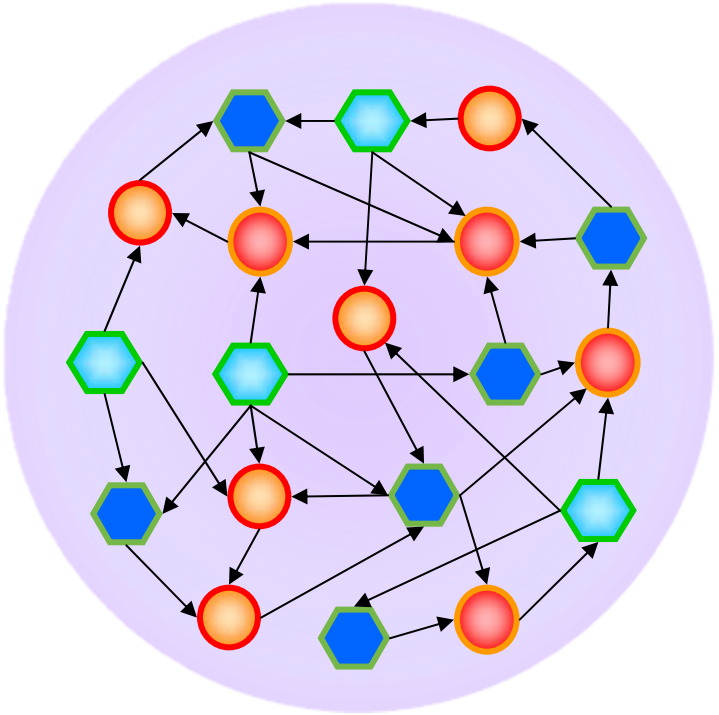
Genetic Network Programming with Reinforcement Learning Using Sarsa Algorithm

Shingo Mabu, Hiroyuki Hatakeyama,
Kotaro Hirasawa and Jinglu Hu
Waseda University, Japan

Contents

I. Research Background

- **Genetic Network Programming (GNP)**
  - GNP consists of **Judgment nodes** and **Processing nodes**
  - Characteristics
    - directed graph structure
    - compact structure
    - implicit memory function



Basic structure of GNP

Objective

- Making graph structures by combining **evolution** and **reinforcement learning (RL)**
  - Evolution: Many individuals work and better ones are selected after task execution.

Diversified search

  - RL: One individual works and learn action rules during task execution.

Intensified search

search for better solutions
and improve the search speed

Objective

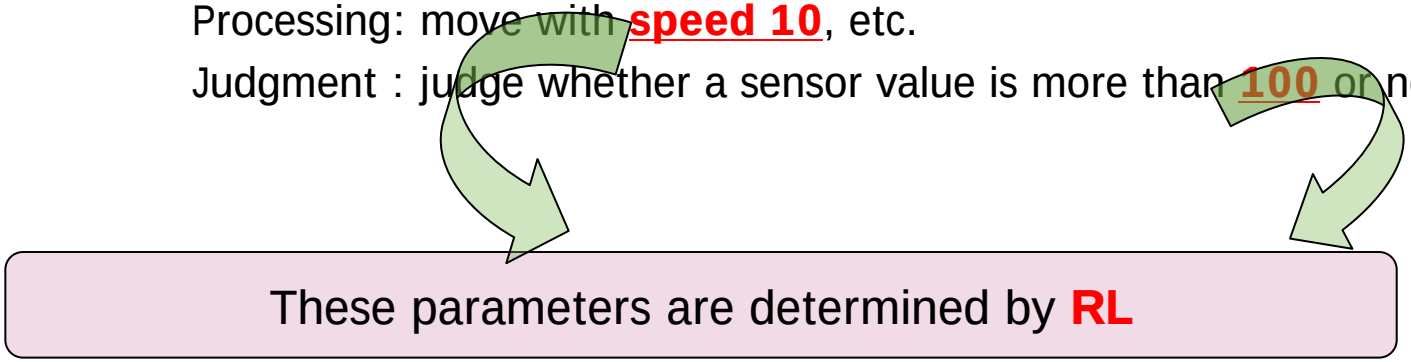
- Dealing with numerical inputs and outputs
  - Conventional GNP
    - Ex) robot behavior:

      Processing: move forward, turn right, etc.

      Judgment : judge whether an obstacle **exists or not** in front of the robot
  - Proposed method

    Processing: move with **speed 10**, etc.

    Judgment : judge whether a sensor value is more than **100** or not



Details

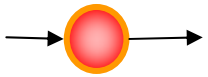
- The role of evolution and RL
  - Evolution
    - **Determine GNP structures** (connections between nodes) and **randomly change node parameters** after task execution (offline)
  - RL (Sarsa)
    - **Select appropriate parameters** based on the information obtained during task execution (online)

- **Why Sarsa ?**

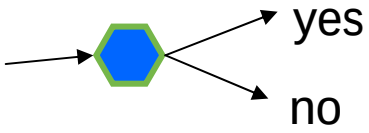
  Sarsa can learn action rules considering the action policy (e.g. ε -greedy) used by an agent

II. Genetic Network Programming with RL

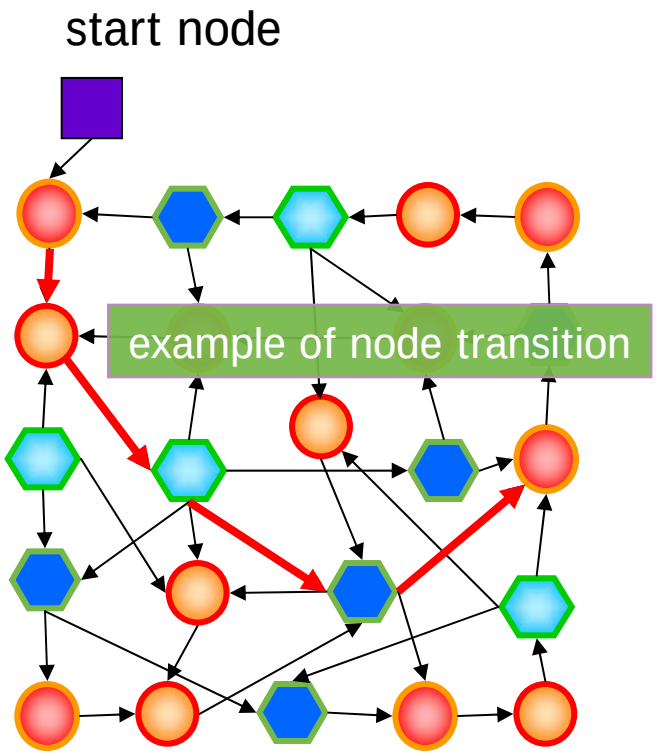
- Components
  - Processing node
    - Each processing node has the unique action function.

Ex.)  Move with speed 10

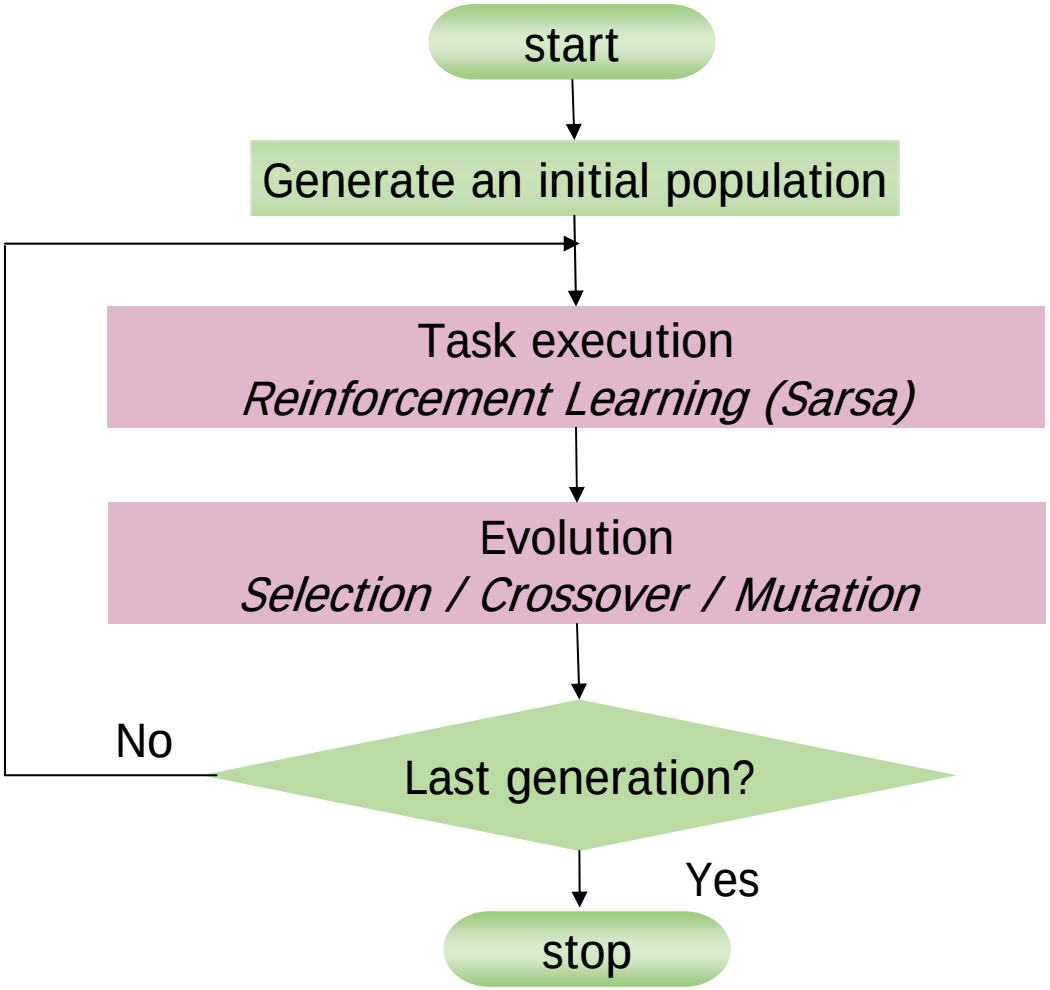
  - Judgment node
    - Each judgment node has the unique judgment function

Ex.) 

A sensor value is more than 100 or not

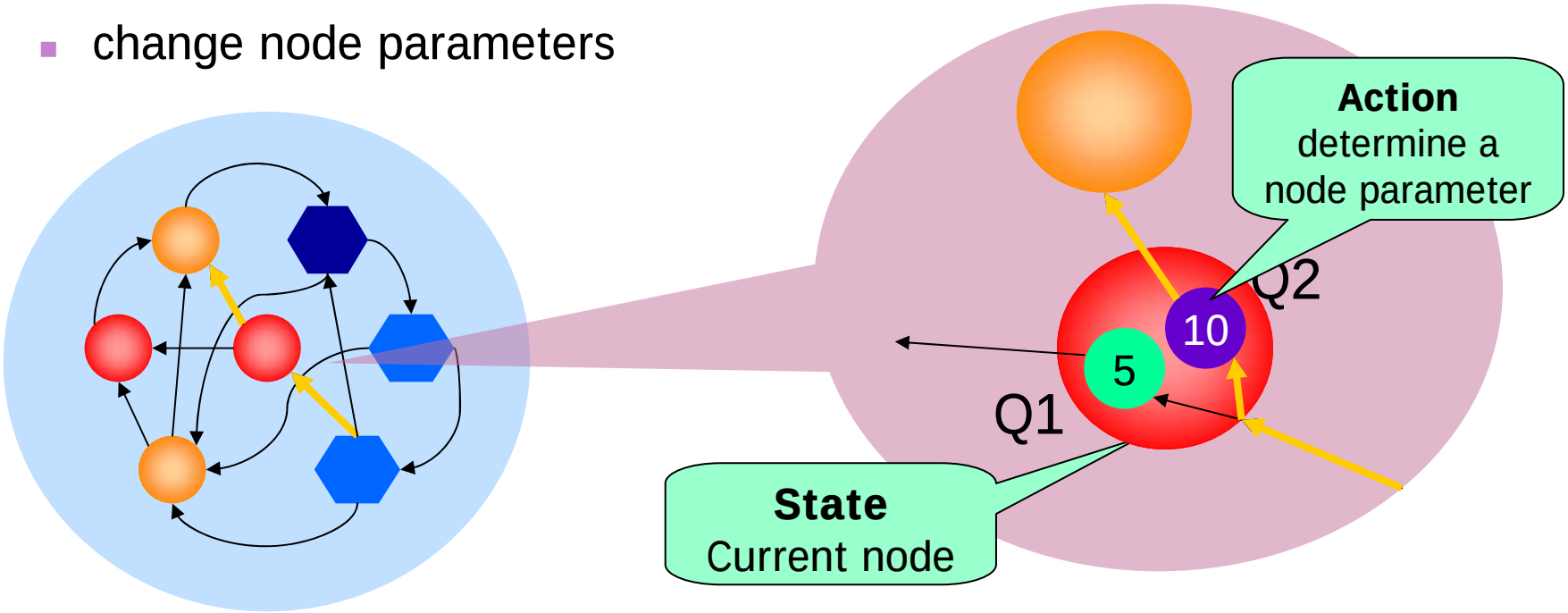


Flowchart


start
Generate an initial population
Task execution
Reinforcement Learning (Sarsa)
Evolution
Selection / Crossover / Mutation
Last generation?
No
Yes
stop

The role of evolution and learning

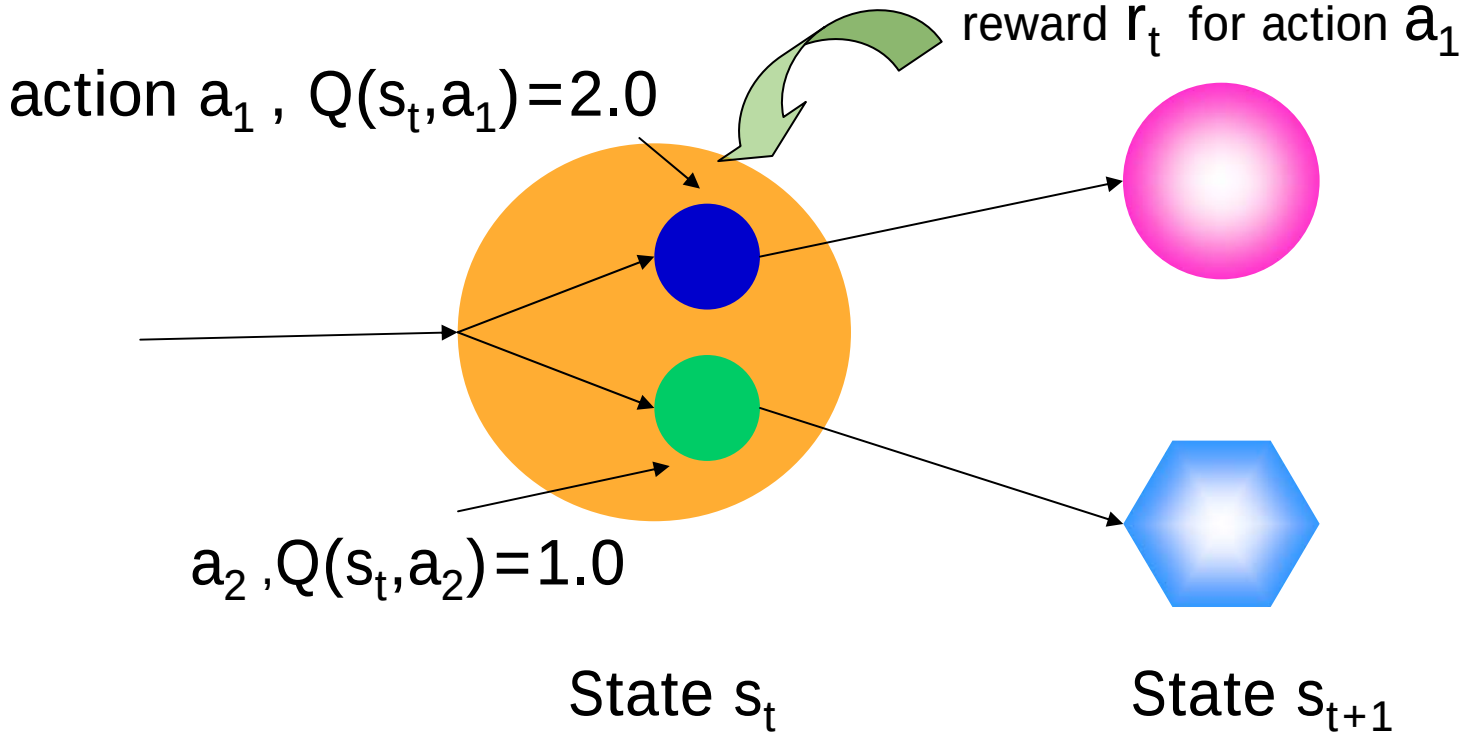
- Evolution
  - determine a structure
    - Node connections
  - change node parameters
- RL (Sarsa)
  - determine node parameters



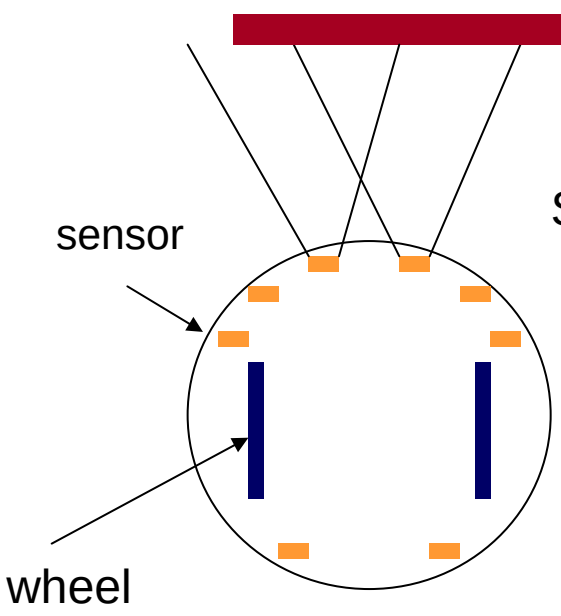
Q1, Q2: action value (Q value)

node transition rules

- Action policy (ε-greedy)
  - Select an action (node parameter) having the max Q value with the probability of 1- ε , or select one of them randomly.



Khepera robot



Sensor value

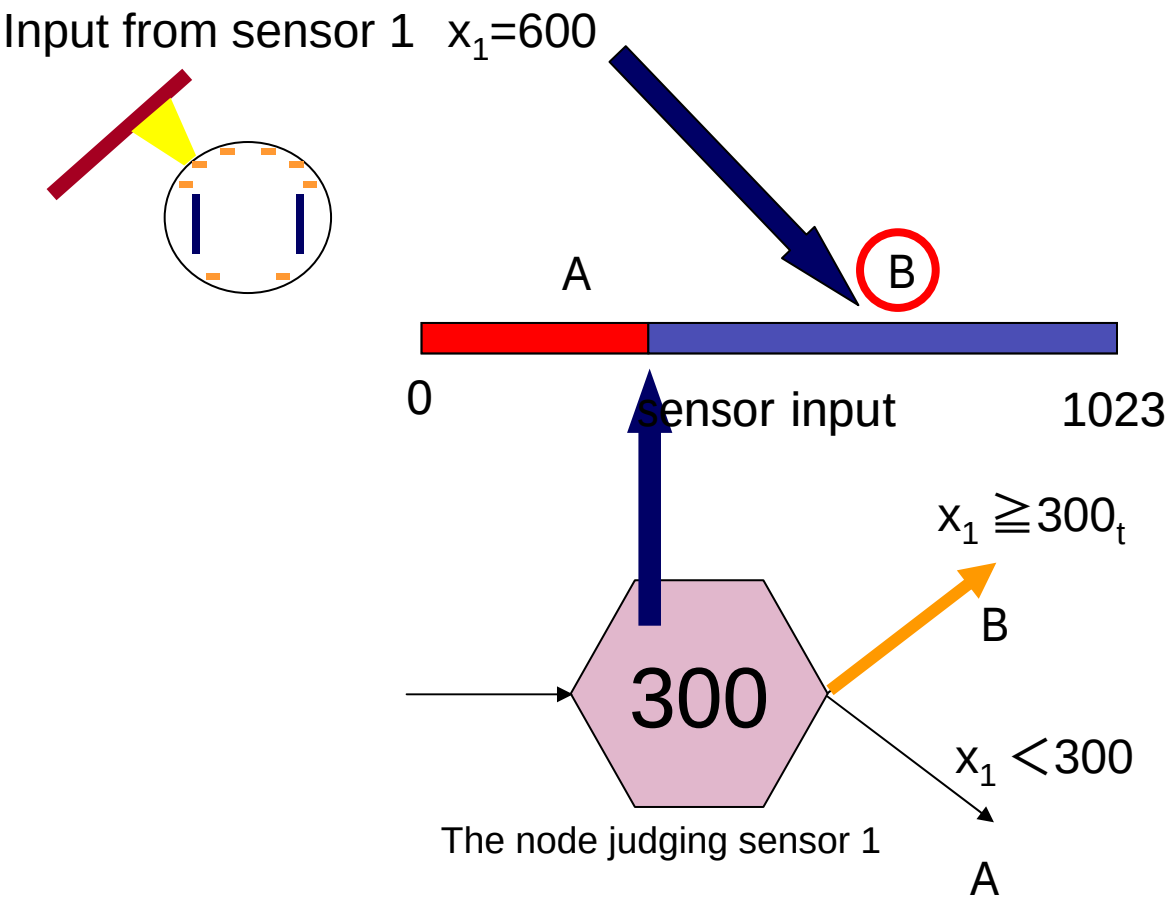
Far from obstacles → Close to zero

Close to obstacles → Close to 1023

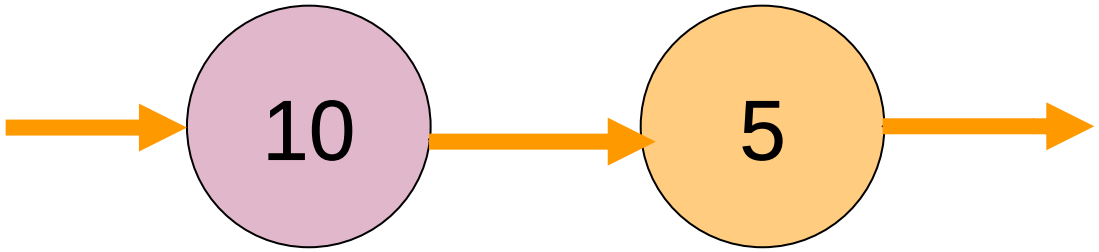
Speed of the right wheel V_R → -10 (back) ~ 10 (forward)

Speed of the left wheel V_L → -10 (back) ~ 10 (forward)

Function of Judgment node

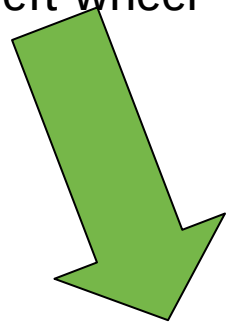


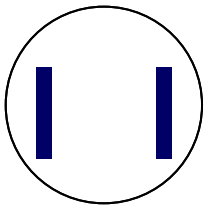
Function of Processing node



Determine the speed of the right wheel

left wheel





$V_R = 10$

$V_L = 5$

Learning node parameters

- Q-values are updated using Sarsa algorithm

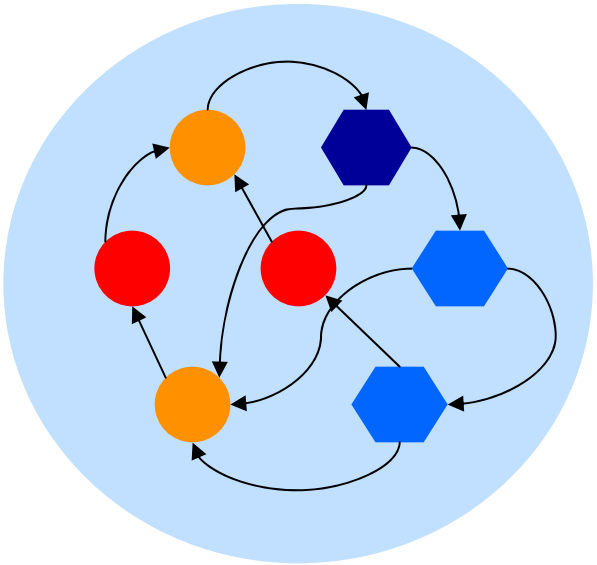
After judgment or processing, execute the following

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

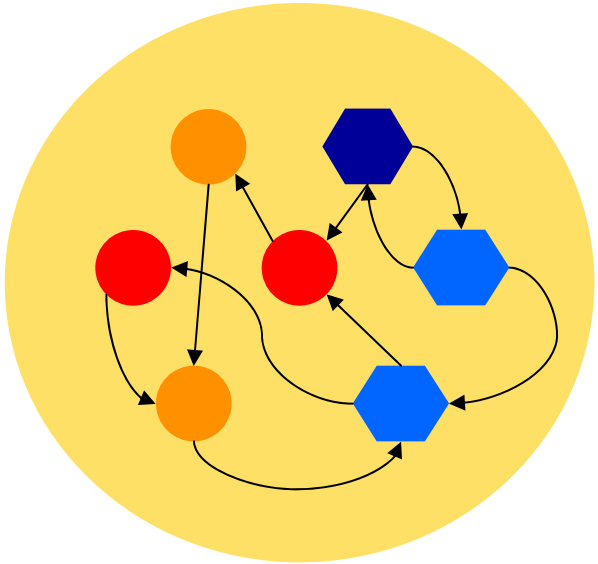
r: reward
α : learning rate
γ : discount rate
$Q(s_t, a_t)$: value of action a_t at state s_t

Genetic operator (Crossover)

1. Two individuals are selected as parents.
2. Each node is selected with the probability of P_c and its genes (content and connections) are exchanged.



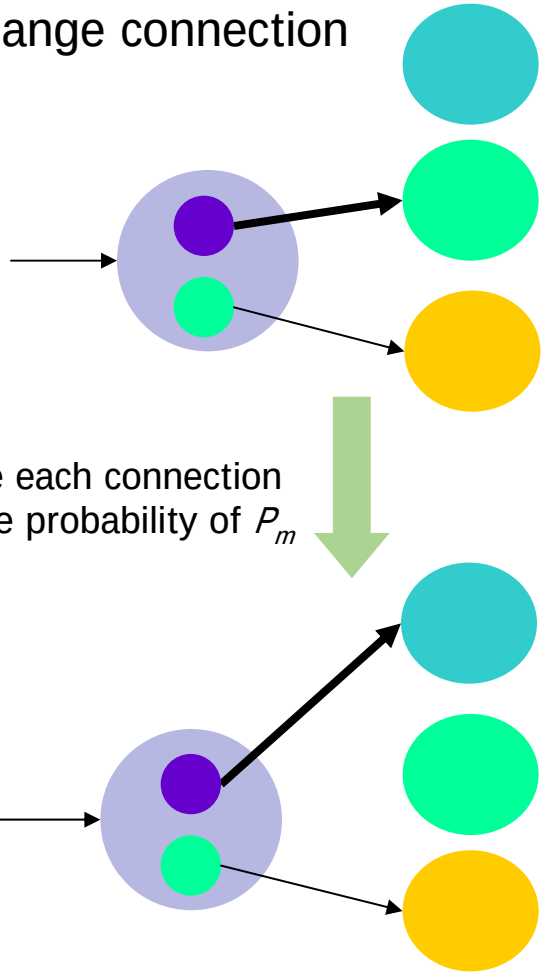
Individual 1



Individual 2

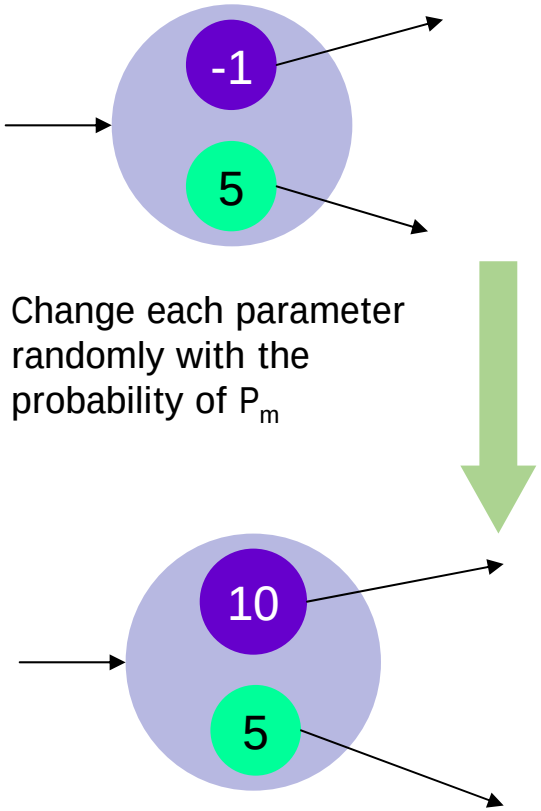
Genetic operator (Mutation)

- change connection



Change each connection with the probability of P_m

- change node parameter



Change each parameter randomly with the probability of P_m

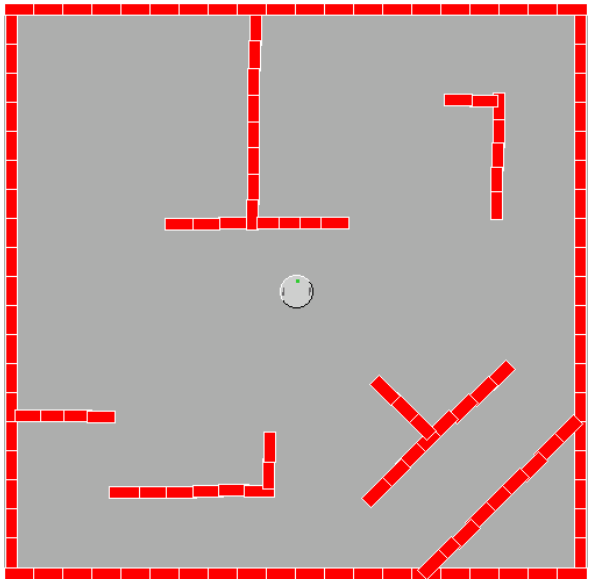
III. Simulation

- Wall-following behavior
  1. All the sensor values are less than 1000
  2. At least one sensor value is more than 100
  3. Move straight
  4. Move fast

$$\text{Reward}(t) = \frac{v_R(t) + v_L(t)}{20} \times \left(1 - \sqrt{\frac{v_R(t) - v_L(t)}{20}}\right) \times C$$

$$\text{fitness} = \left(\sum_{t=1}^{1000} \text{Reward}(t)\right) / 1000$$

$$C = \begin{cases} 1 & : \text{If the condition 1 and 2 is satisfied} \\ 0 & : \text{otherwise} \end{cases}$$



Simulation environment

Node functions

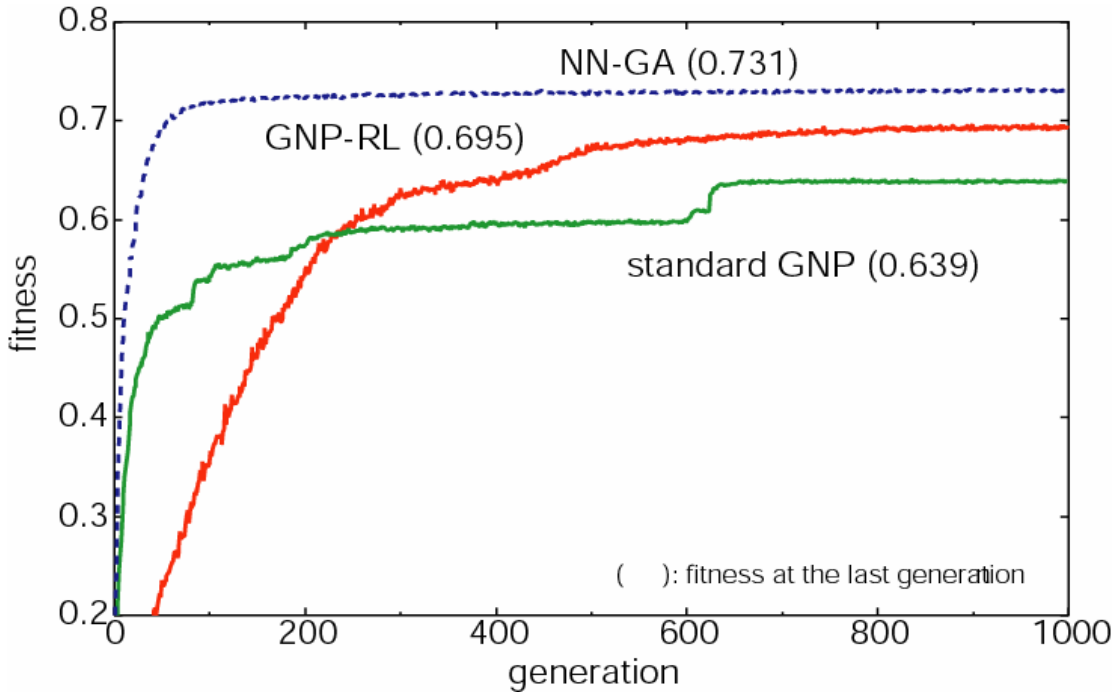
Processing node (2 kinds)	Judgment node (8 kinds)
• Determine the speed of right wheel • Determine the speed of left wheel	• Judge the value of sensor 0 ⋮ • Judge the value of sensor 7

Simulation conditions

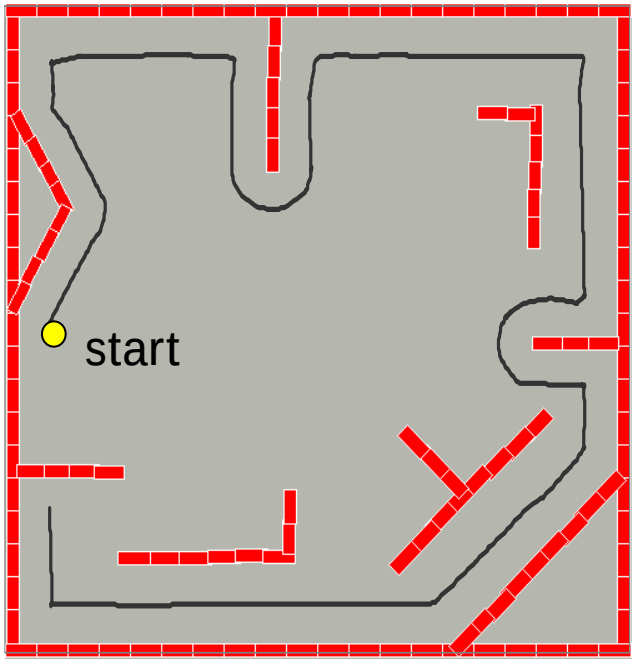
- Evolution
  - The number of individuals: 600
  - The number of nodes: 60
    - Judgement nodes: 40
    - Processing nodes: 20
  - Crossover rate P_c : 0.1
  - Mutation rate P_m : 0.01
- Learning
  - Learning rate α : 0.1
  - discount rate γ : 0.4
  - ε : 0.1

Simulation 1

Start with fixed position



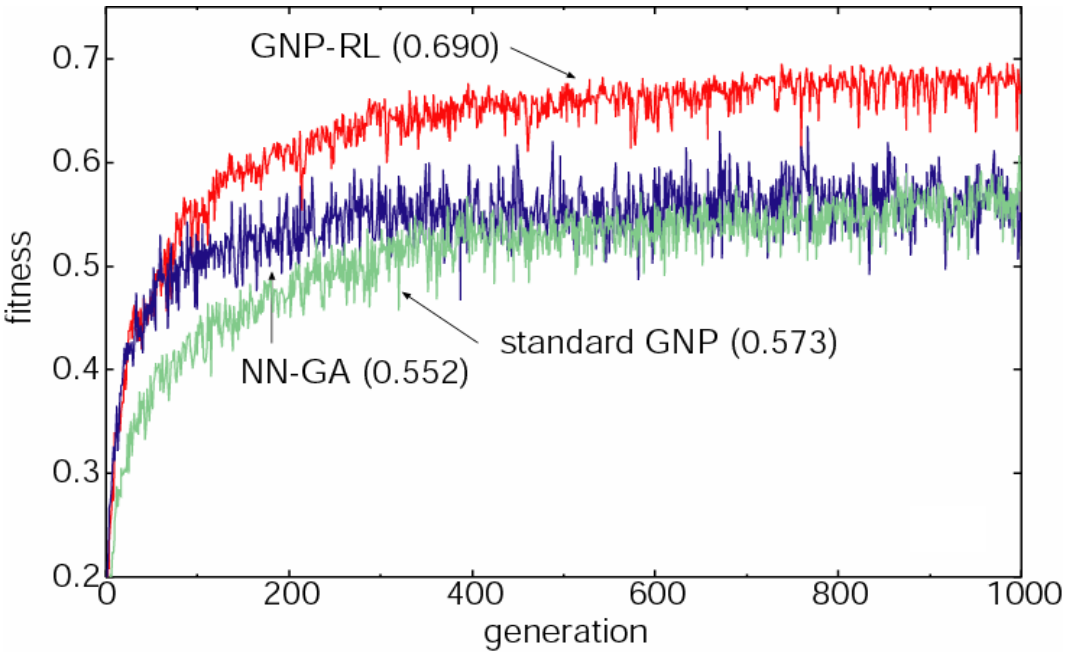
fitness curves of the best individuals averaged over 30 independent simulations



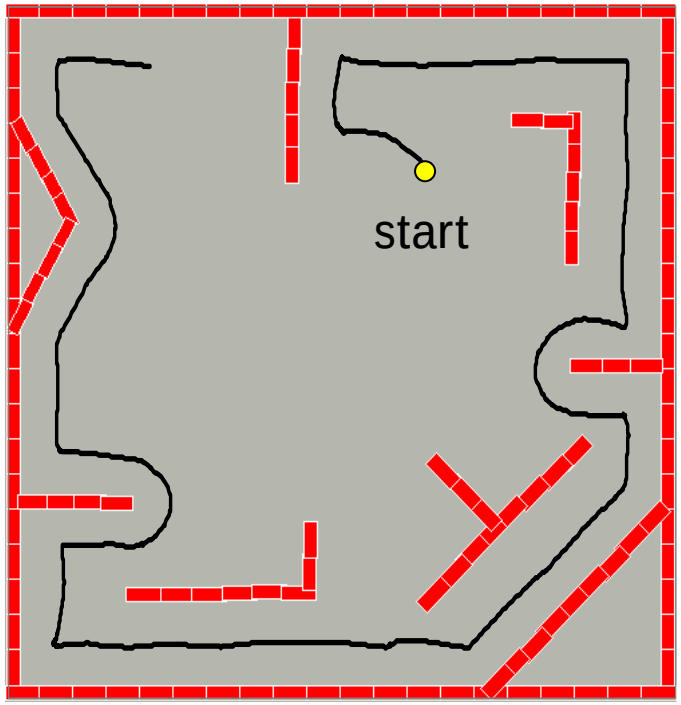
Track of the robot

Simulation 2

- Start with random position



fitness curves of the best individuals averaged over 30 independent simulations

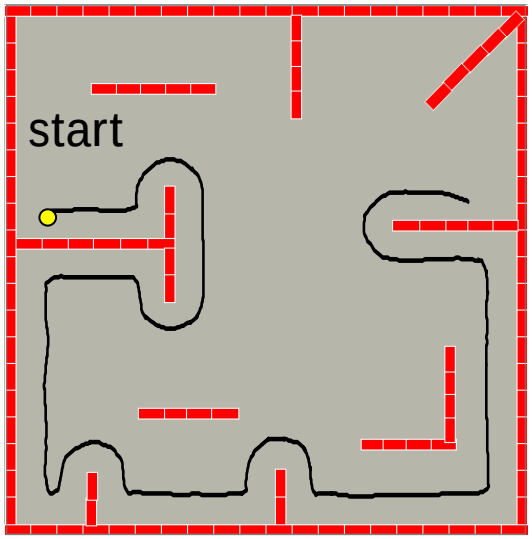


Track of the robot

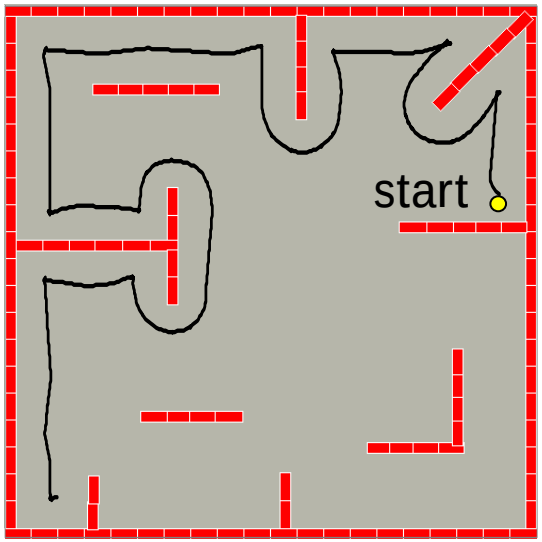
Simulation 3 generalization ability

The best program obtained in Simulation2

Execute in the inexperienced environment



Example 1



Example 2

The robot shows appropriate wall-following tracks

Simulation 3 generalization ability

- Data on the fitness (30 simulations)

	GNP with RL	Standard GNP	NN with GA
Average	0.426	0.356	0.204
Standard deviation	0.082	0.146	0.046
t-test between GNP-RL and each method P-value[%]	———	3.54	5.75×10^{-10}

IV. Conclusion

- GNP with RL is proposed to search for solutions efficiently.
  - Evolution determines structures and change parameters after task execution.
  - RL selects parameters based on rewards obtained during task execution.
  - The proposed method is applied to make khepera robot behavior
    - It can learn wall-following behavior well.
- Future work
  - Apply the proposed method to real world applications (stock prediction)
  - Compare with other evolutionary algorithms