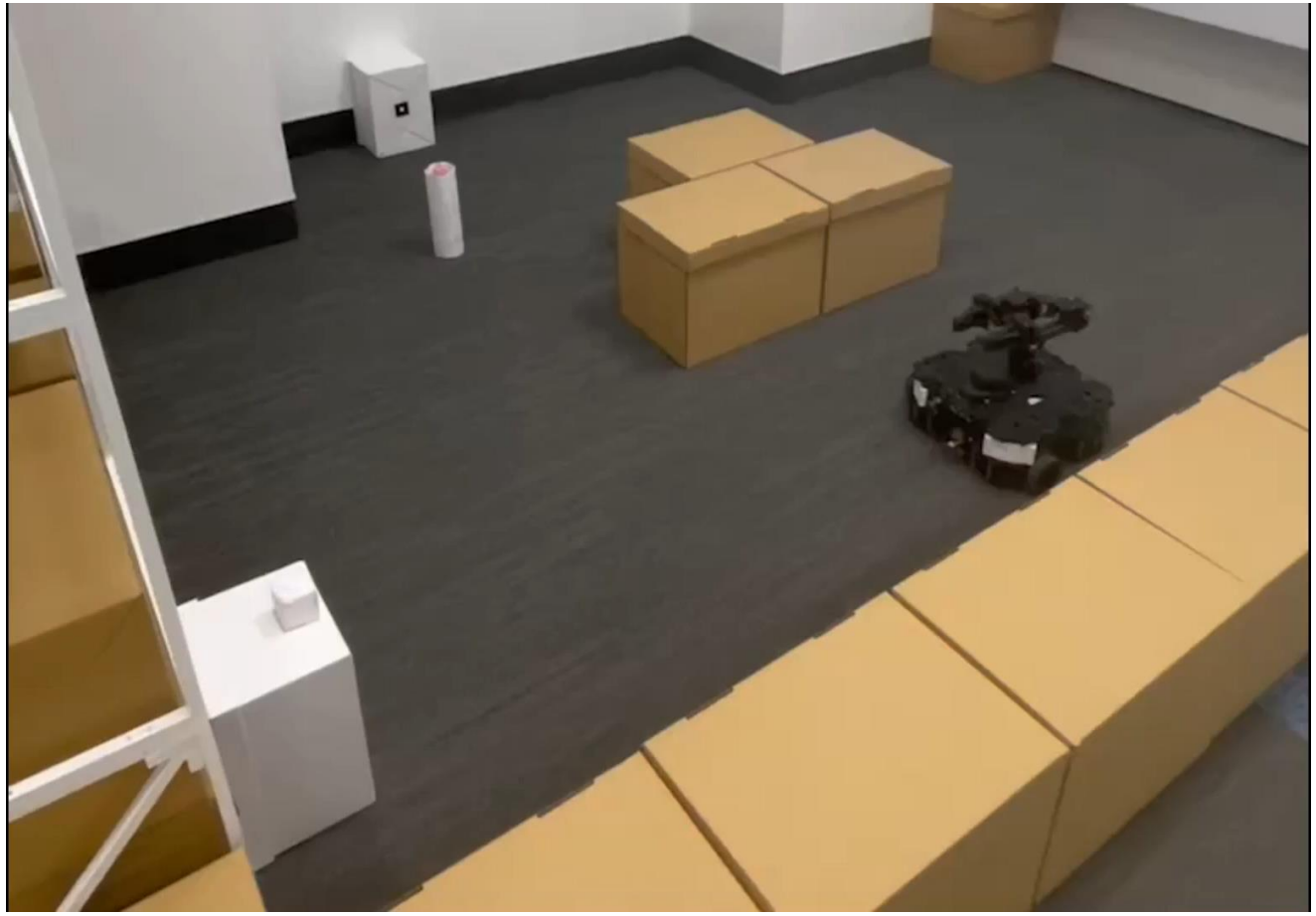
Added Obstacle Area

Computational Results

Path Planning Results

Algorithms	Computational Time (s)	Path Length	Overall Motion Time (s)
Global Path Planning Phase			
Improved RRT*	1.367	38.68	-
BiRRT	0.416	66.07	-
Local Path Planning Phase			
Improved RRT*	6.568	82.81	-
BiRRT	1.539	147.21	-
Overall Phase			
Improved RRT*	7.935	89.52	69
BiRRT	1.955	151.55	143

Recording of the Experiment



Conclusion

- A reduced random map size technique was proposed to reduce the used node from the feasible region in a global environment
- The number of used iterations was reduced
- Better results were given in two situations, simple and complex environments
- The proposed RRT* algorithm can explore the environment faster than the traditional RRT* algorithm
- Unknown static obstable avoidance was realized

Multi Mobile Robots Path Planning by Improved RRT* Algorithm with Right-of-Way Rule

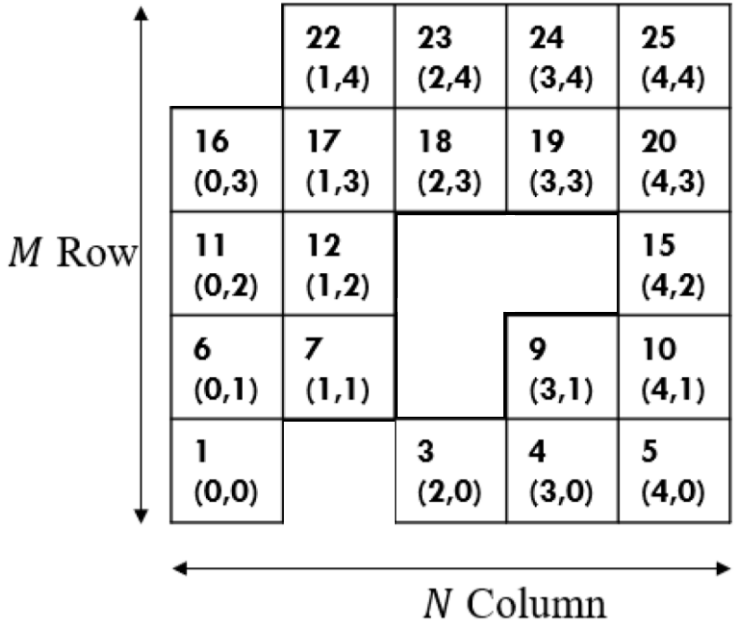
Background

- Using multiple robots is one of the challenges to the path-planning algorithm
- Several challenges of multi-robot path planning
  1. Collision Avoidance
  2. Path Optimality
  3. Scalability
  4. Dynamic Environments

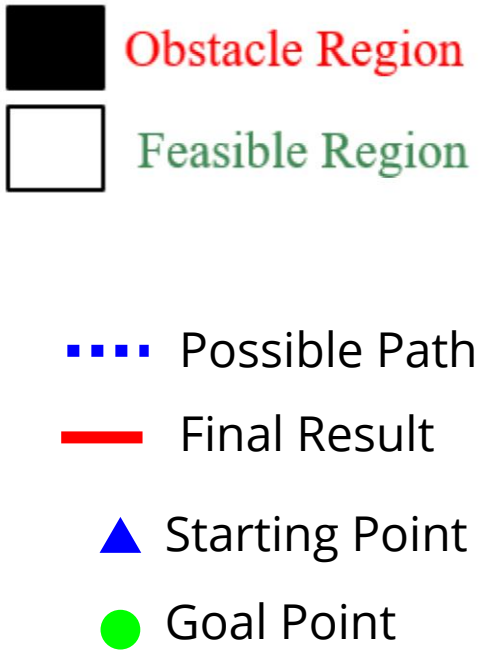
Waffle Pi Robot

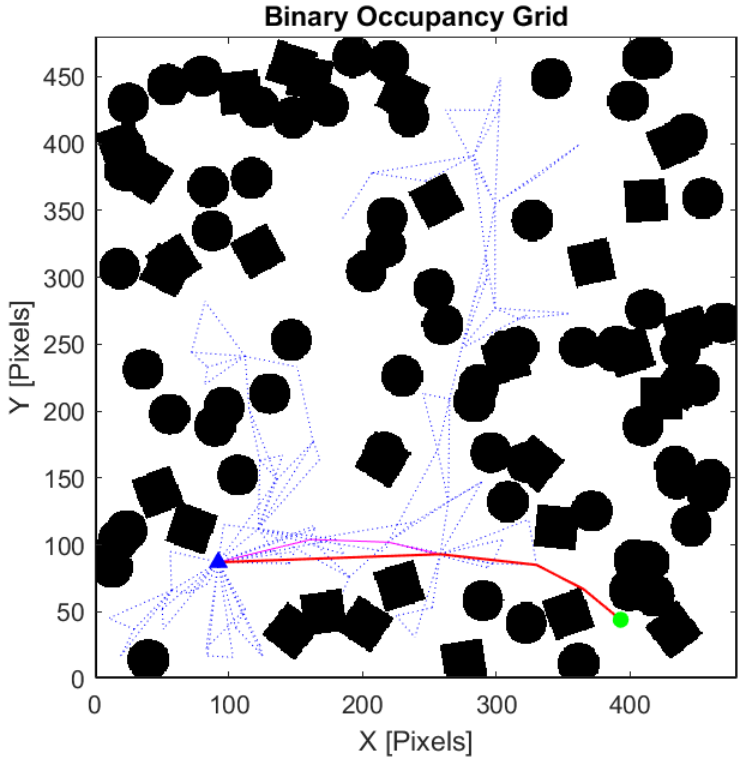
Previously Proposed Algorithm

- The Improved RRT* with Reduced Random Map Size (IRRT*-RRMS)



Environment Mapping

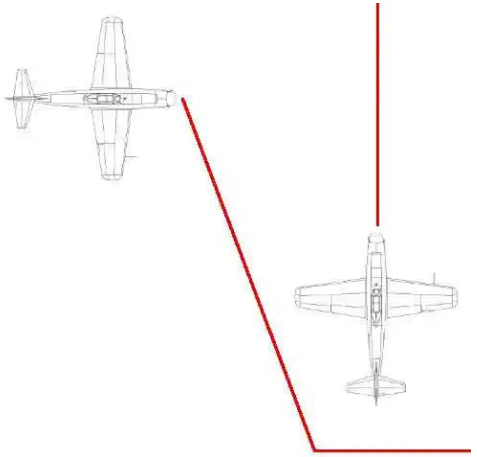




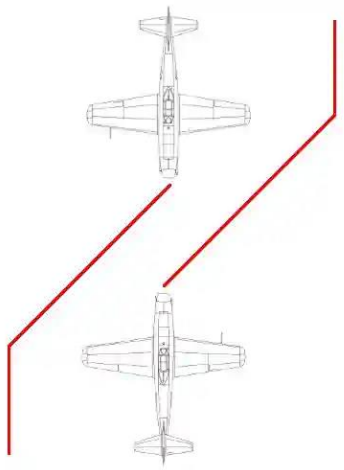
IRRT*-RRMS Path-planning Result

Right-of-Way Rule in Airplane Traffic

- The pilot shall give way to that aircraft and may not pass over, under, or ahead of it unless well clear



Converging
The aircraft to the others right has the right of way

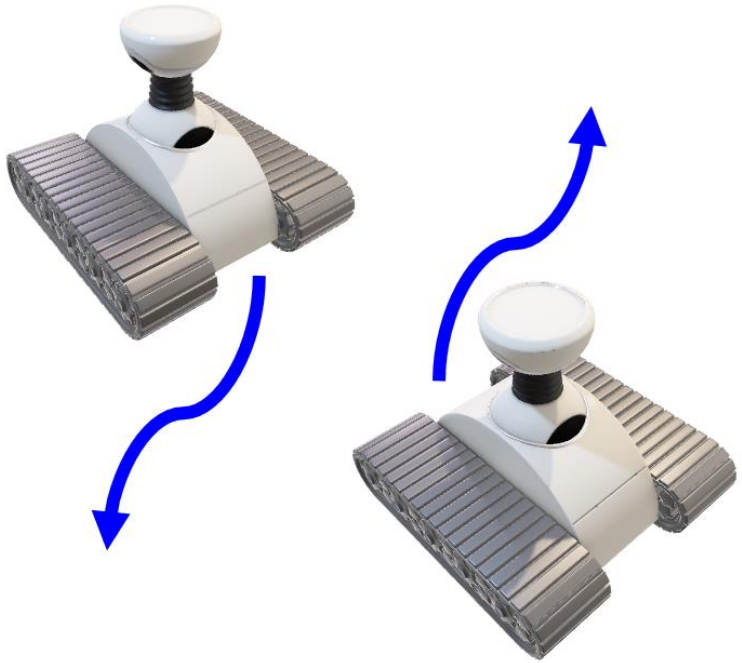


Head-on
When approaching head-on or near head-on each pilot shall alter course to the right

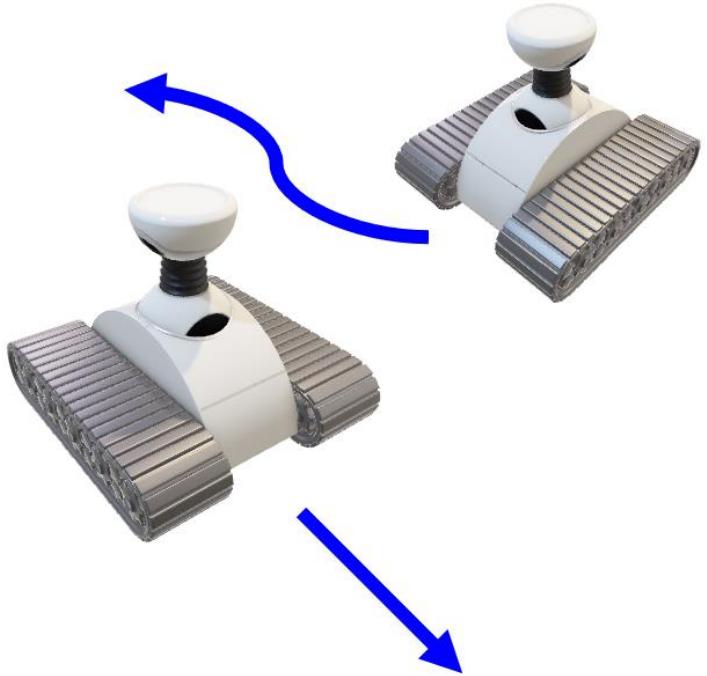
Right-Of-Way (cfinotebook.net)

Right-of-Way Rule for Multi-robot

Head-on Situation

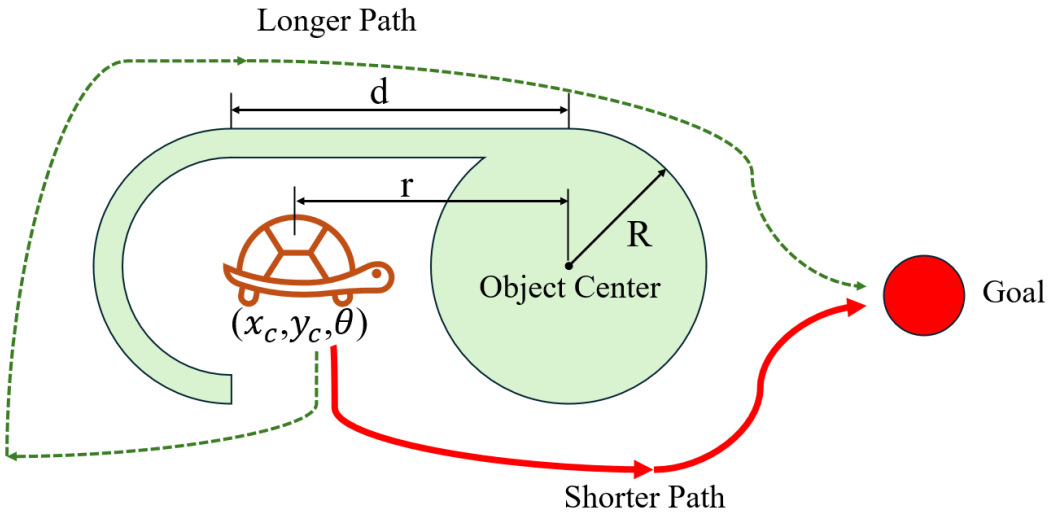


Converging Situation



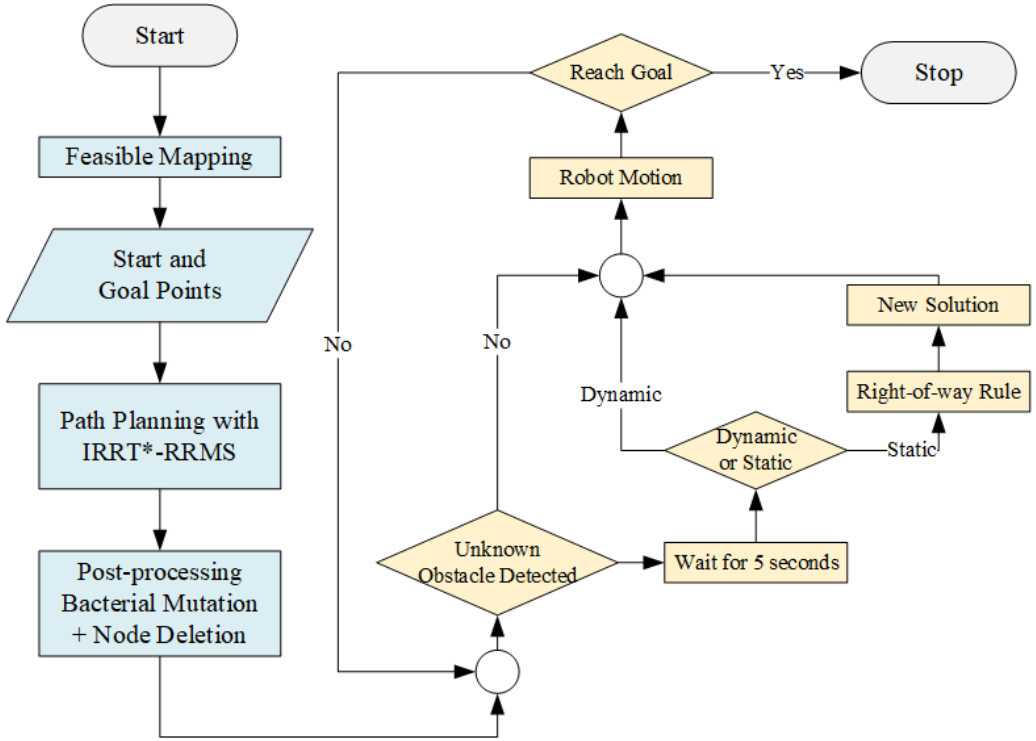
Propose Algorithm

Virtual Obstacle



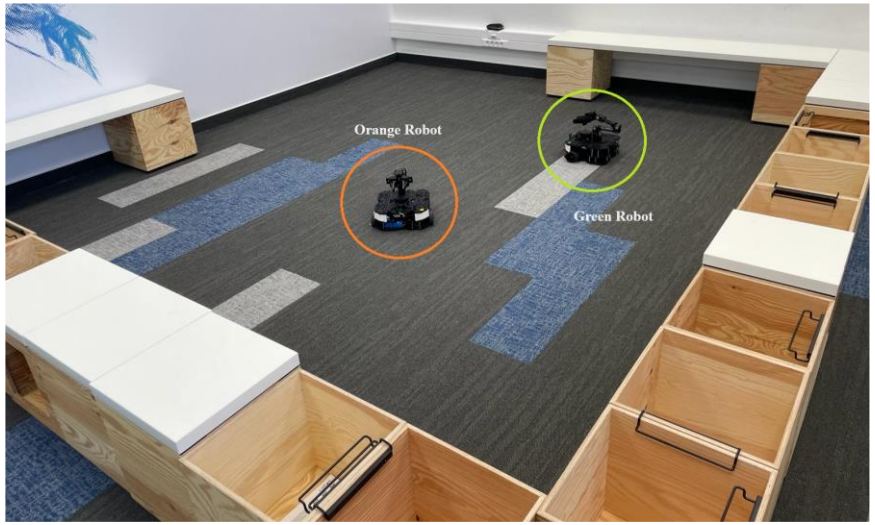
Virtual Obstacle Illustration.

Flowchart



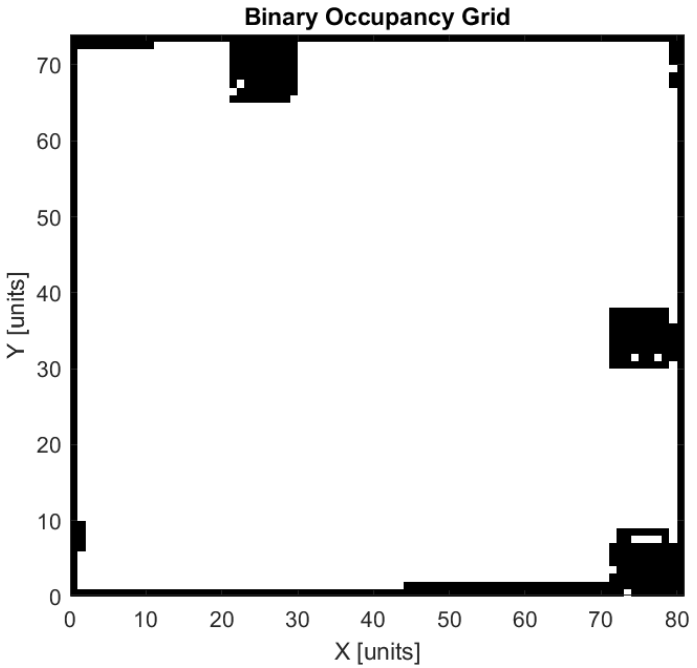
Environment and Mapping

Real-world Environments (6.60 meters x 6.00 meters)

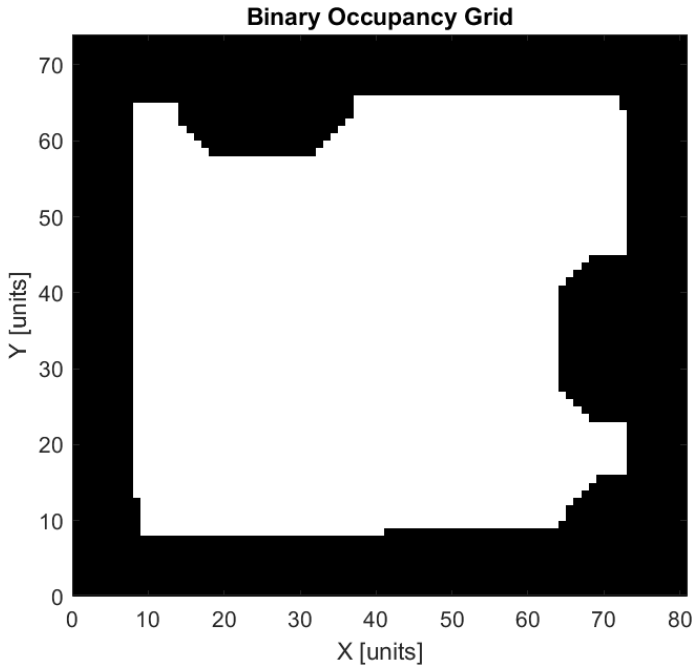


Real-world
Global Environment and Robots

Scanned Map from ROS



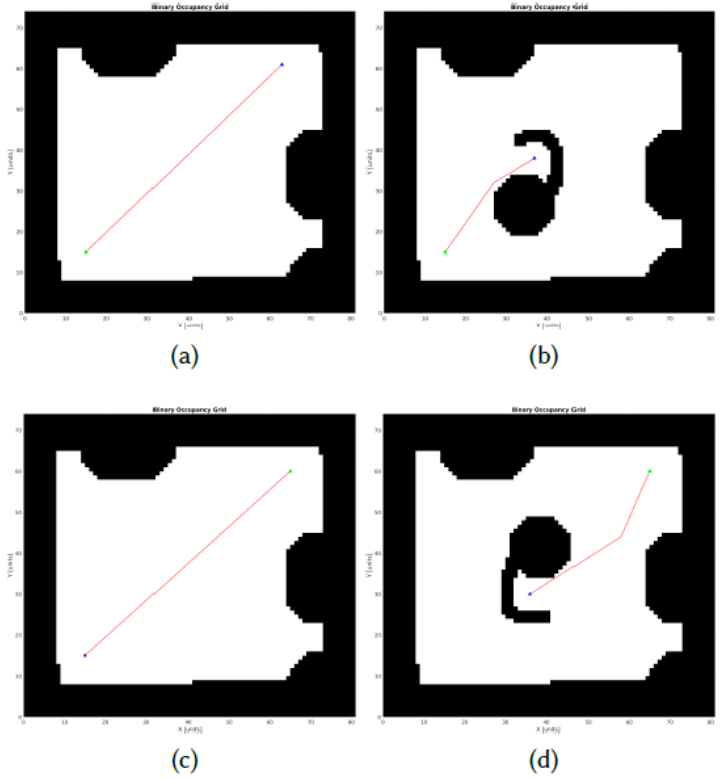
Global Environment
Original SLAM Version



Global Environment
Extended Area Version

Case 1: Head-on with No Obstacles

Path Planning Results



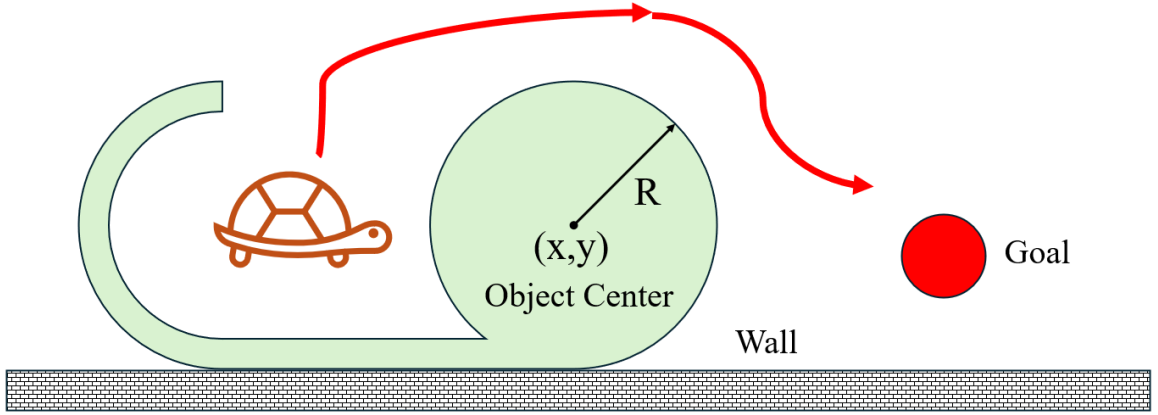
Case 1: Head-on with No Obstacles
(a) OR, Global Path Solution,
(b) OR, Local Path Solution after Head-on Situation,

(c) GR, Global Path Solution,
(d) GR, Local Path Solution after Head-on Situation.

Recording

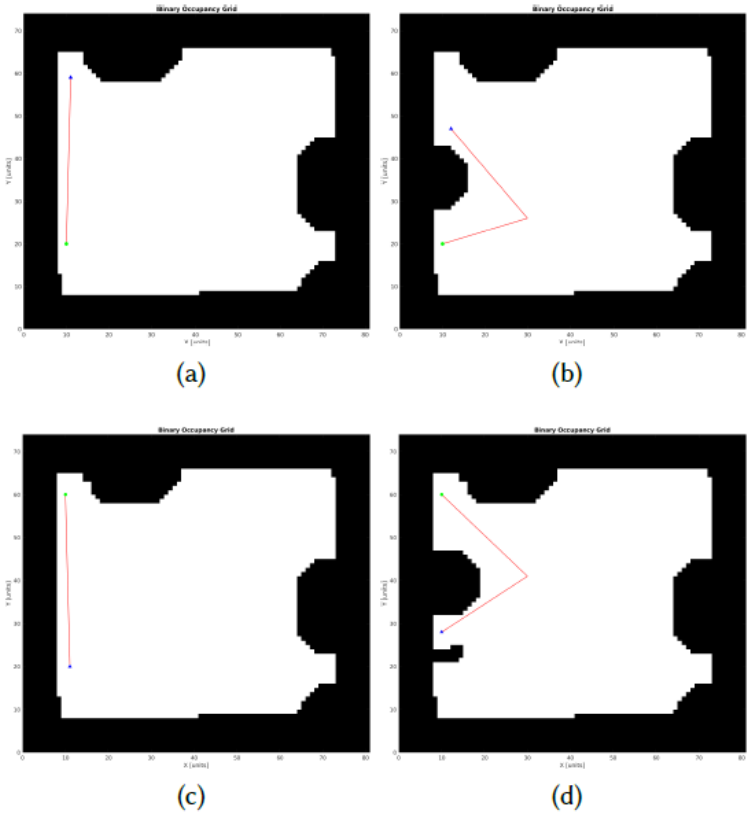
Disadvantage of RoW Rule

- When the mobile robot applies the RoW rule by creating a virtual obstacle in front
- However, it will subsequently encounter a situation where the nearest path is interrupted by a wall
- The inverse of the RoW rule could be applied, called the Left-of-Way Rule (LoW)



Case 2: Head-on with Static Unknown Obstacle and Close to the Wall

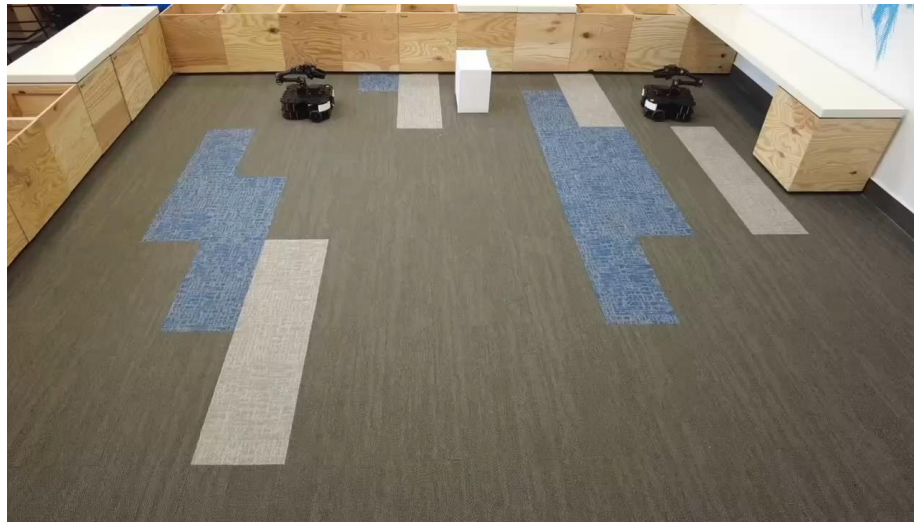
Path Planning Results



Case 2: Static Unknown Obstacle and Close to the Wall
(a) OR, Global Path Solution,
(b) OR, Local Path Solution

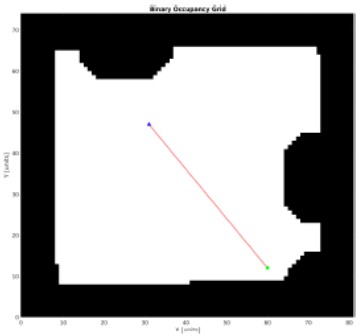
(c) GR, Global Path Solution,
(d) GR, Local Path Solution

Recording

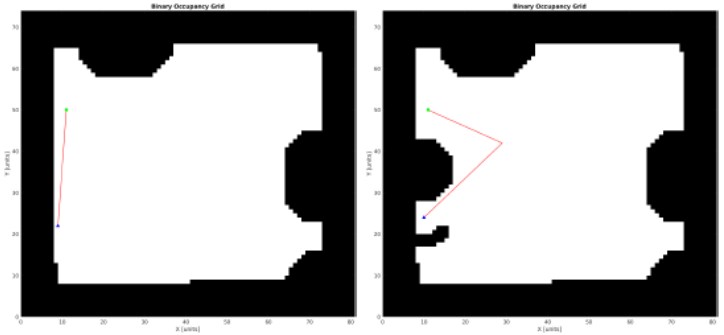


Case 3: Dynamic Unknown Obstacle with Converging Situation

Path Planning Results



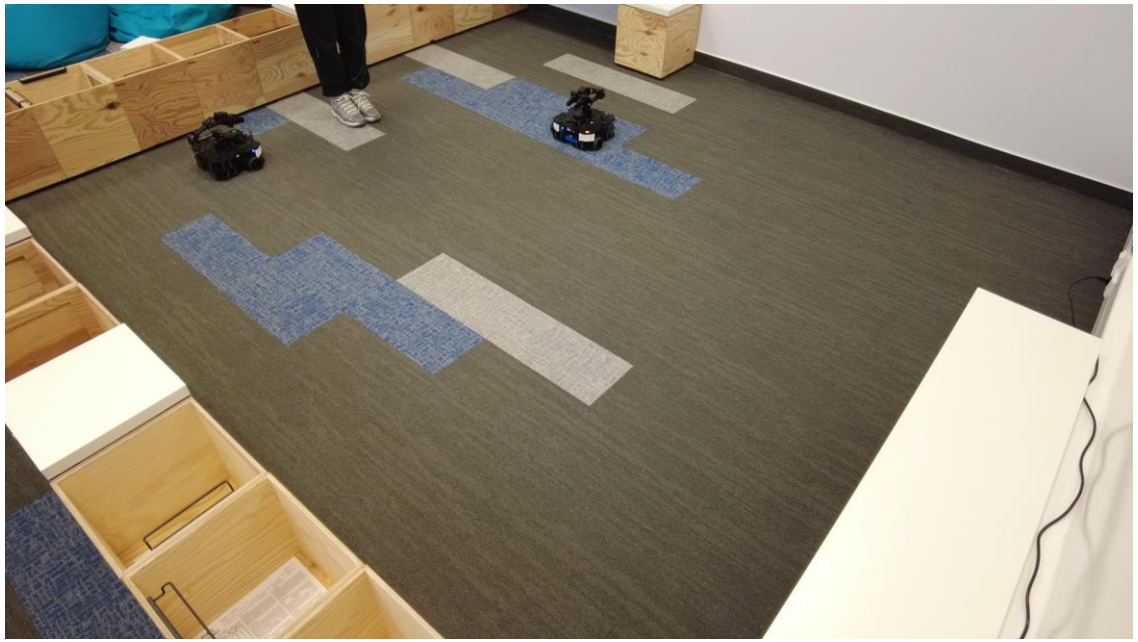
(a)



(b) (c)

Case 3: Dynamic Unknown Obstacle with Converging Situation
(a) OR, Global Path Solution,
(b) GR, Global Path Solution
(c) GR, Local Path Solution.

Recording



Conclusion and Future Works

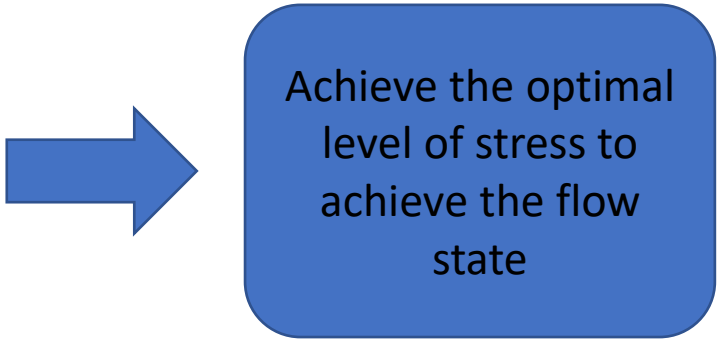
- The path planning results from the IRRT*-RRMS, combined with the Right-of-Way Rule, are collision-free
- The proposed algorithm achieves the multi-robot path planning tasks, which can be reviewed in the real-robot implementation
- The virtual obstacle technique can lead the previously proposed path planning algorithm to achieve the dynamic path planning scenario

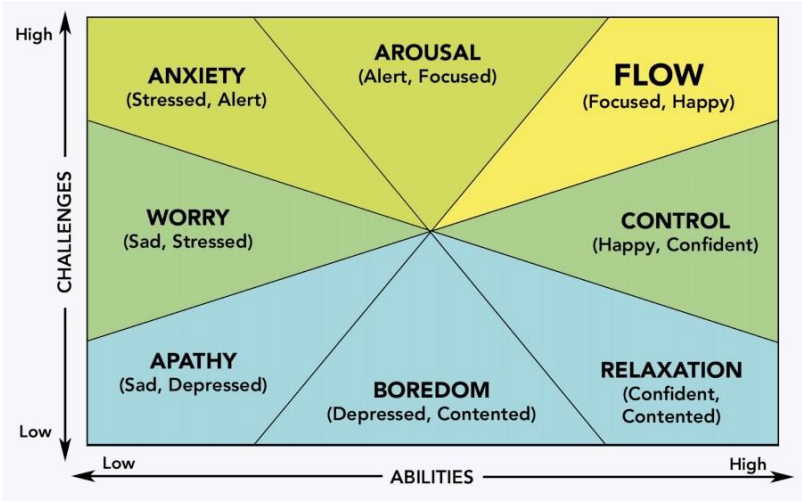
Workpace optimization

New problems in close cooperation with robots

- Robots working in close cooperation needs to adjust their behavior
  - Internal factors (fatigue etc.)
  - External factors (fear from the robot etc.)
- External factors can be improved by a sophisticated robot behavior
  - Velocity
  - Acceleration
  - Time of immobility
  - Trajectory type

} Process time





Workpace

- Robot + Operator = increase in productivity in many cases
- Workpace is essential!
  - A higher skilled operator can feel boring (and stressed) with an optimal pace of a lower skilled operator. On the other hand a lower skilled operator will face fatigue (and stress).
  - The workpace need of an individual can alter also during the day.

https://theconversation.com/why-boredom-can-be-good-for-you-90429

Both case can lead to increase in stress and thus in deficit in productivity

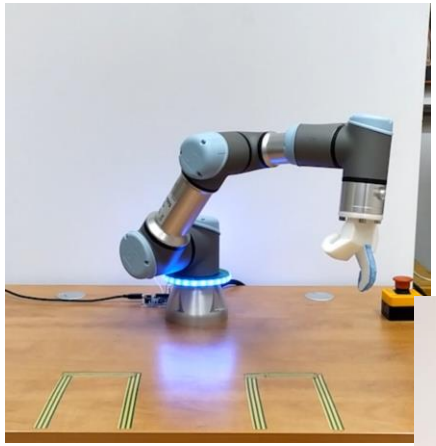
Bacterial Evolutionary Algorithm

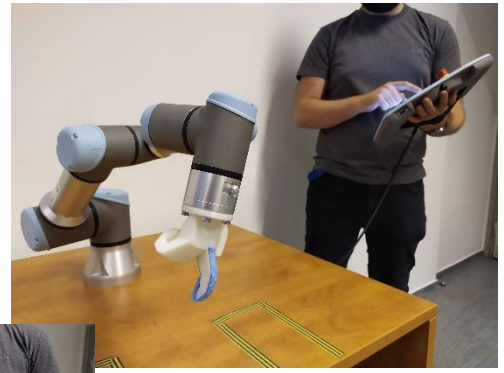
https://ideascale.com/impact-of-stress-on-innovation/

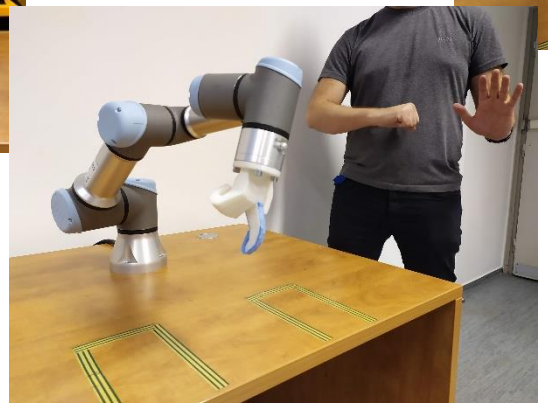
Problems with the cobot applications

- Most cobot applications are low-level applications in terms of interactivity
- Applications lack proper HRI and robot learning

https://robots.ieee.org/robots/lbriiwa/

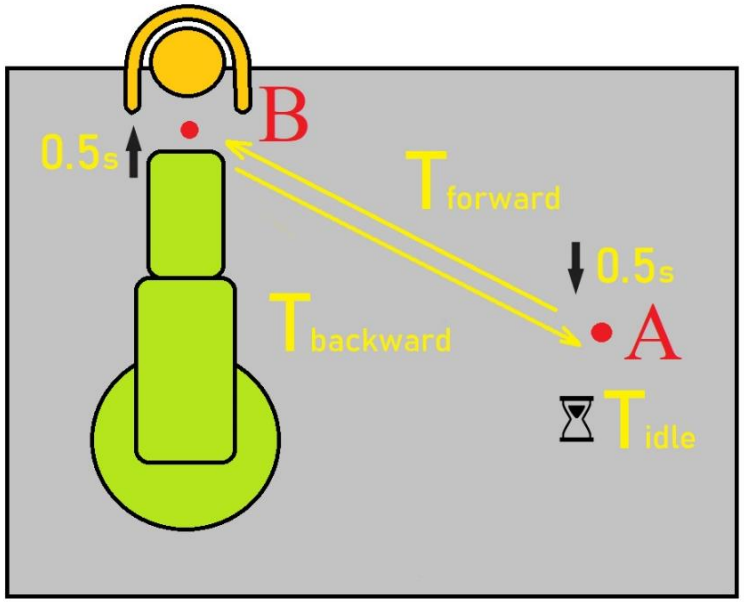






Implemented scenario

- Hand in task as initial test
  - Plastic workpiece quality check
- Simplified scenario
- A, B, and 2 points 10 cm above them
- Vertical movement has fixed time of 0.5s
- 3 variables:
  - $T_{forward}$
  - $T_{backward}$
  - T_{idle}
  - Each variable is in the range of [0.2; 2) s with a 0.1 s resolution
  - 5832 variations



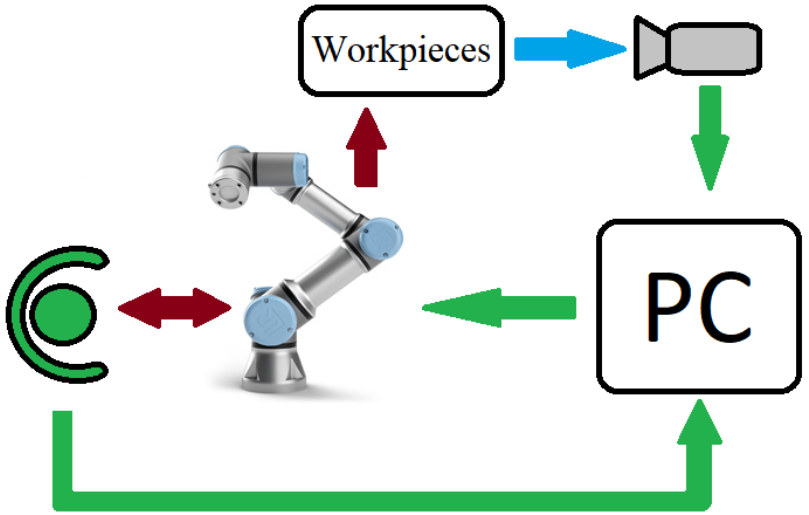
Bacterial Evolutionary Algorithm

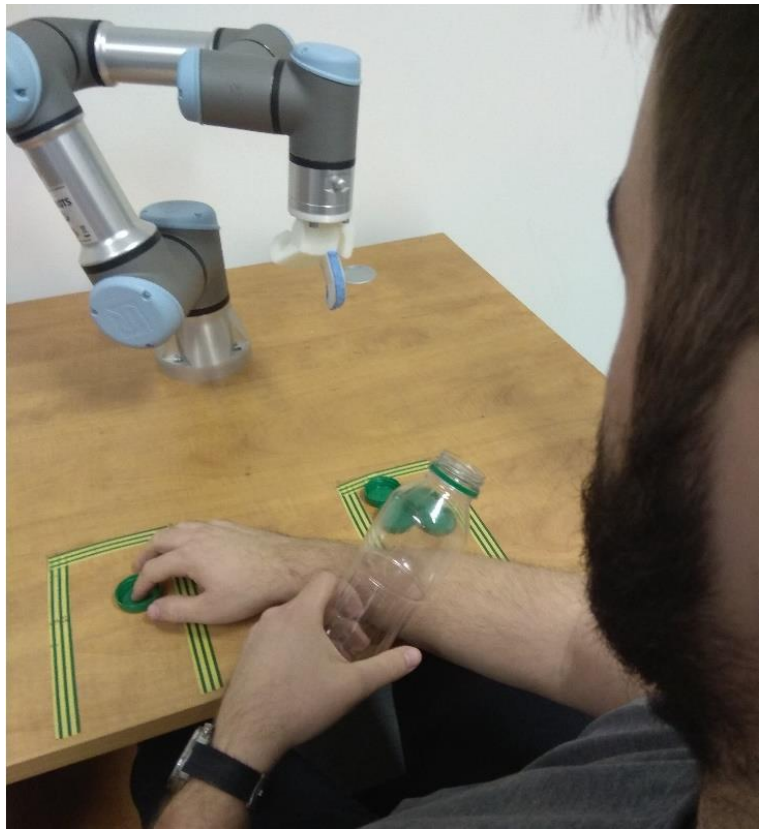
From human:
- Simple feedback
- HRV

From task:
- Total time

Experimental setup

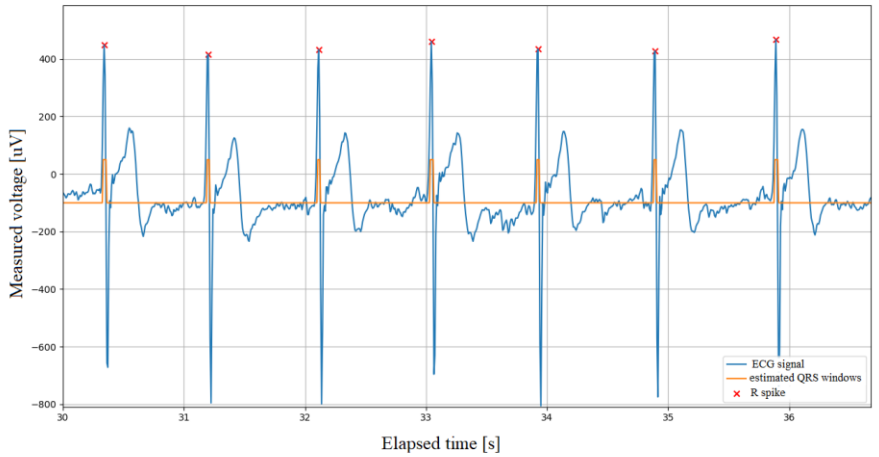
- UR3e collaborative robot
- 3D printed hand
- Basler ACA800-510UC + f6mm objective
- Plastic bottle + caps

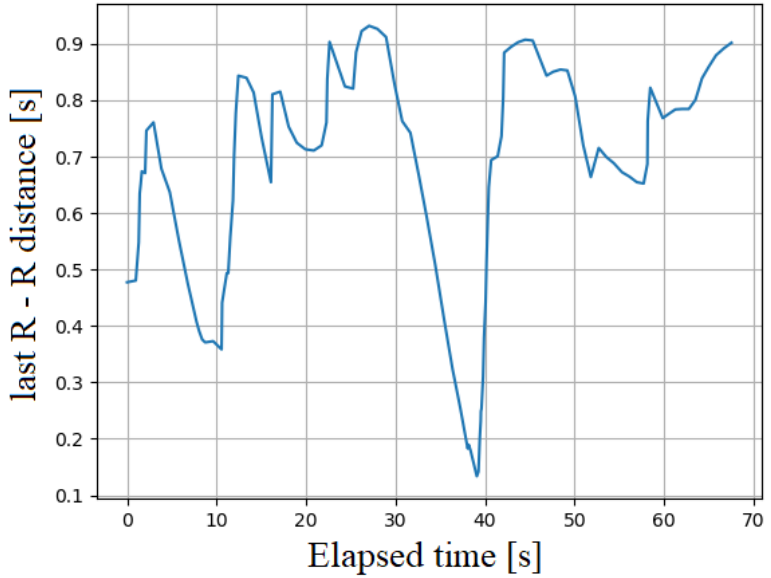




- 5 repetitions of task
- 1.5 min

Experimental setup





- Polar H10 sport belt
- After processing the ECG signals the R-R distances are calculated as an input for the algorithm

Experimental setup

Proposed algorithm

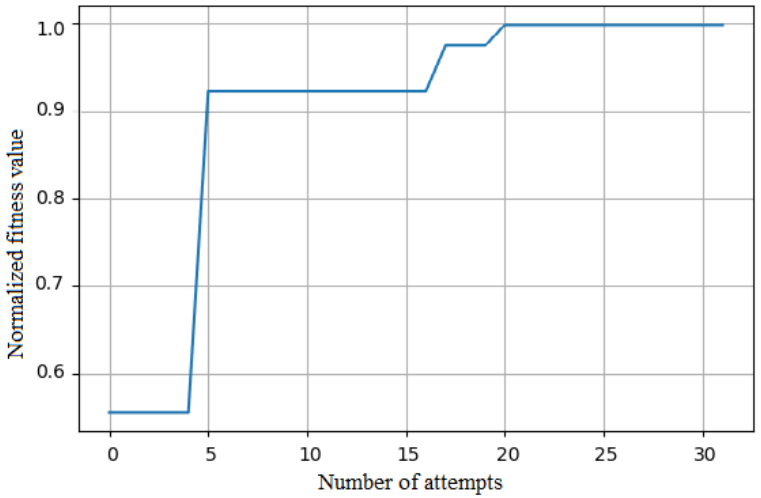
- Probabilistic bacterial mutation:
  - The bacterial mutation operator is applied by a given probability to each individual.
- Interactivity in the fitness evaluation:
  - In interactive evolutionary algorithms the fitness value of the individuals is determined by a human.
  - We combine the human's subjective evaluation with other objective, measurable values in the fitness calculation.
  - The fitness of an individual is calculated as:

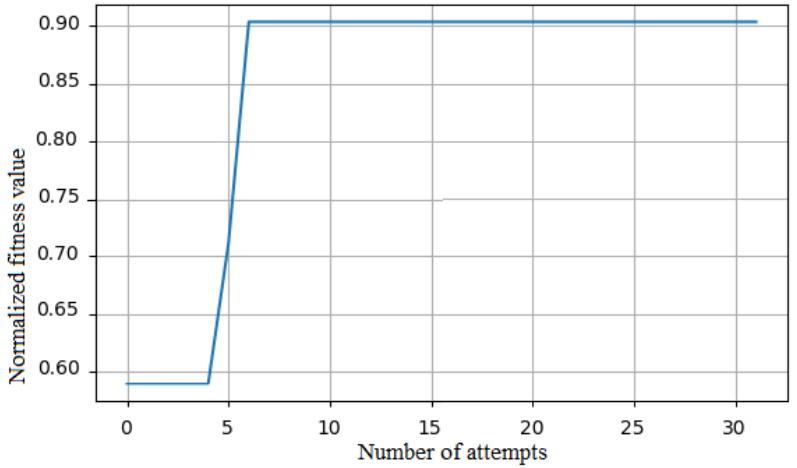
$$fitness = SF - 10\frac{RR_{ref} - \overline{RR}}{RR_{ref}} - \frac{CT}{2}$$

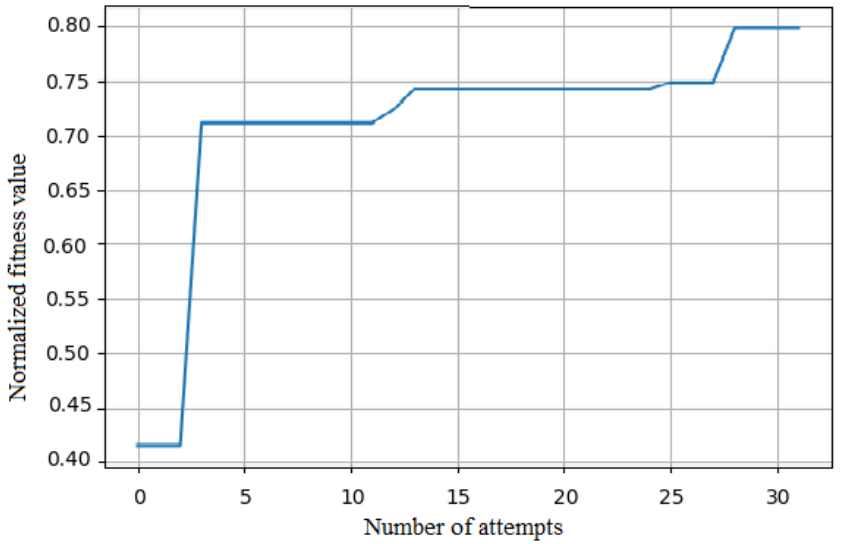
  - SF is the subjective feeling points (between 1 (worst) and 10 (best))
  - $\underline{RR}_{ref}$ is the reference R to R distance measured before the experiment
  - $\overline{RR}$ is the mean of the R to R distances
  - CT is the cycletime:

$$CT = T_{forward} + T_{backward} + T_{idle}$$

Experimental results







Three measurements