Waseda University Doctoral Dissertation

Study on Robustness and Adaptability of Genetic Network Programming with Reinforcement Learning for Mobile Robot

Siti SENDARI

Graduate School of Information, Production and Systems
Waseda University

February 2013

Abstract

In the real implementation of autonomous mobile robots, the environments change dynamically with unknown and inexperienced situations, which make agents do tasks inappropriately. In order to behave appropriately, the agents should be robust to unknown environments and also have the adaptability mechanisms to change the structures/parameters of their controllers adaptively when inexperienced situations occur.

In the classical methods, designing the controllers of the agents using differential equations or conventional network structures have problems, where those methods are difficult to represent all the actual agent behaviors in the dynamic environments. Generally speaking, although the experiences of an expert are needed to design the controllers, the evolutionary algorithms, such as Genetic Algorithm (GA), Genetic Programming (GP) and Genetic Network Programming (GNP) can generate the optimized programs for robot controllers. In order to get the robust controllers, the agents should be trained with a lot of data and the exploration ability should be improved, furthermore, introducing noises may generate various data which improve the ability of the agents to face uncertainty of the environments. On the other hand, the ability of learning the environments using Reinforcement Learning (RL) improves the adaptability to inexperienced troubles in the implementations.

The objectives of this thesis are to study the robustness and adaptability of Genetic Network Programming with Reinforcement Learning (GNP-RL), so that the autonomous mobile robot behaves appropriately in the unknown environments and determines its behaviors appropriately to adapt to the troubles caused by broken sensors. Here, GNP is one of evolutionary algorithms to automatic program generation, which generates intelligent rules for the agent behaviors. The structures of GNP are constructed and optimized in the evolution, then the experiences of an expert are not needed. GNP has some advantages compared to the other methods which make GNP suitable for implementing the robots, that is, (1) re-usability of the nodes which make the structures more compact and use small memory; and (2) applicability to Partially Observable Markov Decision Process (POMDP). Furthermore, GNP was enhanced to GNP with Reinforcement Learning (GNP-RL), which has the ability to learn the node transitions and change the functions adaptively by selecting the appropriate sub nodes when troubles occur. Therefore, GNP-RL is applicable for dynamic environments and suitable for robot control.

The adaptability of GNP-RL using the Sub node Selection method (SS method) has been analyzed, however, the adaptability mechanism of GNP-RL is not enough when severe troubles occur. In addition, the robustness of GNP-RL in the noisy environments has not been analyzed yet. In order to improve the robustness and adaptability of GNP-RL, this thesis studies 4 topics, that is, (1) Fuzzy GNP with Reinforcement Learning (Fuzzy GNP-RL) which is trained with noises to improve the robust-

ness; (2) Fuzzy GNP with Two-Stage Reinforcement Learning, i.e., Fuzzy GNP-TSRL (BS) which improves the adaptability mechanism by combining the Sub node Selection (SS method) of Fuzzy GNP-RL with Branch connection Selection (BS method), thus, Fuzzy GNP-TSRL (BS) can change functions and determine the appropriate next nodes, while Fuzzy GNP-RL can only determine the appropriate functions; (3) the adaptability of Fuzzy GNP-TSRL (BS) is studied furthermore by changing parameters of ϵ greedy policy and learning rate α in order to balance the exploration and exploitation; and (4) another method to improve the adaptability is Fuzzy GNP with Two-Stage Reinforcement Learning, i.e., Fuzzy GNP-TSRL (CS) which combines the Sub node Selection (SS method) of Fuzzy GNP-RL with Credit branch Selection (CS method) of credit GNP, here, Fuzzy GNP-TSRL (CS) has the ability to skip harmful nodes which cannot be done by Fuzzy GNP-RL and Fuzzy GNP-TSRL (BS).

For evaluating the performances, the wall following problem is used as a benchmark method. The performances are studied in the training phase and implementation phase. From the results, Fuzzy GNP-RL improves the exploration ability of the conventional GNP-RL to determine the node transitions more appropriately. In addition, introducing Gaussian noises to the sensors also improves the exploration ability. As a result, the robustness of Fuzzy GNP-RL in unknown environments is improved using probabilistic node transitions and noises during the training phase. Fuzzy GNP-TSRL (BS) can improve the adaptability of Fuzzy GNP-RL, which can determine the appropriate functions and next nodes. Furthermore, the performance of Fuzzy GNP-TSRL (BS) is improved more by changing parameters of ϵ-greedy policy and learning rate α instead of using fixed parameters, which can balance the exploration and exploitation ability. Fuzzy GNP-TSRL (CS) shows the superior performance comparing with Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL, where Fuzzy GNP-TSRL (CS) can change the functions and skip harmful nodes, while the other methods can not avoid harmful nodes of the node transitions. Therefore, the adaptability of Fuzzy GNP-TSRL (CS) has the best performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Background and Motivation

In the real implementation, the environments of autonomous mobile robots (agents) change dynamically with unknown environments and inexperienced situations, which make the agents do the tasks inappropriately. In the classical method, the controllers of the agents were modeled using Differential Equations (DE), where DE has difficulties when representing the actual behaviors in the dynamically changing environments. These difficulties can be solved easily by using network structures representing artificial knowledge behaviors [1]. Various controllers based on network-structures are increasingly used to solve the problems in the dynamic environments, e.g., Fuzzy Logic Controller (FLC) [2], Fuzzy Cognitive Map (FCM) [3], Neuro Evolution through Augmenting Topologies (NEAT) [5], Evolutionary Acquisition of Neural Topologies (EANT) [6], Genetic Programming (GP) [7], Genetic Network Programming (GNP) [8].

In order to do the tasks successfully in the dynamic environments, the agents should be robust to the uncertain environments and have the adaptability to the changes of the environments caused by inexperienced situations [9–12].

The robustness is defined as the generalization ability of the agents using existing structures to face inexperienced changed of the environments. In order to improve the robustness, the agent can be trained with the following several mechanisms.

1. To use a large number of the training data [13], then the agent can learn various situations, which improves the generalization ability. This mechanism takes a long time to train the agent.

2. To improve the exploration ability [14], which may avoid early premature convergence.

3. To introduce noises when training the systems [15], which implies a large number of training data.

On the other hand, the adaptability of the controller of an agent can be improved by using mechanisms to change the structures/parameters adaptively in inexperienced situations, such as breaking sensors in the implementations. Here, the agent should have learning ability to observe the changes of the environments. Supervised Learning (SL) [16–18] and Unsupervised Learning (UL) [19, 20] were mostly used as the adaptability mechanisms of the systems in the neural network. While SL aims to

learn a mapping from the inputs to outputs whose the correct values are provided by a supervisor, UL aims to find the regularities to map the inputs using a structure, where the appropriate behaviors are decided without using a supervisor. Rather than relying SL using correct and incorrect behaviors, interacting experiences of an agent with its environments are more attracting to infer a strategy for solving the tasks. Reinforcement Learning (RL) is a kind method of UL, which studies the interaction between an agent and its environments based on trial and error, where the objective of the agent is to maximize the reward [21]. Many research show that RL is suitable to learn the control policies for mobile robot navigations [22–29].

While RL improves the performance of the agent using local optimization, Evolutionary Algorithm (EA) [30] is a method to improve the performance of the agents using the global optimization. In EA, the structures and their parameters are represented as the individuals which are optimized by the mechanisms of selection, crossover and mutation.

The integration of EA and RL is studied in many research [31–35], where the individuals use RL to estimate the local rewards from the current generation. The individuals are evaluated for generating the new individuals in the next generation to obtain the global optimization. Thus, the integration of EA and RL improves the performances of the agents to do the task in the dynamically changing environments.

GNP [8] is one of EAs to program generation for efficient agent behaviors, which imitates the human decision making by using if-then rules like GP [7]. While the individuals of GP are represented by tree structures, the individuals of GNP are represented by graph based structures, that is, the individuals of GNP are represented by directed graph structures consisting of three kind nodes, i.e., a start node, judgment nodes and processing nodes, where the nodes are connected to each other by directed links. The behaviors of the agents based on GNP are determined by the node transitions in the structures. The performances of GNP are better compared with GP [8], where GNP has advantages such as, (1) re-usability of the nodes, which makes GNP more compact and (2) applicability to Partially Observable Markov Decision Process (POMDP) [36], while the tree structures of GP can bring bloat problems [37, 38]. GNP was successfully applied to the problems in the dynamically changing environments [39–42].

GNP was extended to GNP with Reinforcement Learning (GNP-RL) which was successfully implemented to navigate the mobile robots [43]. In GNP-RL, the node structures are modified, so that each node has several sub nodes representing the alternative functions. GNP-RL combines evolution and reinforcement learning, where the structures (functions and connections) are constructed by evolution, while the node transitions are learned by RL. This combination has some advantages, that is, speed up the learning process and use small memory which are suitable for the systems like robots. In addition, GNP-RL has improved the robustness and adaptability of GNP, that is, the robustness is achieved by training the individuals with several start positions of the robot, then the generalization ability is improved when the robot is implemented in different environments [43]; the adaptability of GNP-RL has been analyzed [44], i.e., when inexperienced situations occur like sensors' break in the implementation, GNP-RL has mechanisms to adapt to the changes of the environments by adaptively changing the alternative functions.

The features of GNP-RL for mobile robots should be studied furthermore because the environments of the robot are very dynamic, for example noises and inexperi-

enced situations like sensors' sudden break occur in the real implementation. In order to overcome these problems, the robustness and adaptability of GNP-RL need to be improved and studied more in this thesis, because the adaptability mechanism of GNP-RL is not enough when severe troubles occur.

1.2 Objectives and Frameworks

The objectives of this thesis are to study the robustness and adaptability of GNP-RL. Here, two problems of dynamical changing environments are studied, that is, (1) the unknown environments which is determined by using different environments and (2) inexperienced situations which are caused by broken sensors in the implementation.

In order to deal with these problems, how to improve the robustness and adaptability of GNP-RL is proposed as shown in Fig.1.1.

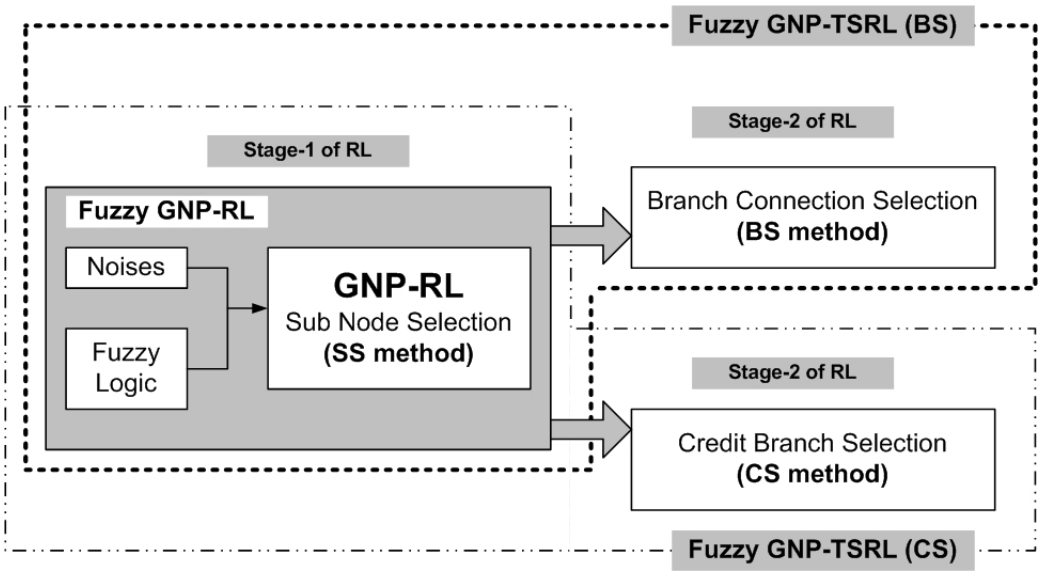


Figure 1.1: Frameworks for improving the robustness and adaptability of GNP-RL

1.2.1 Robustness

The robustness id defined as the generalization ability of the agents using existing structures to face inexperienced changes of the environments.

The robustness of GNP-RL to unknown environments is studied firstly. In order to improve the robustness of GNP-RL to unknown environments, it is proposed to integrate Fuzzy logic to the judgment nodes of GNP-RL, which is named Fuzzy GNP-RL. While the conventional GNP-RL determines the node transitions based on the threshold values, Fuzzy GNP-RL determines the node transitions probabilistically, which overcomes the sharp boundary problems and improves the exploration ability.

In addition, the robustness of Fuzzy GNP-RL is also improved by introducing noises to the sensors during the training phase, where noises can generate more various data for training Fuzzy GNP-RL. Thus, Fuzzy GNP-RL has more generalization ability when implemented in the noisy environments.

1.2.2 Adaptability

The adaptability is defined as the ability of the agents to work adaptively by changing their structures or parameters depending on the changing of the systems.

In order to improve the adaptability to inexperienced situations which are caused by broken sensors in the implementation, Two-Stage Reinforcement Learning methods are proposed. Since the adaptability mechanism of GNP-RL and Fuzzy GNP-RL is the same, that is, One-Stage Reinforcement Learning for Sub node Selection (SS method), the functions are changed adaptively using the SS method when troubles occur. The adaptability mechanisms using Two-Stage Reinforcement Learning is studied in 2 methods, that is, (1) Fuzzy GNP with Two-Stage Reinforcement Learning using Branch connection Selection method, i.e., Fuzzy GNP-TSRL (BS) and (2) Fuzzy GNP with Two-Stage Reinforcement Learning using Credit branch Selection method, i.e., Fuzzy GNP-TSRL (CS).

Fuzzy GNP-TSRL (BS)

In this method, the structures of Fuzzy GNP-RL are enhanced to Fuzzy GNP-TSRL (BS) by combining the SS method and BS method which are learned in the first stage and second stage of RL, respectively. Thus, Fuzzy GNP-TSRL (BS) can change the programs adaptively by changing function and/or connections when troubles occur, which may improve the adaptability in inexperienced situations. Here, Fuzzy GNP-TSRL (BS) uses two kinds of Q-tables, i.e., Q_{SS} table and Q_{BS} table, so that the size of Q tables is increased. Thus, the exploration parameters of ϵ-greedy policy and learning rate α should be considered more carefully in order to balance the exploration and exploitation. Then, Fuzzy GNP-TSRL (BS) is studied furthermore using changing parameters of ϵ-greedy policy and learning rate α, which is compared to that with fixed parameters. The effects of changing parameters are studied in the training phase and implementation phases.

Fuzzy GNP-TSRL (CS)

In this method, the structures of Fuzzy GNP-RL are enhanced to Fuzzy GNP-TSRL (CS) combining the SS method and CS method which are learned in the first stage and second stage of RL, respectively. Different from GNP-TSRL (BS), Fuzzy GNP-TSRL (CS) can change the programs adaptively by skipping the nodes which are considered as harmful nodes.

1.3 Related Works on Fuzzy Logic and Integrating EA with RL

This section introduces the related works of this research.

1.3.1 Fuzzy Logic

Fuzzy logic has capabilities to represent uncertain situations using human-based decision rules named Fuzzy Inference Systems (FIS), which maps inputs $x_1, ..., x_k$ to output y as follows.

$$R_j : \textbf{If } \mathbf{x_1} \textbf{ is } \mathbf{X_1^h} \textbf{ and}... \mathbf{x_i} \textbf{ is } \mathbf{X_i^{h'}} \textbf{ and}... \mathbf{x_k} \textbf{ is } \mathbf{X_k^{h''}}, \textbf{ then y is } \mathbf{a_l} \textbf{ and}... \mathbf{a_m} \textbf{ and}... \mathbf{a_n}, \quad (1.1)$$

where, R_j is j-th rule of the rule base, X_i^h is h-th linguistic value of input i and $h = 1, 2, ..., q$, where q is the number of Fuzzy membership functions. y denotes the output of the system with a_m is possible value for y where $m = 1, ..., n$.

The expert can design as many FIS rules as necessary. However, these rules may be robust for only the specific systems and lack of generalization ability [45]. Then, determining the number of rules and parameters of Fuzzy membership functions for the system is the main problem in designing FIS. EA can overcome this problem, where the number of rules and parameters of the Fuzzy membership functions can be optimized [46]. Furthermore, Fuzzy logic can be integrated to NN structures to enhance the adaptability, where the adaptation involves the tuning of the control rules by changing the control actions and adjusting output gains [28, 32].

Another implementation of Fuzzy systems to deal with uncertainties of the environments is a Probabilistic Fuzzy Logic (PFL) approach, which works in a similar way as regular FIS, but improves the stochastic capability [47–49]. In PFL, Eq. 1.1 is modified with the probability of success as follows.

a_l with the probability of P_{jl},
a_m with the probability of P_{jm},
a_n with the probability of P_{jn}.

1.3.2 Reinforcement Learning (RL) in POMDP

RL is a learning framework which provides adaptation mechanisms in the dynamic environments through trial and error. Instead of using a supervisor, the outputs of the controller are determined using the rewards which are given by the environments depending on the actions taken by the agent. The agent and its environment interact at discrete time steps, $t = 0, 1, 2, ...$ At time step t, the agent receives the information from the environment representing state $s \in S$, where S is the set of states and selects action $a \in A$, where A is the set of actions as depicted in Fig. 1.2. One time step later, the agent receives reward $r_t \in R$ after taking an action and perceives the new state $s' \in S$.

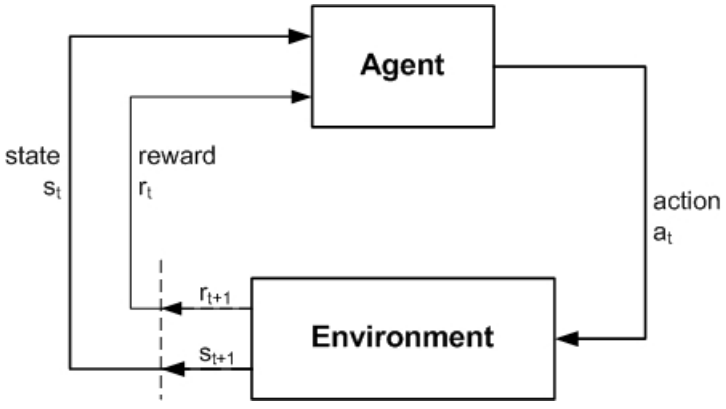


Figure 1.2: Interacting the agent with its environment

The objective of the agent is to maximize the rewards received during the life time by improving a policy to select an action in each state. While the sequence of the

rewards received after time step t are denoted as $r_{t+1}, r_{t+2}, r_{t+3}, ...$, the expected total reward until the end of the life time T, i.e., R_t is defined as a specific function of the reward sequence, that is,

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + ... + r_T. \tag{1.2}$$

In order to make a good estimation of R_t in the dynamic environments, the agent should improve its policy $\pi(s,a)$ which is represented as action-value function ($Q(s,a)$) by selecting action a at state s to maximize R_t as Eq. 1.3,

$$\begin{aligned} Q^{\pi}(s,a) &= E_{\pi}\{R_t | s_t = s, a_t = a\}, \\ &= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}, \end{aligned} \tag{1.3}$$

where, γ is a discount rate ($0 \leq \gamma \leq 1$). If $\gamma = 0$ means the agent maximizes the immediate reward, while γ approaches 1, the agent takes future rewards into account more strongly.

There are two classes of RL for learning the optimal behavior in Markovian domains, that is, Q-learning [50] which is called as off-policy method and Sarsa-learning as on-policy method [51], where the expected values are represented as the Q-value.

- **Q-Learning**. The agent is in state s, it does action a, then reward r is received and the state is transferred into state s'. Here, the actions are selected using a policy according to the Q values, where the Q values are updated by the estimated max Q values. Q-learning is carried out as shown in Fig. 1.3.

Algorithm : Q-Learning

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode)
  Initialize s
  Repeat (for each episode)
    Choose a from s using policy derived from Q (e.g., ϵ-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha(r_t + \gamma max_{a'} Q(s',a') - Q(s,a))$
    $s \leftarrow s'$
  until s is terminal

Figure 1.3: Q-learning algorithm

- **Sarsa Learning**. The agent is in state s, it does action a, then reward r is received and the state is transferred into state s' from which it decides action a'. As on-policy, the agent selects the actions using a policy according to the Q values, while Q values are updated by the estimated Q values. Sarsa learning algorithm is carried out as shown in Fig. 1.4.

Algorithm : Sarsa Learning

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode)
  Initialize s
  Choose a from s using policy derived from Q (e.g.,ϵ-greedy)
  Repeat (for each episode)
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ϵ-greedy)
    $Q(s,a) \leftarrow Q(s,a) + \alpha(r_t + \gamma Q(s',a') - Q(s,a))$
    $s \leftarrow s'; a \leftarrow a'$
  until s is terminal

Figure 1.4: Sarsa learning algorithm

In the RL framework, the agent makes decisions as a function of a signal from the environment called environment's state. In particular, a property of the environments and their state signals should be the Markov property and a RL that satisfies the Markov property is a called Markov Decision Process (MDP). Furthermore, Partially Observable Markov Decission Process (POMDP) provides an essential model for agent operating under uncertainties/incomplete information, [52–54]. A standard POMDP model assumes a discrete state, observation, an action space for the agent, and a reward that the agent gets in each time step. J. Loch [55] confirmed that Sarsa is the best policy for estimating the reward in POMDP.

1.3.3 Integrating EA with RL

EA is an optimization technique to create potential solutions using genetic operators, i.e., selection, crossover and mutation, where the potential solutions are selected on the basis of their quality/fitness. EA is used in various tasks, such as connection weight training, architecture design, learning rule adaptation, input feature selection, rule extraction, etc., where the evolution process of EA is shown in Fig. 1.5.

Algorithm : EA-RL

1. Generate the initial population $G(0)$ at random and set $i = 0$;
2. Repeat
   (a) Evaluate each individual in the population;
   (b) Select parents from $G(i)$ based on their fitness in $G(i)$;
   (c) Apply genetic operators (crossover and mutation) to parents and produce offspring which form $G(i+1)$;
   (d) $i = i + 1$;
3. Until 'terminal criterion' is satisfied;

Figure 1.5: EARL algorithm

Recently, the researchers are attracted to integrate EA with RL (EA-RL) to improve

the adaptability of the agent in the dynamically changing environments.

The integration of GA with RL (GA-RL) is widely used to optimize the architectures of ANN, which is called Evolutionary Artificial Neural Network (EANN) [56–58] and can also be used to optimized Fuzzy Inference System (FIS) [28, 60]. This integration shows the effectiveness to optimize the structural and parametric characteristics without any priory knowledge on the system.

Another EA method which can directly generate programs to dynamic problems is GP. There are two different methods to integrate GP with RL (GP-RL), that is, (1) the individuals are optimized in the evolution phase, but RL is used outside of the evolution process, that is, for the implementation with continuous states and actions [33], (2) the other method uses RL inside of the evolution process with discrete states and actions [34]. The second method can speed up the learning process of RL for learning more different situations and is widely used in the training phase as shown in Fig. 1.6.

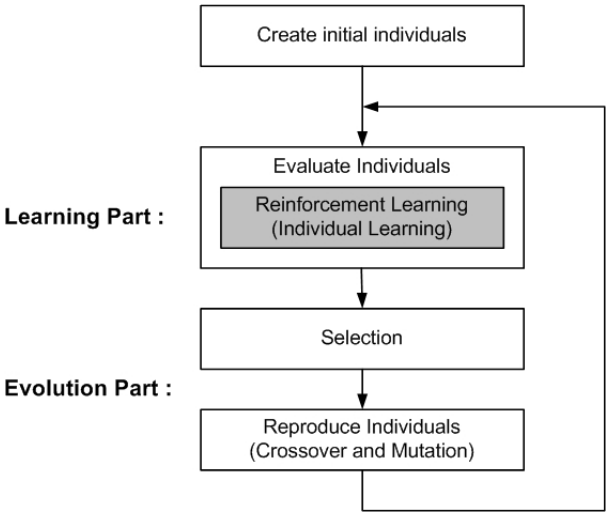


Figure 1.6: Process of integrating EA with RL

In the first time step of EA-RL, the individuals of a population should be created. The representation of the individuals can be changed depending on the tasks. The individuals of GA are represented as numerical strings of the weights of ANN, as used by Fuzzy Q-Learning Genetic Algorithm (FQLGA) [60], Neuro Evolution of Augmenting Topologies (NEAT) [5] and GeNeralized Acquisition of Re-current Links (GNARL) [61]. On the other hand, the individuals of GP are represented by tree structures which consist of function nodes and terminal nodes [7].

The training phase of EA-RL can be grouped into two parts, that is, learning part and evolution part.

Learning part

In papers [28, 33, 34, 56–60], the problems have the Markov properties and Q tables are used to estimate the state-action pairs, where the optimization is divided into two methods, that is, (1) partitioning of the environments and (2) partitioning the structures of the agents.

The optimization of EA-RL using partitioning of the environment is shown in Fig. 1.7, where the positions in the environment represent the states and the agent behaviors to move in the environment represent the actions. The generalization ability of this method should be confirmed, because different environments need different representation of Q tables. Furthermore, the Q tables also should be large when the environment is more complex, then the Q table take a long time to converge.

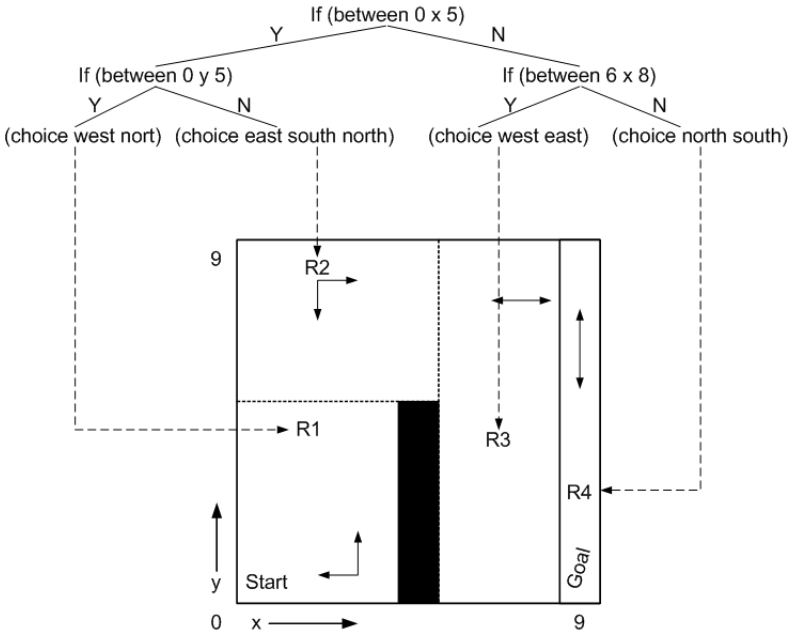


Figure 1.7: An example of optimization using GP for partitioning the environment (K. L. Downing [59])

The other method is the optimization of EA-RL using partitioning the structures of the agents. While the structure of the agent is NN, then the nodes represent the states while the weights or parameters represent the actions [28, 60]. On the other hand, while the structure of the agent is a tree structure of GP, the terminal nodes represent the states which consist of several variables representing the actions [34] as shown in Fig. 1.8. Thus, using this method, the Q table can converge faster and the agent has more generalization ability for implementation in unknown environments.

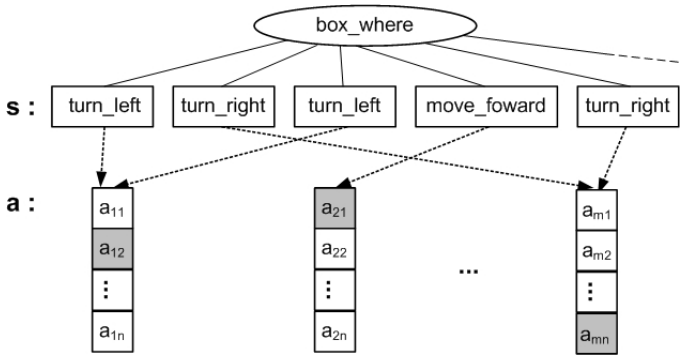


Figure 1.8: An example of optimization of the structure of GP (S. Kamio [34])

Evolution part

While RL works on each individual to make local optimization, the evolution process works on individuals in a population to make the global optimization using selection, crossover and mutation operators.

The selection method is used to select good individuals in the population as parents. The selection methods which are commonly used are *roulette wheel selection* and *tournament selection*. In the roulette wheel selection method, each individual has a chance to become a parent using the proportion to its fitness, where the individuals with larger fitness (slot sizes) have more chances to be selected. On the other hand, the tournament selection method selects a number of the individuals randomly, where the individual with the highest fitness becomes the parent. J. Zhong [62] found that the tournament selection method works more efficiently than the roulette wheel selection method to encounter the problems of prematurity when the size of the population is large enough.

Crossover combines the structures of two individuals to generate two offspring and serves as an accelerator to generate behaviors. On the other hand, mutation works on an individual to generate an offspring and serves to create random diversity in the population. The example of crossover operation is shown in Fig. 1.9 and mutation operation is shown in Fig. 1.10.

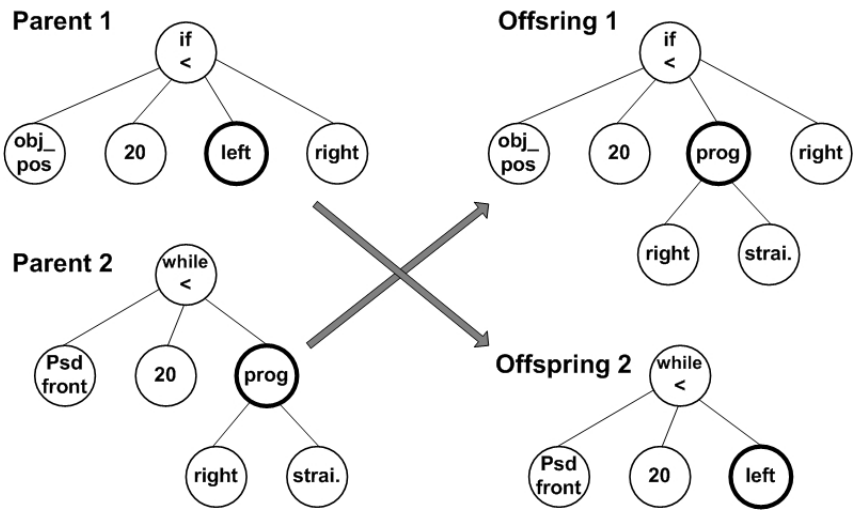


Figure 1.9: An example of crossover operation on GP (T. Braunl [66])

Crossover probability, i.e., Pc and mutation probability, i.e., Pm are known to critically affect the behavior and performance of EA. Instead of using the uniform mutation and crossover, the adaptive probabilities of crossover and mutation is proposed to improve the performance of EA [63]. On the other hand, when the individuals have numerical parameters, the nonuniform mutation [64] shows better performance than the adaptive mutation [65]. Thus, the nonuniform mutation [64] is used in this research.

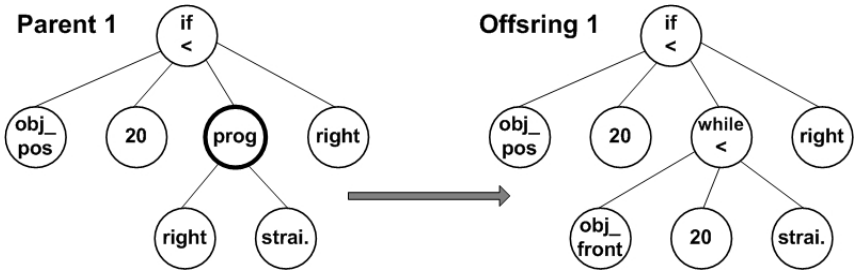


Figure 1.10: An example of mutation operation on GP (T. Braunl [66])

1.3.4 Genetic Network Programming

Standard Genetic Network Programming (Standard GNP) has been proposed [8], which has the ability to generate problem solving programs. The individuals of standard GNP are represented by directed graph structures as shown in Fig. 1.11, which consist of a start node and a fix number of the judgment nodes and processing nodes.

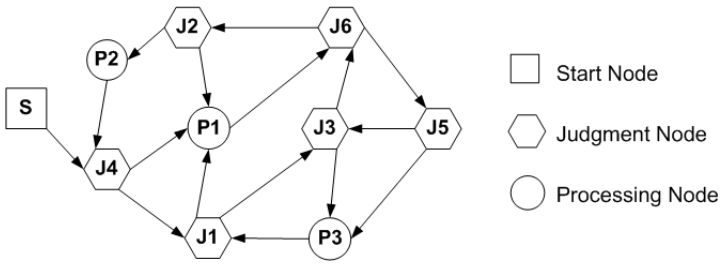


Figure 1.11: Basic Structure of GNP

Standard GNP has been extended as follows.

1. Standard GNP was integrated with RL which improves the adaptability in the dynamic environments, that is, GNP-RL [35]. GNP-RL has several sub nodes in each of the judgment nodes and processing nodes representing the alternative functions which are learned using Sarsa learning. The agent behaviors are determined directly using node transitions.

2. Standard GNP has been extended to Fuzzy GNP [67, 68], that is, Fuzzy logic is integrated to the judgment nodes to deal with the continuous values. The Fuzzy parameters are evolved using non uniform mutation only, but this method lacks of the global optimization, because the individuals use the same Fuzzy parameters. Fuzzy GNP has been used to generate rules and they are stored in the rule pool during the training phase to be used in the implementation.

3. Standard GNP has been extended to Credit-GNP [69], where the credit branch is implemented to GNP with rules [70]. Here, Credit-GNP with rules has the ability to skip harmful nodes during the training phase to be used in the implementation.

This research is enhancing of GNP-RL which can determine the agent behaviors directly using node transitions, therefore, this method uses small memory. In addition,

the programs of GNP-RL have the ability to change adaptively when the environments are changed. Thus, GNP-RL is suitable for the implementation in the dynamic environments, such as mobile robot applications.

1.3.5 GNP as a Mobile Robot Controller

A mobile robot is an automatic machine which has capabilities to move in the environments. In this research a wheeled robot is used to study the proposed method, i.e., a Khepera robot. In the evolutionary robots, many individuals should be trained and evaluated in the training phase, which are difficult to use the real robot, then, the robots are trained using robot simulator. Webots [71] is a robot simulator, which is widely used in the field of autonomous systems, intelligent robotics and evolutionary robotics [72].

Actually, the environments in the real implementation are different from the those in the training environments, where inexperienced changes of the environments are included. Then, the robot should have the robustness and adaptability to these changes.

A Khepera robot has characteristics as follows.

- 8 infra red distance sensors, which return values from 0 to 1023, that is, when the sensor touches the wall, then it returns 1023, while far from the wall it returns 0.
- 2 differential wheels (the right and left wheels, i.e., V_R and V_L, respectively), where the speed of the wheel can be set from -10 to 10.

The behaviors of the robot can be determined as shown in Table. 1.1. If the speed of the wheels, that is, the right and left wheels have the same and positive values, then, the robot moves forward, the robot moves backward while the values are negative. On the other hand, while the speed of the right wheel (V_R) is larger than that of the left wheel (V_L), then, the robot turns left, vice versa.

Table 1.1: Behaviors of the robot

	Behavior
$V_R = V_L(+)$	move forward
$V_R = V_L(-)$	move backward
$V_R > V_L$	turn left
$V_R < V_L(+)$	turn right

GNP is an automatic program generation for efficient agent behaviors, which is suitable as robot controllers. GNP as a robot controller is shown in Fig. 1.12, which is represented by a directed graph structure consisting of

- a start node, which determines the first node to be executed,
- judgment nodes, which judge the assigned inputs and return the judgment results to determine the next nodes to be executed,
- processing nodes, which set the speed of the wheels.

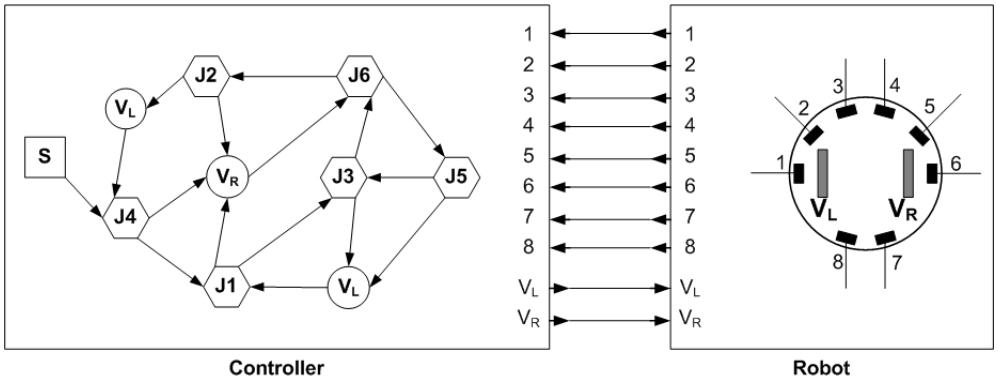


Figure 1.12: GNP as a robot controller

The nodes are connected each other by directed links, where the connections, functions and parameters of the nodes are changed in the evolution part.

The behaviors of the agents are determined by the node transitions of GNP. Here, the unique definitions are used in the case of GNP, as follows.

- **Node Transitions** mean the sequences of executing the judgment nodes and processing nodes, for example : $J4 \rightarrow J1 \rightarrow J3 \rightarrow VL$.
- **1 step** is defined as the least delay time spent by the node transitions, for example: 8 time units.
- **time unit** means the delay time for executing a node, for example: a judgment node spends 1 time unit and a processing spends 5 time units.

If the node transitions are $J4(1) \rightarrow J1(1) \rightarrow J3(1) \rightarrow VL(5)$. Then, the node transitions spend 8 time units, which mean that 1 step is obtained.

1.4 Contents of this Research

This thesis includes four topics to be studied based on the objectives as follows.

In Chapter 2, Fuzzy Genetic Network Programming with Reinforcement Learning (Fuzzy GNP-RL) has been proposed. The structures of Fuzzy GNP-RL are based on GNP-RL, where Fuzzy Logic is integrated to the judgment nodes, then the node transitions of Fuzzy GNP-RL can be determined probabilistically. The proposed method is simulated using Webots simulator [71] and evaluated for the wall following behaviors of a Khepera robot. The robustness of Fuzzy GNP-RL is carried out by using different training and implementation environments. As a result, Fuzzy GNP-RL improves the robustness in unknown environments. Furthermore, the robustness of Fuzzy GNP-RL in the dynamic environments is also studied by introducing Gaussian Noises during the training phase. The results show that Fuzzy GNP-RL with noises improve the robustness in inexperienced environments. Based on the results, the Fuzzy judgment nodes will be used for the next methods in Chapter 3, 4 and 5 and the Fuzzy judgment nodes are named as the judgment nodes for simplicity.

In Chapter 3, Fuzzy Genetic Network Programming with Two-Stage Reinforcement Learning using Branch Selection Method, i.e., Fuzzy GNP-TSRL (BS) has been

proposed. Fuzzy GNP-TSRL (BS) combines the Sub node Selection method (SS method) like Fuzzy GNP-RL and Branch connection Selection method (BS method). The SS method and BS method are learned in the first stage and in the second stage of RL, respectively. In Fuzzy GNP-TSRL (BS), the actions are divided into two groups using two kinds Q tables, that is, Q_{SS} table and Q_{BS} table. Fuzzy GNP-TSRL (BS) enhances the adaptability mechanism of Fuzzy GNP-RL, which can adaptively change the programs by selecting not only the appropriate functions, but also the appropriate next node connections. The adaptability Fuzzy GNP-TSRL (BS) is evaluated when inexperienced troubles occur by making several sensors break in the implementation. The adaptability of Fuzzy GNP-TSRL (BS) which uses Two-Stage Reinforcement Learning is compared with that of Fuzzy GNP-RL which uses One-Stage Reinforcement Learning. The results show that the adaptability mechanism of Fuzzy GNP-TSRL (BS) works effectively and efficiently so the performance of Fuzzy GNP-TSRL (BS) is better than Fuzzy GNP-RL.

In Chapter 4, Changing ϵ-greedy and learning rate α for improving the performance of Fuzzy GNP-TSRL (BS) is studied. ϵ-greedy and learning rate α are set at a larger value in early generations and decreased gradually in latter generations. Thus, a larger exploration is carried out in early generations, however, a larger exploitation is carried out in last generations. Changing ϵ-greedy and learning rate α is also studied in the implementation phase, that is, when sudden changes occur, the exploration is increased and gradually decreased until the end of the life time. The performance of Fuzzy GNP-TSRL (BS) with changing parameters of ϵ-greedy and learning rate α is studied by comparing that with fixed parameters of ϵ-greedy and learning rate α. As a result, changing ϵ-greedy and learning rate α can improve the performance of Fuzzy GNP-TSRL (BS), which means that the balance of the exploration and exploitation can be controlled efficiently and effectively.

In Chapter 5, another method of Two-Stage Reinforcement Learning based on GNP is studied, that is, Fuzzy Genetic Network Programming with Two-Stage Reinforcement Learning using Credit branch Selection method (Fuzzy GNP-TSRL (CS)). The proposed method combines the Sub node Selection method (SS method) of Fuzzy GNP-RL and Credit branch Selection method (CS method). While Fuzzy GNP-TSRL (BS) provides the alternative functions and alternative connections, Fuzzy GNP-TSRL (CS) provides the alternative functions as Fuzzy GNP-RL, but this method can skip harmful nodes. The results show that Fuzzy GNP-TSRL (CS) is superior than Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL, because skipping harmful nodes, which cannot be done by Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL may recover the troubles more quickly. Thus, Fuzzy GNP-TSRL (CS) determines node transitions more efficiently and effectively, as a result the adaptability can be improved.

Chapter 2

Fuzzy GNP-RL

2.1 Introduction

The integration of Fuzzy logic to standard GNP (Fuzzy GNP) has been introduced in [67,68], where the Fuzzy membership functions are used for extracting class association rules. In these methods, each individual in the population uses the same Fuzzy parameters which are mutated generation by generation. Although these methods can improve the performance of standard GNP, but the Fuzzy parameters lack of the global optimization.

Based on these studies, the proposed method integrates Fuzzy logic to GNP-RL, which is named Fuzzy GNP-RL. Like the conventional Fuzzy GNP, Fuzzy GNP-RL also integrates Fuzzy logic to the judgment nodes, then the node transitions of Fuzzy GNP-RL can be determined probabilistically. However, the different points between Fuzzy GNP-RL with the conventional Fuzzy GNP [67,68] are

- each node of each individuals has several sub nodes which are characterized by their own Fuzzy parameters,
- the Fuzzy parameters are evolved generation by generation by using selection, crossover and mutation operations to get the global optimal Fuzzy parameters,
- Sarsa learning is implemented to learn the optimal Fuzzy parameters of the Sub node Selection method (SS method),
- the agent behaviors are determined directly by using the probabilistic node transitions based on the Fuzzy membership values.

2.2 Motivation of Fuzzy GNP-RL

The idea of integrating Fuzzy logic to GNP-RL is to deal with the continuous values of sensors. While the conventional non-Fuzzy GNP-RL divides these values into two categories, that is, Yes/No using the threshold values, Fuzzy GNP-RL divides the sensor values into Yes/No category based on the degrees of truth using Fuzzy membership functions, which means the probabilistic node transitions. In addition, the idea of Fuzzy GNP-RL is to explore the structures more appropriately when the values of sensors are near to the threshold values, then Fuzzy GNP-RL improves the robustness of non-Fuzzy GNP-RL in the dynamic environments. The judgment nodes

of the proposed method are characterized by the Fuzzy membership functions and the node transitions are determined probabilistically.

Many applications of Fuzzy logic for mobile robot navigations were studied to provide more precise distance information in uncertain environments by reducing the effects of noises [9, 12, 18]. In Fuzzy GNP-RL, Gaussian noises are introduced to the structures in order to improve the robustness. As a result, Fuzzy GNP-RL with noises is robust in unknown environments.

The objective of this chapter is to study the robustness of Fuzzy GNP-RL compared with non-Fuzzy GNP-RL in uncertain environments. The effects of introducing noises for improving the robustness are also studied in the training phase and implementation phase. These methods are evaluated using the wall following behaviors of a Khepera robot.

2.3 Algorithm of Fuzzy GNP-RL

The new point of Fuzzy GNP-RL is to determine the node transition probabilistically. In order to realize the proposed method, Fuzzy logic is integrated to the judgment nodes, while the start node and processing nodes works like the conventional GNP-RL. The node structures of Fuzzy GNP-RL are described in subsection 2.3.1, where the probabilistic node transition is described in subsection 2.3.2. In addition, the evolution phase is explained in subsection 2.3.3. In subsection 2.3.4, introducing noises to Fuzzy GNP-RL is described.

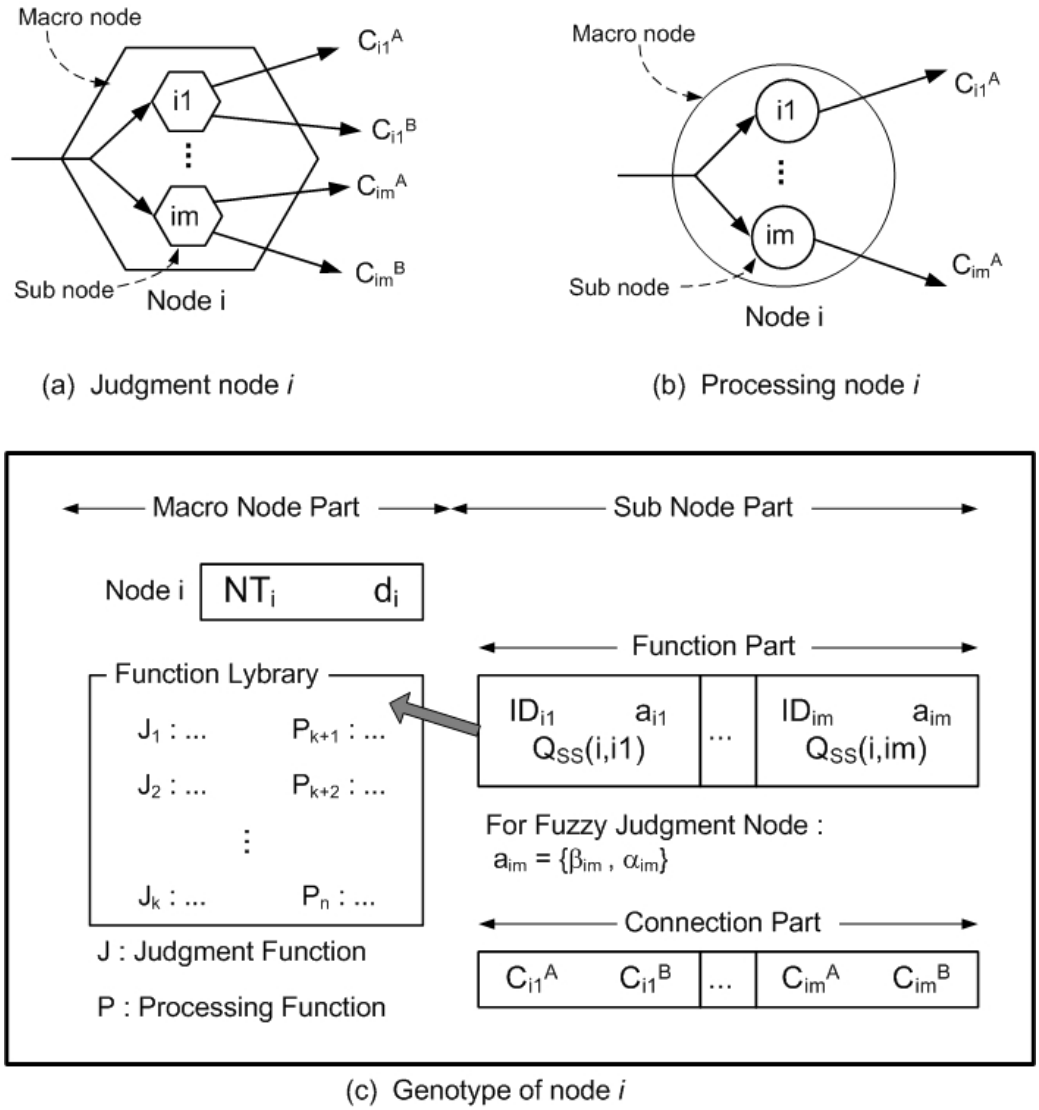
2.3.1 Structure Representation

The individual of Fuzzy GNP-RL is represented as a directed graph structure like the standard GNP as shown in Fig. 1.11, which consists of 3 kind nodes, that is, start node, judgment node and processing node. These nodes have the following functions.

- The start node has no function and its only role is to determine the first node to be executed.
- The judgment node has if-then type conditional branch decision functions to judge the assigned inputs from the environments. The decision functions are carried out probabilistically to determine the next node connected.
- The processing node has no if-then type conditional branch decision functions and its function is to determine the agent behaviors.

The structure of Fuzzy GNP-RL has n nodes consisting of a start node and a fix number of the judgment nodes and processing nodes.

In order to enhance the ability of the conventional GNP-RL in the dynamic environments, each node of Fuzzy GNP-RL also has several sub nodes representing the alternative functions which are evolved and learned like the conventional GNP-RL. It is supposed that node $i \in \{0, 1, ..., n-1\}$ has m sub nodes, where the structure of node i and its gene structure are shown in Fig. 2.1. The gene structure is divided into macro node part and sub node part.



(a) Judgment node *i*

(b) Processing node *i*

(c) Genotype of node *i*

Figure 2.1: Phenotype and genotype of Fuzzy GNP-RL

Macro node part

The macro node part of node i is defined by NT_i and d_i. Where NT_i represents a node type and d_i represents the time delay spent on executing node i, where NT_i and d_i are shown in Table 2.1.

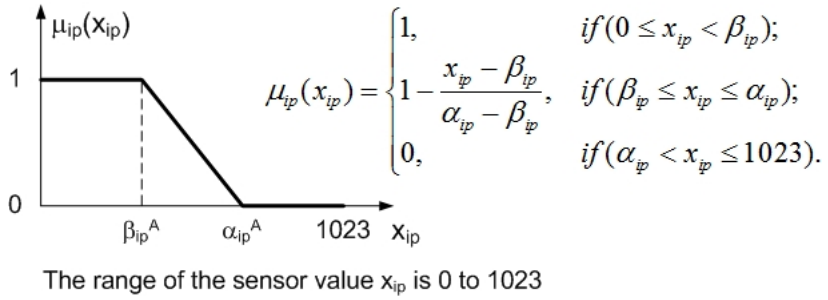
Table 2.1: NT_i and d_i of node i

Node	NT_i	d_i (time unit)
Start node	0	0
Judgment node	1	1
Processing node	2	5

Sub node part

The sub node part is composed of two parts, i.e., function part and connection part. The function of sub node $p \in \{1, ..., m\}$ of node i is defined by ID_{ip}, a_{ip} and $Q_{SS}(i, ip)$ as follows.

- ID_{ip} is a code number of the judgment/processing sub node p of node i encoded by a unique number shown in the *function library*.
- a_{ip} is a parameter of the judgment/processing sub node p of node i.
  When the node is a judgment node, $a_{ip} = \{\beta_{ip}, \alpha_{ip}\}$ is used to determine the parameter of a Fuzzy membership function as shown in Fig. 2.2.
  On the other hand, when the node is a processing node, a_{ip} is used to determine the speed of the wheel of a Khepera robot.



$$\mu_{ip}(x_{ip}) = \begin{cases} 1, & if(0 \le x_{ip} < \beta_{ip}); \\ 1 - \dfrac{x_{ip} - \beta_{ip}}{\alpha_{ip} - \beta_{ip}}, & if(\beta_{ip} \le x_{ip} \le \alpha_{ip}); \\ 0, & if(\alpha_{ip} < x_{ip} \le 1023). \end{cases}$$

Figure 2.2: Fuzzy membership function of Fuzzy judgment node

- $Q_{SS}(i, ip)$ means the Q value for the SS method which is assigned to each pair of state s and action a, here, s is the current node i and a is the selection of sub node ip.

The connection part of sub node p of node i shows the next node connected from this sub node. If node i is a judgment node, each sub node has 2 connections, i.e., C^A_{ip} and C^B_{ip}, here, A and B correspond to the judgment results of Yes/No. On the other hand, the processing nodes have no conditional branch, then the connection part is defined by C^A_{ip} only.

2.3.2 Probabilistic Node Transition

The node transitions are sequences of the judgment nodes and processing nodes, which determine the agent behaviors. The example of the node transition is shown in Fig. 2.3, that is, $P1 \rightarrow J6 \rightarrow J5 \rightarrow P3$. At first, the agent behavior is determined by executing $P1$, after executing rules "$if\,(J6 \wedge J5),\ then\ P3$", the behavior will be changed by executing processing node $P3$. When the current node is a judgment node, the next node is determined probabilistically, as follows.

1. Sensor value x_{ip} is mapped into the Fuzzy values using the Fuzzy membership functions as shown in Fig. 2.2.

2. The judgment nodes have 2 branches, i.e., A and B. The next node of the judgment node is determined probabilistically using Eq. 2.1 and Eq. 2.2,

$$P_{ip}^{A} = \mu_{ip}(x_{ip}), \tag{2.1}$$

$$P_{ip}^{B} = 1 - P_{ip}^{A}. \tag{2.2}$$

   (a) The next node shown by C_{ip}^{A} is selected, i.e., where branch A of sub node p of node i is selected by the probability of P_{ip}^{A}.

   (b) The next node shown by C_{ip}^{B} is selected, i.e., where branch B of sub node p of node i is selected by the probability of P_{ip}^{B}.

3. In the case of the processing node, the next node is always C_{ip}^{A}.

Instead of using the threshold values, branch A and branch B can be determined using the probabilistic node transitions, where input x_{ip} of sub node p of judgment node i judges has a value between β_{ip} and α_{ip}. Thus, the exploration ability of the judgment nodes can be improved.

The learning process of Fuzzy GNP-RL using Sarsa learning is explained as shown in Fig. 2.4.

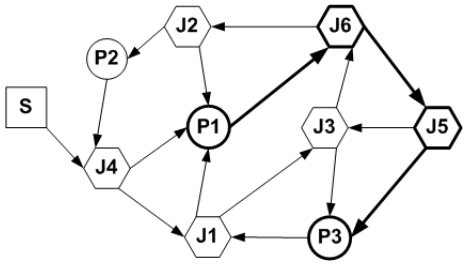


Figure 2.3: An example of node transition of GNP

1. At time step t, the current node is node i and the number of sub node is m. Referring to all Q values of the sub nodes, i.e., $Q_{SS}(i, i1), ..., Q_{SS}(i, im)$, one sub node is selected based on ϵ-greedy policy. The sub node which has maximum Q value is

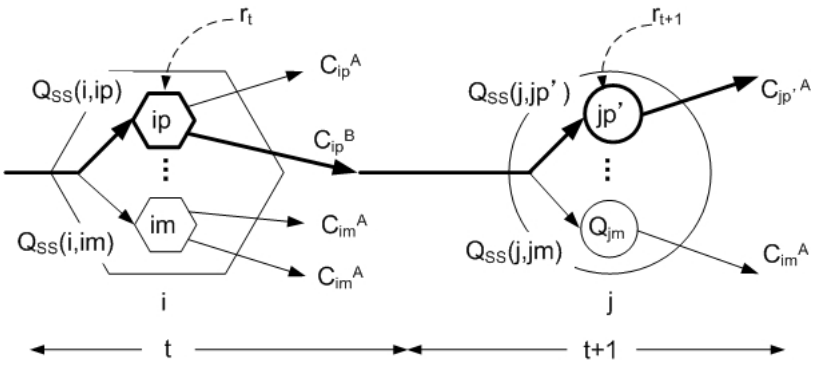


Figure 2.4: Sarsa learning of Fuzzy GNP-RL

selected by the probability of $1-\epsilon$, or random one is selected by the probability of ϵ. When sub node p of judgment node i is selected, the corresponding node function becomes ID_{ip}, $a_{ip} = \{\beta_{ip}, \alpha_{ip}\}$ and $Q_{SS}(i, ip)$.

2. The corresponding node function at time step t is executed, then the next node is determined probabilistically as explained above.

3. After executing the node function, reward r_t is given.

4. At time $t+1$, the current node i is transferred to the next node j selected at time step $t+1$. Then, one sub node is selected using the same way in step 1. It is supposed $Q_{SS}(j, jp')$ is selected.

5. Then, $Q_{SS}(i, ip)$ is updated by

$$Q_{SS}(i, ip) \leftarrow Q_{SS}(i, ip) + \alpha(r_t + \gamma Q_{SS}(j, jp') - Q_{SS}(i, ip)), \tag{2.3}$$

where, α is learning rate $0 < \alpha \leq 1$ and γ is discount rate $0 \leq \gamma \leq 1$.

6. $i \leftarrow j, p \leftarrow p'$ and $t \leftarrow t+1$, then return to step 2.

2.3.3 Genetic Operator

The evolution process of Fuzzy GNP-RL starts from the initialization of individuals. Using the tournament selection, good parents are selected and evolved using crossover and mutation generation by generation. This process is done as [35]. The Flowchart of Fuzzy GNP-RL is shown in Fig. 2.5.

Initialization of individuals

Each individual is initialized as follows.

- Function ID_{ip} is assigned by a unique number which is shown in the *function library*.

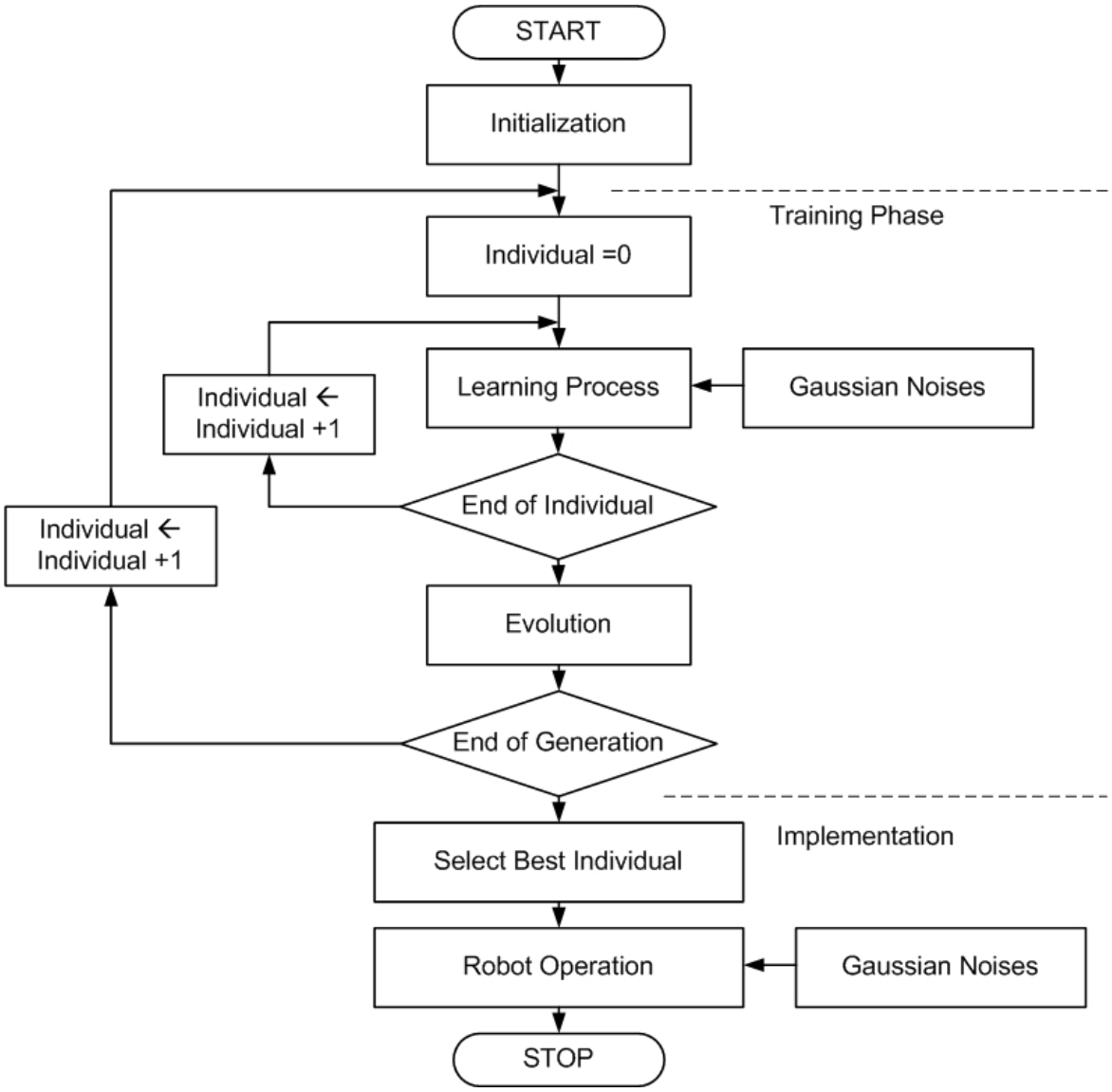


Figure 2.5: Flowchart of Fuzzy GNP-RL

- Parameter a_{ip} is set at a randomly selected integer. When the node is a processing node, its parameter is set between -10 and 10, While the node is a judgment node, its parameter is $a_{ip} = \{\beta_{ip}, \alpha_{ip}\}$, where α_{ip} should be larger than β_{ip} , that is, α_{ip} is set between 0 and 1023, while β_{ip} is set between 0 and α_{ip}.
- $Q_{SS}(i, ip)$ value is set at 0.
- The connection nodes of C^A_{ip} and C^B_{ip} of sub node p of node i are determined randomly in the graph structure.

Crossover and Mutation

Crossover and mutation are carried as the conventional GNP-RL, as follows.

Crossover. Crossover is executed between two parents, and two offspring are generated. The crossover process is described as follows.

1. Two parents are selected by the tournament selection.
2. Node i is selected as a crossover node with the probability of P_c.
3. Two parents exchange the genes of the corresponding crossover node (i.e., with the same node number).
4. Generated new individuals become the new ones in the next generation.

Mutation. Mutation is executed in one individual, and a new individual is generated. The mutation process is described as follows.

1. One individual is selected by the tournament selection.
2. In Fuzzy GNP, three mutations are used.
   (a) Connection. Each node branch is re-connected to another node with the probability of P_m.
   (b) Function. Each node function is changed to another one with the probability of P_m. For example, in the Fuzzy judgment node, $ID_{ip} = 1$ which corresponds to sensor number *1* is changed to $ID_{ip} = 3$ which corresponds to sensor number 3, or in the processing node, $ID_{ip} = 0$ which corresponds to the right wheel is changed to $ID_{ip} = 1$ which corresponds to the left wheel.
   (c) Parameter. The mutation Fuzzy parameters are explained next.
3. Generated new individual becomes the new one in the next generation.

Mutation of Fuzzy Parameters

In order to deal with continuous values, mutation of parameters of Fuzzy GNP-RL is done as follows.

- Parameter a_{ip} of the processing node is changed randomly selected integer between -10 and 10.

- Parameter $a_{ip} = \{\beta_{ip}, \alpha_{ip}\}$ of the judgment node is changed using *non uniform mutation* [64], which makes *long jump* mutation at early generations and *fine tuning* at later generations, as shown Eq. 2.4.

$$a'_{ip} = \begin{cases} a_{ip} + \Delta(g, UB - a_{ip}) & \text{if } \xi = 0, \\ a_{ip} - \Delta(g, a_{ip} - LB) & \text{if } \xi = 1, \end{cases} \tag{2.4}$$

where, $a'_{ip} = \{\beta'_{ip}, \alpha'_{ip}\}$ is a parameter of Fuzzy GNP-RL after mutation, ξ is a random digit, LB is lower bound (0) and UB is upper bound (1023) in the case of α_{ip}, while $UB = \alpha_{ip}$ in the case of β_{ip}. $\Delta(t, y)$ is described in the following

$$\Delta(g, y) = y\left(1 - r^{\left(1 - \frac{g}{G}\right)^b}\right), \tag{2.5}$$

where, g is the current generation number, G is the total generation number and r is a random number between 0 and 1. b is a system parameter which determines the degree of dependency on the iteration number (Here, b = 2 is used).

2.3.4 Introducing Noises

Generally, the robustness could be done by (1) increasing the number of training data, (2) reducing the redundant structures or (3) adding noises in the training phase, where the last one is a simple way. In this chapter, the robustness of Fuzzy GNP-RL is improved by introducing Gaussian noises in the training phase.

Introducing noises for Fuzzy GNP-RL is carried out as follows.

- In the training phase, random Gaussian noises are added to the sensor values. It is supposed that the current node i is a judgment node, and sub node p is selected. When noise x_n is added to sensor value x_{ip}, then the sensor value is changed to x'_{ip} by Eq. 2.6. In this case, the judgment nodes learn incorrect information.

$$x'_{ip} = x_{ip} + x_n. \tag{2.6}$$

- Sensor value x'_{ip} is mapped into the Fuzzy values using the Fuzzy membership functions in Fig. 2.2, and these Fuzzy values are used to determine the probability of selecting C^A_{ip} and C^B_{ip} using Eq. 2.1 and Eq. 2.2. Because the sensor values change dynamically, then the probability of selecting the connections can be changed. As a result, flexible variation of node transitions are learned for improving the generalization ability in the implementation phase.

2.4 Comparison between Fuzzy and Non-Fuzzy GNP-RL

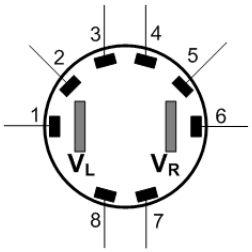
The difference between Fuzzy GNP-RL and non-Fuzzy GNP-RL has been explained in section 2.1, that is, the node transitions are determined probabilistically. While the input values of non-Fuzzy GNP-RL are divided into crisp values based on the threshold values to determine the connections, those of Fuzzy GNP-RL are mapped into the Fuzzy values to determine the probability of selecting the connections.

2.5 Simulations

The performance of Fuzzy GNP-RL is studied in the benchmark of the wall following behaviors for a Khepera robot using Webots simulator [71].

2.5.1 Khepera Robot

The simulated Khepera robot is shown in Fig.2.6. It has eight infrared distance sensors which are used to perceive objects in front of it, behind of it, to the right and left of it by its reflection. Each sensor returns a value ranging between zero and 1023. Zero means that no object is perceived, while 1023 means that an object is very close to the sensor (almost touching the sensor). Intermediate values may give an approximate idea of the distance between the sensor and object. Two motors turn the right and left wheels of the robot, respectively. The speeds of the right wheel, i.e., v_R and the left wheel, i.e., v_L are between -10 to +10. Negative values rotate the wheel backward, while positive values rotate the wheel forward.



V_L	V_R	Behavior
+	+	Move foward
+	-	Turn right
-	+	Turn left
-	-	Move backward

Figure 2.6: Khepera robot

2.5.2 Reward and Fitness in the Wall Following Behavior

In each time step, the judgment nodes judge the values of the sensors and the processing nodes determine the speed of the wheels according to node function ID_{ip} and parameter a_{ip}, while the robot moves in the environment and gets rewards. Each individual works during the life time T, then the fitness is calculated. In this simulation, Fuzzy GNP-RL learns the wall following behaviors, i.e., the robot must move along the wall as fast as and as straight as possible. Reward r_t at time step t and fitness is calculated by

$$r_t = \frac{v_R(t) + v_L(t)}{20} \times \left(1 - \sqrt{\frac{|v_R(t) - v_L(t)|}{20}}\right) \times C, \tag{2.7}$$

$$Fitness = \sum_{t=1}^{T} r_t / T. \tag{2.8}$$

If all the sensors have the values less than 1000 and at least one of them is more than 100, then C is equal to 1, otherwise C is equal to 0.

2.5.3 Simulation Environments

The performance of Fuzzy GNP-RL is evaluated using the following 4 simulation conditions comparing with non-Fuzzy GNP-RL, as follows.

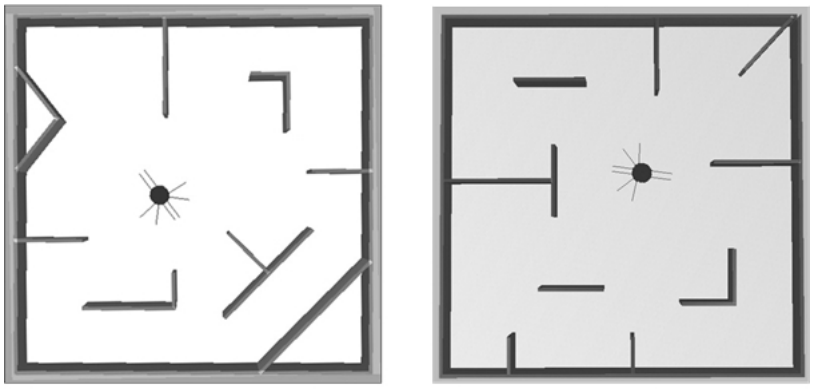


Figure 2.7: Training and implementation environments

1. In simulation 1, Fuzzy GNP-RL and non-Fuzzy GNP-RL are trained without Gaussian noises (WN_{Train}), where the individuals are trained for the wall following behaviors for a Khepera robot in the environment shown in Fig 2.7(a). Each method is simulated 10 times to study the generalization ability in the training phase. In each simulation, the start positions of robot are determined randomly from 10 different start positions.

2. Simulation 2 is done like simulation 1, but Gaussian noises are introduced during the training phase. Various Gaussian noises are used as shown in Table 2.2. Mean μ of noises is usually taken to be 0, but standard deviation σ is set at various values, that is, $\sigma = 50$ (named $N_{Train}50$), $\sigma = 100$ ($N_{Train}100$), $\sigma = 150$ ($N_{Train}150$) and $\sigma = 350$ ($N_{Train}350$).

Table 2.2: Noises for Fuzzy GNP-RL during the training and implementation phase

Phase	Symbol	μ	σ
Training	$N_{Train}50$	0	50
	$N_{Train}100$	0	100
	$N_{Train}150$	0	150
	$N_{Train}350$	0	350
Implementation	$N_{Impl.}50$	0	50
	$N_{Impl.}100$	0	100
	$N_{Impl.}150$	0	150
	$N_{Impl.}350$	0	350

3. In simulation 3, the best individuals from 10 independent training simulations of simulation 1 are selected. The individuals trained without introducing Gaussian

noises WN_{Train} are implemented in the different implementation environment as shown in Fig 2.7(b). The implementation simulations are carried out in two different cases,

(a) **Case 1**, i.e., the implementation without introducing Gaussian noises $WN_{Impl.}$. The generalization ability of Fuzzy GNP-RL is compared with non-Fuzzy GNP-TSRL.

(b) **Case 2**, i.e., the implementation with introducing Gaussian noises $N_{Impl.}$, where various noises for implementation are set as shown in Table 2.2, that is, $N_{Impl.}50$, $N_{Impl.}100$, $N_{Impl.}150$ and $N_{Impl.}350$.

The fitness is averaged over 3000 trials, that is, each individual is implemented 300 times for 10 different start positions.

4. In Simulation 4, the best individuals of simulation 2 which are trained with introducing Gaussian noises are implemented in the environments shown in Fig 2.7(b). The implementations are done to study the robustness of Fuzzy GNP-RL like simulation 3.

2.5.4 Simulation Conditions

The node functions of the judgment nodes and processing nodes are shown in Table 2.3. Each judgment function J_0, ..., and J_7 judges the sensor values and determines the next node as explained in Section 2.3.2. Each processing node determines the speed of the left or right wheel. The parameters of evolution and learning are shown in Table 2.4. These values are selected appropriately through the simulations. At the end of each generation, 300 individuals are generated to form a new population in the next generation; 179 individuals are generated by mutation, 120 individuals are generated by crossover, and one individual is the elite individual. Each individual uses 61 nodes including 40 judgment nodes (5 for each kind), 20 processing nodes (10 for each kind) and one start node. Each of the judgment nodes and processing nodes has 2 sub nodes. The initial individuals in a population are determined as explained in section 2.3.3. To consider future rewards and to keep the balance of the exploration and exploitation, the learning parameters of $\alpha = 0.1$, $\gamma = 0.9$ and $\epsilon = 0.1$ are given.

Table 2.3: Node functions used in the function library

Symbol	ID	Functions
$j_0, ...J_7$	0, ...7	judge the value of the sensor 1,2,..., 8
P_0	0	determine the speed of the right wheel
P_1	1	determine the speed of the left wheel

2.5.5 Simulation Results

Result of Simulation 1

The idea of integrating Fuzzy logic to GNP-RL is to deal with the continuous values of sensors. While non-Fuzzy GNP-RL divides these values into two categories, that is,

Table 2.4: Simulation conditions

	Fuzzy GNP-RL	non-Fuzzy GNP-RL
The number of individuals	(300) mutation: 179, crossover: 120, elite: 1	
The number of nodes	(61) 20 processing nodes, 40 Fuzzy judgment nodes, and 1 start node	
The number of sub nodes	2 for each Fuzzy judgment and processing nodes	
Parameter of evolution	P_c = 0.1, P_m = 0.01, tournament sizes = 7	
Parameter of learning	α = 0.1, γ = 0.9, ϵ =0.1	
The life time (Training & Implementation)	1000 time steps	

Yes/No using the threshold values, Fuzzy GNP-RL divides these values into Yes/No category based on the degrees of truth using Fuzzy membership functions. Fig. 2.8 shows the fitness curves of the best individual of Fuzzy GNP-RL compared with non-Fuzzy GNP-RL in the training environment without noise (WN_{Train}). The fitness of Fuzzy GNP-RL converges faster and higher than that of non-Fuzzy GNP-RL. It confirms that the node transitions which are determined probabilistically can improve the search ability for determining the agent behaviors more appropriately.

Result of Simulation 2

Various Gaussian noises are used during the training phase as shown in Table 2.4, that is, mean μ of noises is usually taken to be 0, but the standard deviation is set at various values; $\sigma = 50$ (named N_{Train}50), $\sigma = 100$ (N_{Train}100), $\sigma = 150$ (N_{Train}150) and $\sigma = 350$ (N_{Train}350). The effects of introducing noises on Fuzzy GNP-RL and non-Fuzzy GNP-RL are shown in Fig. 2.9.

The noises change the sensor values, which make Fuzzy GNP-RL and non-Fuzzy GNP-RL determine inappropriate node transitions for the agent behaviors, then the fitness of the individuals trained with noises (Fig. 2.9) is lower than the fitness of the individuals trained without noises (Fig. 2.8). Here, larger noises make the fitness smaller as shown in Fig. 2.9. Larger noises make the judgment nodes more difficult to determine Yes/No category appropriately, as a result, various node transitions are determined and the individuals have more experiences to determine the behaviors. Thus, noises increase the exploration ability. However, because the incorrect information is included, inappropriate behaviors also occur, which decreases the fitness.

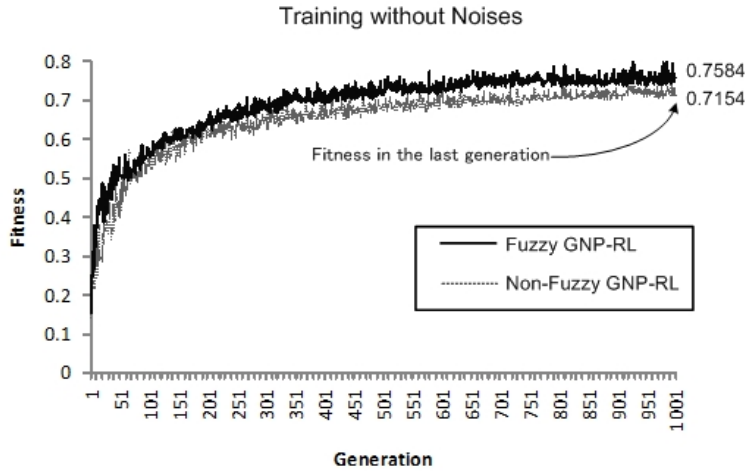


Figure 2.8: Fitness curves of Fuzzy GNP-RL comparing with non-Fuzzy GNP-RL in the training environment without noises

Result of Simulation 3

In this simulation, the individuals which are trained without introducing noises (WN_{Train}) are implemented in new environments without noises (Case 1) and also with noises (Case 2). The average rewards are calculated over 10 individuals from 10 independent training simulations as explained in sub-section 2.5.3 (Simulation 3), where the individuals are implemented 3000 times, that is, each individual is implemented 300 times for 10 different start positions of the robot. The average rewards in the implementation phase are shown in Table 2.5.

Table 2.5: Average rewards of individuals trained without introducing noises (WN_{Train}) in the implementation

		Implementation				
Training without Noises		Case 1	Case 2			
		$WN_{Impl.}$	$N_{Impl.}50$	$N_{Impl.}100$	$N_{Impl.}150$	$N_{Impl.}350$
WN_{Train}	Fuzzy GNP-RL	0.383	0.143	0.089	0.067	$*$
	non-Fuzzy GNP-RL	0.370	0.401	0.237	0.125	$*$

From Table 2.5 in Case 1, the average reward of Fuzzy GNP-RL is higher compared with that of non-Fuzzy GNP-RL in the implementation without noises ($WN_{Impl.}$). In this case, when the sensor values are less than β_{ip} or higher than α_{ip}, Fuzzy GNP-RL works in the same way as non-Fuzzy GNP-RL. However, when sensor values are in the range between α_{ip} and β_{ip}, Fuzzy GNP-RL has more flexibilities to determine the appropriate next nodes. Thus, the probabilistic transitions of Fuzzy GNP-RL improve the exploration ability.

However in Case 2, the average rewards of Fuzzy GNP-RL become worse compared with non-Fuzzy GNP-RL in the implementation with noises ($N_{Impl.}50$, $N_{Impl.}100$, $N_{Impl.}150$, etc.). In this case, Fuzzy GNP-RL which is trained without noises (WN_{Train}) learns and evolves with insufficient data. Then, when these individuals are imple-

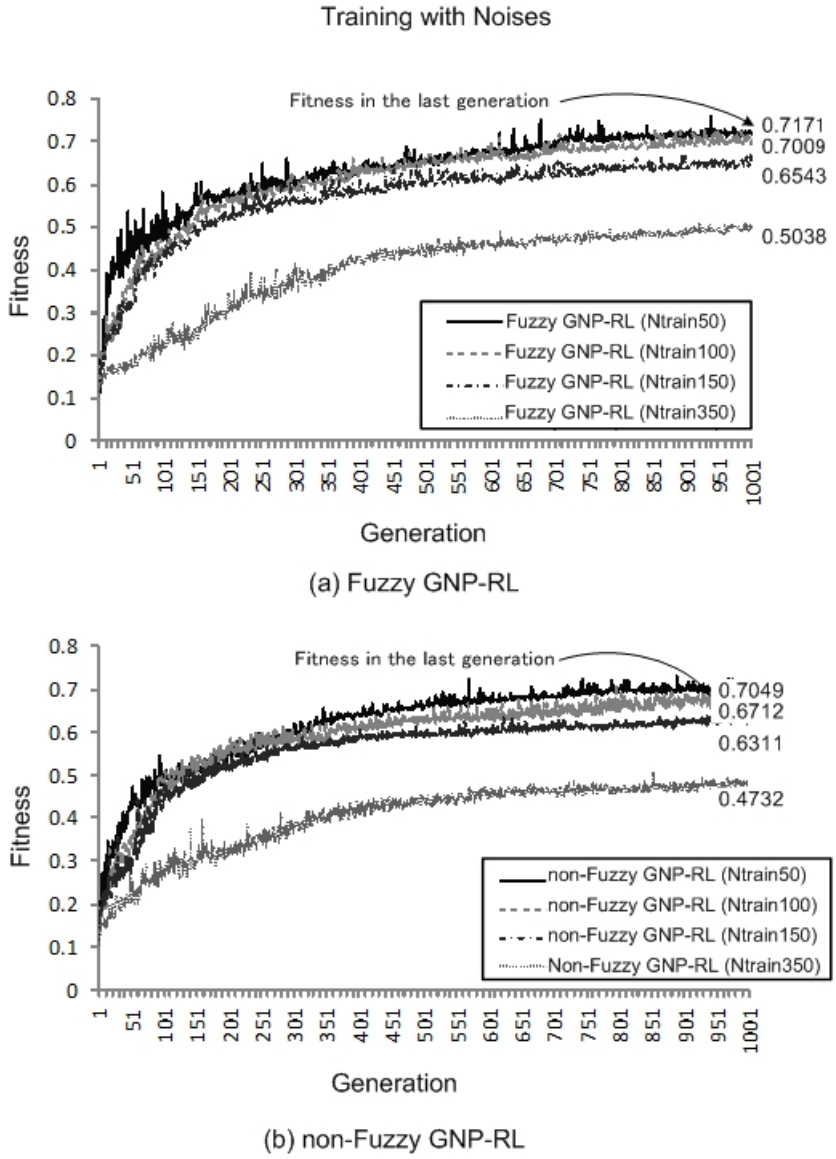


Figure 2.9: Fitness curves of Fuzzy GNP-RL comparing with non-Fuzzy GNP-RL in the training environment with noises

mented with noises, the probability to determine the Yes/No categories becomes difficult to learn/exploit the good connections, because the noises change the sensor values which make various probabilities. In order to overcome this problem, Fuzzy GNP-RL should be also trained with more data, but it takes a long time. Then, a simple way to solve this problem is to use artificial data, such as adding noises during the training phase.

On the other hand, non-Fuzzy GNP-RL trained without noises has better average reward than that of Fuzzy GNP-RL. When small noises ($N_{Impl.}50$) is used in the implementation, although the noises change the sensor values over the threshold, the node transitions are not changed so much. However, when larger noises ($N_{Impl.}150$) is used, the noises frequently change the sensor values over the threshold, then non-Fuzzy GNP-RL cannot learn the good connections for determining the appropriate node transitions. Thus, larger noises also decrease the average reward of non-Fuzzy GNP-RL.

Result of Simulation 4

This simulation learns the effects of introducing noises during the training phase and implementation phase. In this simulation, the individuals which have been trained with noises ($N_{Train}50$, $N_{Train}100$, $N_{Train}150$ and $N_{Train}350$) are implemented in the environment without noises (Case 1) and with noises (Case 2) to study the generalization ability. The simulation results are shown in Table 2.6.

Table 2.6: Average reward of individuals trained with noises and evaluated in the implementation

		Implementation				
Training with Noises		Case 1	Case 2			
		$WN_{Impl.}$	$N_{Impl.}50$	$N_{Impl.}100$	$N_{Impl.}150$	$N_{Impl.}350$
$N_{Train}50$	**Fuzzy GNP-RL**	**0.396**	**0.470**	**0.391**	**0.216**	$*$
	non-Fuzzy GNP-RL	**0.372**	**0.434**	**0.340**	**0.199**	$*$
$N_{Train}100$	Fuzzy GNP-RL	0.383	0.465	0.449	0.360	$*$
	non-Fuzzy GNP-RL	0.337	0.389	0.365	0.275	$*$
$N_{Train}150$	Fuzzy GNP-RL	0.355	0.411	0.415	0.374	0.078
	non-Fuzzy GNP-RL	0.296	0.365	0.362	0.348	0.115
$N_{Train}350$	Fuzzy GNP-RL	0.240	0.292	0.308	0.313	0.254
	non-Fuzzy GNP-RL	0.195	0.283	0.288	0.313	0.223

When the individuals of Fuzzy GNP-RL are trained with noises, they learn more different situations because noises change the sensor values, which make various probabilities to determine Yes/No categories. During the training phase, the appropriate Fuzzy parameters are evolved and Yes/No categories are determined with various probabilities. Then, Fuzzy GNP-RL carries out more various node transitions. In the training phase, larger noises make Fuzzy GNP-RL face more various situations, however, larger noises also make Fuzzy GNP-RL difficult to exploit the good connections, then the fitness decreases as shown in Fig. 2.9. The advantages of introducing noises

during the training are confirmed in the implementation as shown in Table 2.6.

- **Case 1**. In the implementation without noises ($WN_{Impl.}$), the results confirm that the exploration ability of Fuzzy GNP-RL is better than non-Fuzzy GNP-RL. The average reward of individuals of Fuzzy GNP-RL and non-Fuzzy GNP-RL which are trained with noises ($N_{Train}50$) in Table 2.6 is larger than those without noises (WN_{Train}) in Table 2.5. In the other words, introducing noises during the training phase increases the average reward in the implementation without noises. In this case, ($N_{Train}50$) can improve the exploration ability and determine the good connections which are useful for implementation.

- **Case 2**. In the implementation with noises, the individuals of Fuzzy GNP-RL which have been trained with noises ($N_{Train}50$, $N_{Train}100$, $N_{Train}150$ and $N_{Train}350$) have average reward larger than those of non-Fuzzy GNP-RL. In this case, introducing noises to Fuzzy GNP-RL improve the robustness, then the average rewards are better than those of non-Fuzzy GNP-RL. The individuals of Fuzzy GNP-RL trained with noises gets more explorations during the training phase and the node transitions learn various situations. Then, when they are implemented in the noisy environments, they have larger average rewards than those of Fuzzy GNP-RL trained without introducing noises during the training phase (WN_{Train}). The individuals with $N_{Train}50$ have the largest average reward in the implementation. Actually, the individuals trained with small noises ($N_{Train}50$) have more advantage to make explorations to determine the appropriate node transitions, then the average reward of $N_{Train}50$ is the largest. The larger noises make more exploration ability, however, the good connections are difficult to be determined. Thus, the average reward is decreased.

Overall, Fuzzy GNP-RL using the probabilistic node transitions improves the robustness of non-Fuzzy GNP-RL, because more various node transitions are learned by Fuzzy GNP-RL. In addition, introducing noises during the training phase gives more experiences for finding the appropriate node transitions in unknown environments. But, as too much noises in the training phase generate more changes of the training data, it decreases the fitness and also decreases the average reward in the implementation.

2.6 Summary

Fuzzy GNP-RL has been proposed to improve the robustness in the uncertain environments. The important part of Fuzzy GNP-RL is the judgment nodes, where Fuzzy logic is integrated. The training and implementation results show that Fuzzy GNP-RL can explore more various appropriate node transitions by using probabilistic node transitions.

In addition, introducing noises can also improve the robustness of Fuzzy GNP-RL, where the noises can change the sensor values and make more various node transitions. Then, noises increase the exploration ability and improves the robustness of Fuzzy GNP-RL in uncertain environments.

Thus, the robustness of Fuzzy GNP-RL with noises is confirmed in this chapter. The advantage of the integrating Fuzzy logic to judgment nodes is also used for the

next chapters. So, improving the adaptability of Fuzzy GNP-RL under the changes of environments is considered in the next chapter.

Chapter 3

Fuzzy GNP-TSRL (BS) with Fixed ϵ-greedy Policy and Learning Rate α

3.1 Introduction

In Chapter 2, integrating Fuzzy logic to GNP-RL confirmed that Fuzzy GNP-RL improves the robustness in uncertain environments. In this Chapter, improving the adaptability Fuzzy GNP-RL is studied. The structures of Fuzzy GNP-RL are similar to the conventional GNP-RL, that is, the node structures consist of several sub nodes to represent functions, which are selected by a policy learned using Sarsa learning. GNP-RL has advantages to adaptively change the programs when inexperienced troubles occur, that is, sensors' break suddenly in the implementation [44]. In this case, Fuzzy GNP-RL has the same mechanism to adapt to the changes of environments.

Actually, when sensors' troubles occur, the robot can perceive the environments incorrectly and create inappropriate behaviors which may be dangerous. In this case, Fuzzy GNP-RL has an adaptation mechanism to change the programs adaptively using the Sub node Selection method (SS method), that is, sub nodes/functions of judgment and processing nodes are determined without directly detecting broken sensors, but indirectly by detecting the troubles through the changes of the Q_{SS} values.

In paper [44], the adaptability to cope with the sudden changes of the environments were studied with various number of sub nodes. A larger number of sub nodes makes more alternative functions which can be selected according to the Q_{SS} values. However, a larger number of sub nodes has difficulties to exploit good sub nodes, then GNP-RL takes longer times to recover from the troubles. Thus, this chapter aims to enhance the adaptability of Fuzzy GNP-RL, when the sudden changes of the environments occur by broken sensors.

3.2 Motivation of GNP-TSRL (BS)

In order to enhance the adaptability of Fuzzy GNP-RL when some sensors break in the implementation, Fuzzy GNP with Two-Stage Reinforcement Learning using Branch connection Selection, i.e., Fuzzy GNP-TSRL (BS) is studied.

The basic concept is that when the functions of the current node give inappropriate

agent behaviors, executing the next nodes properly may recover the troubles more quickly. Then, beside using the alternative functions like Fuzzy GNP-RL, the ability to search the next nodes is increased using a larger number of connections in order to quickly recover the troubles. Due to the increase of the number of connections, the Branch connections Selection method (BS method) is used, that is, the selection of the branch connections should be learned using Reinforcement Learning. Then, Fuzzy GNP-TSRL (BS) has two-stage learning, that is, the Sub node Selection method (SS method) and Branch connection Selection (BS method) which are learned in the first stage and in the second stage of RL, respectively.

Therefore, the structures and learning processes of Fuzzy GNP-RL are enhanced in Fuzzy GNP-TSRL (BS). While the SS method works in the same way as Fuzzy GNP-RL, the BS method selects branch connections to determine the next nodes, that is, when the current node determines inappropriate functions, the proposed method should select another connections to determine the next nodes which may recover the troubles quickly. The BS method can change the programs adaptively by selecting the branch connections and determining the next nodes to be executed.

In order to study the adaptability, the performance of Fuzzy GNP-TSRL (BS) which uses Two-Stage Reinforcement learning is compared with Fuzzy GNP-RL which uses One-Stage Reinforcement Learning in the training phase and implementation phase.

3.3 Algorithm of Fuzzy GNP-TSRL (BS)

In order to realize Fuzzy GNP-TSRL (BS), the judgment nodes and processing nodes should be modified. The node structures of the proposed method are described in subsection 3.3.1. The mechanisms of determining the node transitions and learning processes are described in sub-section 3.3.2. In addition, the recovery process is explained in sub section 3.3.3 to show the advantages of the proposed.

3.3.1 Structure Representation

The individual of Fuzzy GNP-TSRL (BS) is represented by a directed graph structure as the standard GNP shown in Fig. 1.11, which consists of 3 kind nodes, that is, start node, judgment node and processing node. The functions of these nodes have been introduced in section 2.3.1. The combination of the SS method and BS method are realized on the judgment nodes and processing nodes as shown in Fig. 3.1.

In order to learn these nodes efficiently and effectively, the learning processes are carried out into 2 stages, that is, the SS method and BS method are learned at the first stage and second stage of RL, respectively using two kinds of Q tables, i.e., Q_{SS} table and Q_{BS} table. Therefore, the genes of Fuzzy GNP-TSRL (BS) are also modified as shown in Fig. 3.2.

It is supposed that node $i \in \{0, 1, ..., n-1\}$ has m sub nodes. The gene structure of node i is divided into macro node part and sub node part.

Macro Node Part

The macro node part of node i is defined by NT_i and d_i. Where NT_i represents a node type and d_i represents the time delay spent on executing node i, where NT_i and d_i are shown in Table 2.1.

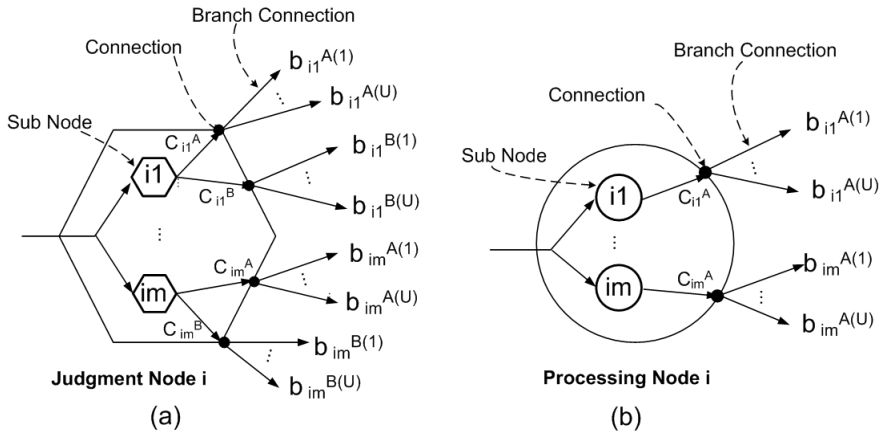


Figure 3.1: Node structures of Fuzzy GNP-TSRL (BS)

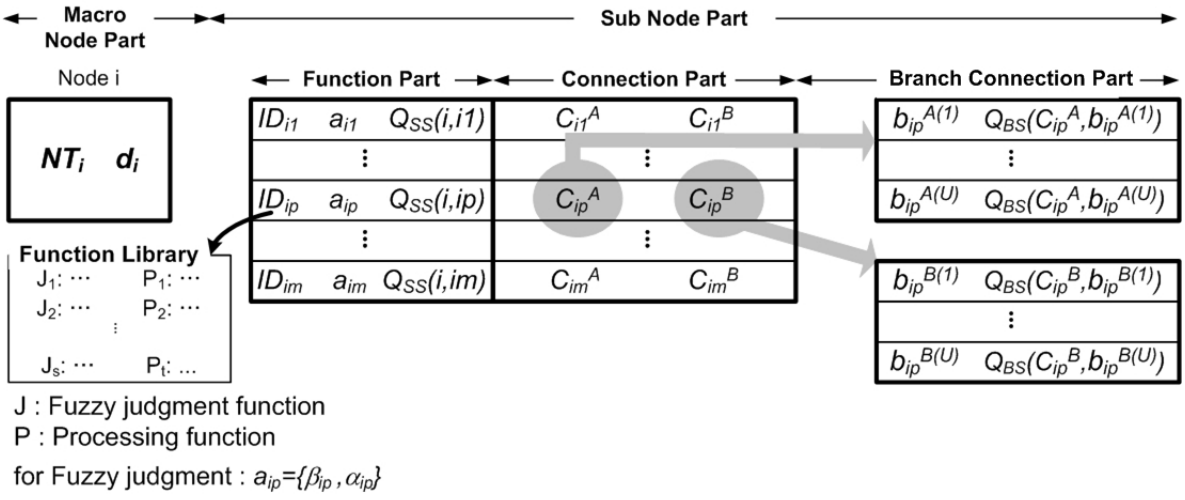


Figure 3.2: Gene structures of Fuzzy GNP-TSRL (BS)

Sub Node Part

The sub node part of Fuzzy GNP-TSRL (BS) is composed of 3 parts, i.e., function part, connection part and branch connection part. While the function part and connection part are the same as Fuzzy GNP-RL, the new part is the branch connection part, that is, each connection has U branch connections.

The function of sub node $p \in \{1, ..., m\}$ of node i is defined by ID_{ip}, a_{ip} and $Q_{SS}(i, ip)$, as explained in sub-section 2.3.1. The SS method of Fuzzy GNP-TSRL (BS) is determined based on ϵ-greedy policy according to the $Q_{SS}(i, ip)$ value at the first stage of RL. Here, the state is the current node i and the action is sub node ip selection, that is, sub node p of node i.

Here, the definitions of the connections and branch connections are defined as follows.

- **Definition 3.1: Connection**. Generally, the judgment nodes have several branches connected to the other nodes. Here, each sub node of the judgment nodes uses 2 branches which are determined by the judgment results, that is , sub node p of judgment node i has connections C^A_{ip} and C^B_{ip} which are selected probabilistically (see sub-section 2.3.2). On the other hand, the processing nodes have only one branch, which determines the next node to be executed, then sub node p of processing node i has connection C^A_{ip} only.

- **Definition 3.2: Branch Connection**. The idea of Fuzzy GNP-TSRL (BS) is to increase the search ability by increasing the number of connections to determine the next nodes. Therefore, each connection (in Definition 3.1) is enhanced by adding U branch connections, then connection C^A_{ip} has branch connections $\{b^{A(1)}_{ip}, ..., b^{A(U)}_{ip}\}$ and connection C^B_{ip} has branch connections $\{b^{B(1)}_{ip}, ..., b^{B(U)}_{ip}\}$.

In order to determine the appropriate next nodes, the selection of branch connections which connects the current node to the next node should be also learned. It is supposed that sub node p of node i selects connection C^A_{ip}, then branch connection $b^{A(u)}_{ip} \in \{b^{A(1)}_{ip}, ..., b^{A(U)}_{ip}\}$ can be selected based on ϵ-greedy policy according to $Q_{BS}(C^A_{ip}, b^{A(u)}_{ip})$ value. Here, the state is connection C^A_{ip} and action is branch connection selection $b^{A(u)}_{ip}$. On the other hand, while sub node p of node i selects connection C^B_{ip}, then branch connection $b^{B(u)}_{ip} \in \{b^{B(1)}_{ip}, ..., b^{B(U)}_{ip}\}$ can be selected based on ϵ-greedy policy according to $Q_{BS}(C^B_{ip}, b^{B(u)}_{ip})$ value, here, the state is connection C^B_{ip} and action is branch connection selection $b^{B(u)}_{ip}$. The Branch connection Selection method (BS method) is done at the second stage of RL.

3.3.2 Learning Process of Fuzzy GNP-TSRL (BS)

The learning process of Fuzzy GNP-TSRL (BS) implements two kinds of Sarsa learning algorithms, that is, the Sub node Selection (SS method) at the first stage and the Branch connection Selection (BS method) at the second stage of RL. When a good action is taken with a positive reward at a certain state, the action is reinforced and will be selected with higher probability when the state is visited again. Using Fig. 3.3, the process of two kinds of Sarsa learning is explained as follows.

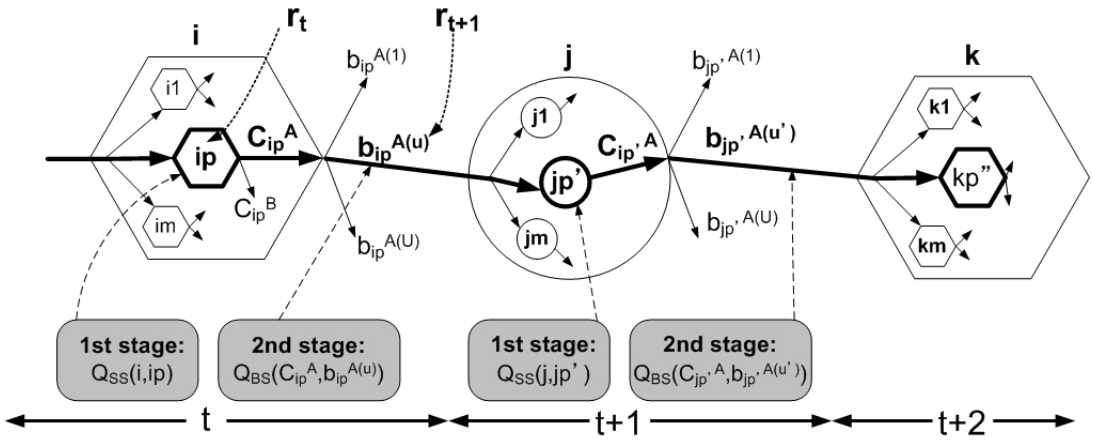


Figure 3.3: Learning process of Fuzzy GNP-TSRL (BS)

1. At time t, it is supposed that the current node is node i, and Fuzzy GNP-TSRL (BS) refers to all Q_{SS} values for sub node selection, i.e., $\{Q_{SS}(i,i1),...,Q_{SS}(i,im)\}$, and selects one of them based on ϵ-greedy policy. It is supposed that Fuzzy GNP-TSRL (BS) selects $Q_{SS}(i,ip) \in \{Q_{SS}(i,i1),...,Q_{SS}(i,im)\}$, and the corresponding node function ID_{ip} and parameter a_{ip} are selected. This step is the first stage of RL.

2. Then, Fuzzy GNP-TSRL (BS) executes function ID_{ip} using parameter a_{ip} and determines a node connection. When the current node is a judgment node, the node connection is determined by the judgment result which can be C^A_{ip} or C^B_{ip}, however, when the current node is a processing node the the connection is only C^A_{ip}. After executing the function, Fuzzy GNP-TSRL (BS) gets reward r_t.

3. In this step, the second stage of RL is started, i.e., the BS method. Fuzzy GNP-TSRL (BS) refers to all Q_{BS} of the branch connections which are connected to connection C^A_{ip} or C^B_{ip} resulted from step 2 and selects one of them based on ϵ-greedy policy. When connection C^A_{ip} is resulted from step 2, a branch connection is selected referring to $\{Q_{BS}(C^A_{ip}, b^{A(1)}_{ip}),...,Q_{BS}(C^A_{ip}, b^{A(u)}_{ip}),...,Q_{BS}(C^A_{ip}, b^{A(U)}_{ip})\}$. On the other hand, when connection C^B_{ip} is resulted from step 2, a branch connection is selected referring to $\{Q_{BS}(C^B_{ip}, b^{B(1)}_{ip}),...,Q_{BS}(C^B_{ip}, b^{B(u)}_{ip}),...,Q_{BS}(C^B_{ip}, b^{B(U)}_{ip})\}$. It is supposed that $Q_{BS}(C^A_{ip}, b^{A(u)}_{ip})$ is selected, then branch connection $b^{A(u)}_{ip}$ of connection C^A_{ip} is determined. Here, the branch connection shows that the next node is node j.

4. At time $t+1$, Fuzzy GNP-TSRL (BS) gets reward r_{t+1} and repeats step 1 to 3 for node j. Here, it is supposed that $Q_{SS}(j,jp')$ and $Q_{BS}(C^B_{jp'}, b^{B(u')}_{jp'})$ are selected.

5. Then, the Q values are updated by the following procedures. Updating Q_{SS}-values at the first stage of RL is

$$Q_{SS}(i,ip) \leftarrow Q_{SS}(i,ip) + \alpha(r_t + \gamma Q_{SS}(j,jp') - Q_{SS}(i,ip)), \tag{3.1}$$

and updating Q_{BS}-values at the second stage of RL is

$$\begin{aligned} &Q_{BS}(C^A_{ip}, b^{A(u)}_{ip}) \leftarrow Q_{BS}(C^A_{ip}, b^{A(u)}_{ip}) + \\ &\alpha(r_{t+1} + \gamma Q_{BS}(C^B_{jp'}, b^{B(u')}_{jp'}) - Q_{BS}(C_{ip^A}, b^{A(u)}_{ip})), \end{aligned} \tag{3.2}$$

where, α is a learning rate ($0 < \alpha \leq 1$) and γ is discount rate ($0 \leq \gamma \leq 1$).

6. $t \leftarrow t+1$, $i \leftarrow j$ and $p \leftarrow p'$. Then, it returns to step 1.

3.3.3 Recovery Process of Fuzzy GNP-TSRL (BS)

Definition 3.3: Recovery process. The recovery process of Fuzzy GNP-TSRL (BS) has the adaptability mechanisms to adaptively change the programs in order to revise incorrect behaviors of the agent.

In order to explain the recovery process understandability, an example of wall behaviors for a Khepera robot is used, where the characteristic of a Khepera robot has been described in sub-section 2.5.1 and the reward is calculated by Eq. 2.7. The behaviors of a Khepera robot can be explained as follows according to the speeds of the right wheel ($V_R(t)$) and left wheel ($V_L(t)$) at time step t.

- Move forward. The robot moves forward when $V_R(t) = V_L(t)$.
- Turn right. The robot turns right when $V_R(t) < V_L(t)$.
- Turn left. The robot turns left when $V_R(t) > V_L(t)$.

Fuzzy GNP-TSRL (BS) can adaptively change the programs (1) by changing the functions and/or (2) by changing the connections. It is supposed that each node of Fuzzy GNP-TSRL (BS) has 2 functions represented by 2 sub nodes and each connection of a sub node has 2 branch connections (see Fig. 3.1).

If the robot moves forward in the current situation, where the robot faces the wall in front of it and on the right side, then the appropriate behavior decided by the robot is to turn left. In order to adapt to this situation, Fuzzy GNP-TSRL (BS) may make recovery processes as follows.

Recovery Process by Changing the Function

It is supposed that the speeds of the robot are $V_R = 7$ and $V_L = 7$ as shown in Fig. 3.4, where Fuzzy GNP-TSRL (BS) executes processing node i which has 2 functions, i.e., sub node $i1$ sets the speed of the left wheel to $V_L = -10$ and sub node $i2$ sets the speed of the right wheel to $V_R = -3$. In the current situation, the Q values of sub nodes are $Q_{SS}(i,i1) = 80$ and $Q_{SS}(i,i2) = 100$. It is supposed that sub node $i2$ is selected by ϵ-greedy policy. After executing the function, the speeds of the robot are to be $V_R = -3$ and $V_L = 7$, which make the robot turns to right and hits the wall, so that $Q_{SS}(i,i2)$ is decreased to 60.

The the recovery process is carried out as shown in Fig. 3.4. After executing several nodes, processing node i maybe visited again. In this case, the Q values of sub nodes of node i are $Q_{SS}(i, i1) = 80$ and $Q_{SS}(i, i2) = 60$, where the alternative function, that is, sub node $i1$ may be selected by ϵ-greedy policy. After executing the function, the speeds of the robot become $V_R = -3$ and $V_L = -10$ which make the robot change the direction and turn to left, so that a large reward is given to increase the value of $Q_{SS}(i, i1)$.

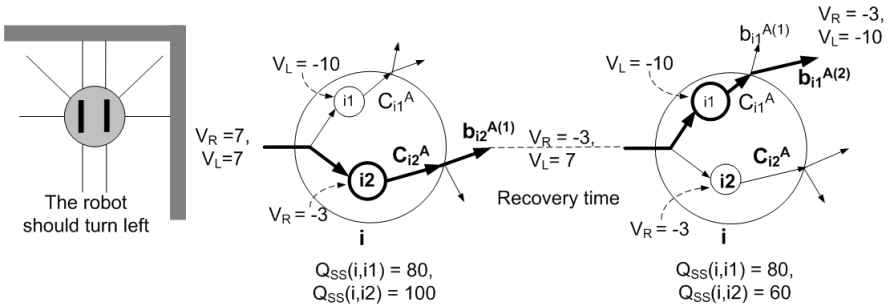


Figure 3.4: Recovery process by changing the function

Recovery Process by Changing the Connection

The recovery process may take a longer time step when changing the functions, because visiting the same node in the node transitions is difficult. The simple way for recovering the process is carried out by selecting the appropriate next node which may change the agent behavior appropriately.

It is supposed that the situation is the same as above, that is, the wheel speeds of the robot are $V_R = 7$ and $V_L = 7$. Then, Fuzzy GNP-TSRL (BS) executes processing node i which has 2 functions, i.e., sub node $i1$ sets the speed of the left wheel to $V_L = -10$ and sub node $i2$ sets the speed of the right wheel to $V_R = -3$. In the current situation, the Q values of sub nodes are $Q_{SS}(i, i1) = 80$ and $Q_{SS}(i, i2) = 100$. It is supposed that sub node 2 is selected by ϵ-greedy policy. After executing the function, the speeds of the robot are $V_R = -3$ and $V_L = 7$ which make the robot turn right and hits the wall, so that a small reward is given.

The the recovery process is carried out as shown in Fig. 3.5. After executing sub node $i2$ of node i, Fuzzy GNP-TSRL (BS) determine the next node by selecting the appropriate branch connections. It is supposed that branch connection $b_{i2}^{A(1)}$ is selected by ϵ-greedy policy according to Q value $Q_{BS}(C_{i2}^{A}, b_{i2}^{A(1)})$. Here, node i is transfered to node j, where sub node $i1$ of node j is selected, then the wheel speeds of the robots become $V_R = -10$ and $V_L = -3$ which make the robot change the direction and turn left, so a large reward is given to increase the value of $Q_{BS}(C_{i2}^{A}, b_{i2}^{A(1)})$.

Thus, the combination of SS method and BS method of Fuzzy GNP-TSRL (BS) may recover the troubles quickly than only using SS method as Fuzzy GNP-RL.

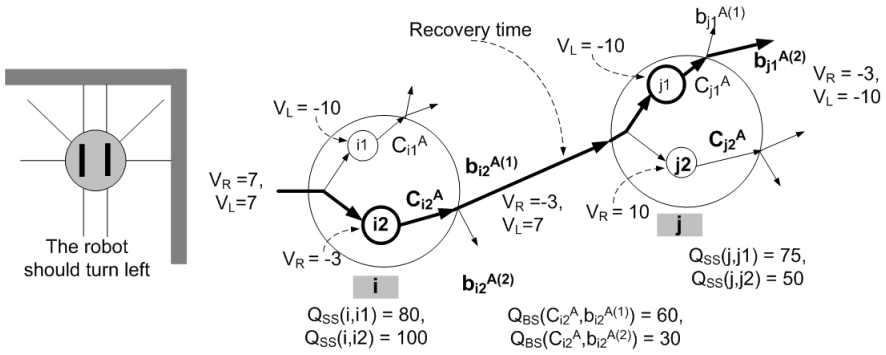


Figure 3.5: Recovery process by changing the connection

3.4 Comparison between Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL

Fuzzy GNP-TSRL (BS) is an enhancement of Fuzzy GNP-RL. While Fuzzy GNP-RL uses One-Stage RL of the SS method, GNP-TSRL uses Two-Stage RL of the SS method combined with BS method. Thus, the recovery process of Fuzzy GNP-TSRL (BS) is also enhanced when severe troubles occur, that is, Fuzzy GNP-TSRL (BS) can adaptively change the programs by changing the functions like Fuzzy GNP-RL, in addition, Fuzzy GNP-TSRL (BS) can adaptively change the programs by selecting the appropriate next nodes. Then, the recovery process of Two-Stage RL can work faster than that of One-Stage RL. As a result, the performance of Fuzzy GNP-TSRL (BS) also can be enhanced, especially when severe troubles occur.

3.5 Simulations

The performance of Fuzzy GNP-TSRL (BS) is studied in the benchmark of the wall following behaviors of a Khepera robot using Webots simulator [71], where the specification of the robot and calculation of the reward has been introduced in section 2.5.1 and section 2.5.2, respectively.

3.5.1 Simulation Environments

The performance of Fuzzy GNP-TSRL (BS) which uses Two-Stage RL is evaluated and compared with Fuzzy GNP-RL which uses One-Stage RL. These methods have advantages to adaptively change the programs to recover from the troubles. Then, in order to confirm the advantages of the proposed method, the performance of Two-Stage RL is studied compared with One-Stage RL in two aspects, that is, in the training phase and in the implementation phase using the environments shown in Fig. 2.7.

1. **Simulation 1**. In the training phase, the individuals of Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL are evaluated and evolved generation by generation, where the functions, parameters, and branch connections are changed by crossover and mutation. The life time for evaluation of each individual is 1000 time steps in each generation. In this simulation Fuzzy logic is integrated to the judgment

nodes of Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL. Gaussian noises ($\mu = 0$, $\sigma = 50$) are introduced as was done in chapter 2.

2. **Simulation 2**. In order to study the adaptability of Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL, the best individuals from the training phase are implemented in the implementation environments, where the life time for the implementation of the individuals is 3000 time steps which are separated as follows.

   - From the first time step to the 500th time step, the individuals is implemented in the normal situations where Gaussian noises ($\mu = 0$, $\sigma = 50$) are also introduced.
   - In order to study the adaptability of Fuzzy GNP-TSRL (BS) comparing with Fuzzy GNP-RL, an inexperienced sudden changes occur by making a sensor to break at the 500th time step, after that the adaptability of recovering the troubles is analyzed.

3. **Simulation 3**. The adaptability of Fuzzy GNP-TSRL and Fuzzy GNP-RL is studied furthermore when severe troubles occur. The same simulation is done as simulation 2, however, the inexperienced sudden changes occur by making several sensors break at the 500th time step, that is, 2, 3, 4 and 5 sensors break. The performances of the proposed method for recovering the troubles are analyzed from the 500th time step to the life time step.

3.5.2 Simulation Conditions

The functions of the nodes in this chapter are the same as the the previous chapter which is shown in Table 2.3. On the other hand, the simulation conditions of Fuzzy GNP-TSRL (BS) are shown in Table. 3.1 comparing with Fuzzy GNP-RL .

In the training phase, 300 individuals are evolved, where at the end of each generation, 300 individuals are generated to form a new population for the next generation; 179 individuals are generated by mutation, 120 individuals are generated by crossover, and one individual is the elite. The evolution parameters are P_c = 0.1, P_m = 0.01 and tournament sizes = 7. Each individual uses 61 nodes including 40 judgment nodes (5 for each kind), 20 processing nodes (10 for each kind) and one start node. Each judgment node and processing node of Fuzzy GNP-TSRL (BS) has 2 sub nodes, and each node connection of the sub node has 2 branch connections ($2SS$&$2BS$) determined by the evolution, while those of Fuzzy GNP-RL have 4 sub nodes ($4SS$). Here, Fuzzy GNP-TSRL (BS) recovers the troubles by changing functions and connections according to the Q_{SS} table and Q_{BS} table, while Fuzzy GNP-RL recovers the troubles by changing the functions only according to the Q_{SS} table. The learning rate $\alpha = 0.1$, discount rate $\gamma = 0.9$, $\epsilon_{SS} = 0.1$ and $\epsilon_{BS} = 0.1$ are given to consider future rewards and to keep the balance of the exploration and exploitation.

3.5.3 Simulation Results

The performance of the Two-Stage RL compared with the One-Stage RL is confirmed in these simulation results.

Table 3.1: Simulation conditions of Fuzzy GNP-TSRL (BS) comparing with Fuzzy GNP-RL

	Fuzzy GNP-TSRL (BS)	Fuzzy GNP-RL
The number of individuals	(300) mutation: 179, crossover: 120 elite: 1	
The number of nodes	(61) 20 processing nodes 40 judgment nodes 1 start node)	
Kinds of Q tables	2	1
Actions Number of Sub Nodes Number of Branch Connections	(2SS & 2BS) 2 2	(4SS) 4 -
Recovery mechanisms	Changing functions and connections	Changing functions only
Parameter of evolution	P_c = 0.1, P_m = 0.01, tournament sizes = 7	
Parameter of learning	α = 0.1, γ = 0.9, ϵ_{SS} =0.1, ϵ_{BS} =0.1,	

Results of Simulation 1

In this simulation, the performances of Fuzzy GNP-TSRL (BS) using the Two-Stage RL are studied comparing with Fuzzy GNP-RL using One-Stage RL in the training phase. Fig. 3.6 shows the average fitness of these methods, where each curve is averaged over 10 best individuals from 10 independent training simulations in each generation. Each individual is trained with 1000 time steps and evolved until 1000 generations. The structures of the individuals are determined by evolution and they contain the learning information saved in the Q tables.

The combination of the SS method and BS method in Fuzzy GNP-TSRL (BS) improves the search ability, because the number of connections from each node is to be twice as many as One-Stage RL, when the number of sub nodes in the nodes is the same. Then, to make fair comparison, Fuzzy GNP-TSRL (BS) uses *2SS*&*2BS*, while Fuzzy GNP-RL uses *4SS*. Actually, using these structures the size of Q tables of the Two-Stage RL becomes larger than that of One-Stage RL, which can make the Q values of the Two-Stage RL take longer times to converge. However, as shown in Fig. 3.6, the curve of Fuzzy GNP-TSRL (BS) converges faster and higher than that of Fuzzy GNP-RL, because the process can recover incorrect situations quickly. It is confirmed that the learning process of Fuzzy GNP-TSRL (BS) combining the SS method and BS method is carried out more efficiently and effectively than Fuzzy GNP-RL.

Results of Simulation 2

In this simulation, the adaptability of Fuzzy GNP-TSRL (BS) in the implementation phase is studied comparing with Fuzzy GNP-RL . In this case, one sensor breaks at the

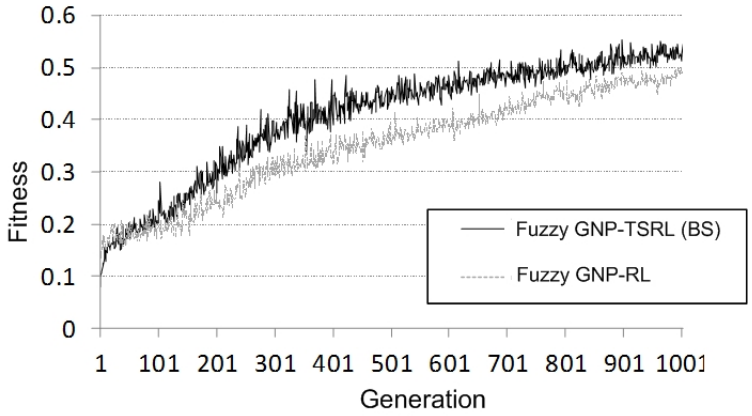


Figure 3.6: Average fitness of Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL in the training phase

500th time step. Then, the performances of the Two-Stage RL are analyzed before and after the sensor breaks comparing with the One-Stage RL. Fig. 3.7 shows the average rewards in each time step over 3000 simulations and Fig. 3.8 shows the behaviors which are carried out by Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL, when one sensor breaks at the 500th time step.

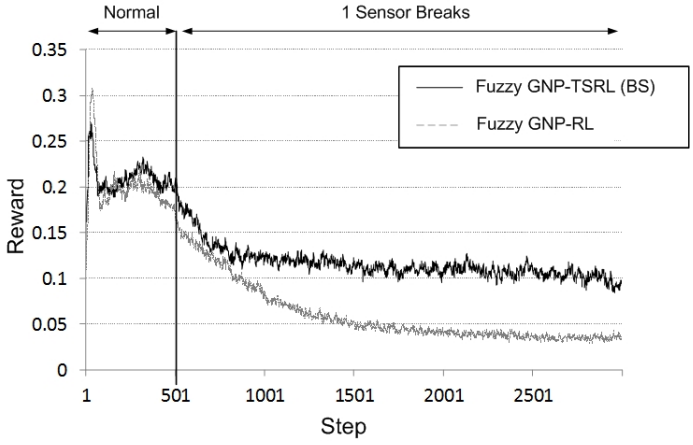
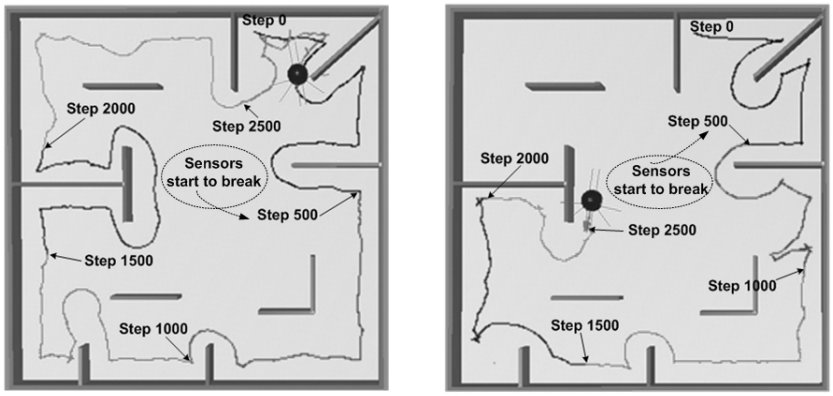


Figure 3.7: Average reward in each time step when 1 sensor breaks in the implementation phase

In the normal situation, that is, no sensor breaks in the implementation, the individuals learn different environments from the training phase, which means they face inexperienced situations and should determine the appropriate behaviors. It is seen from Fig. 3.7 that the average reward of Fuzzy GNP-TSRL (BS) is higher than that of Fuzzy GNP-RL in each time step (time step 1-500). It means that Fuzzy GNP-TSRL (BS) determines the behaviors appropriately, that is, the learning process to determine the node transitions using combination of the SS method and BS method is more efficient and effective than only using the SS method by Fuzzy GNP-RL. In the other



(a) Fuzzy GNP-TSRL (BS) (b) Fuzzy GNP-RL

Figure 3.8: Trajectory path of Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL when 1 sensor breaks in the implementation phase

words, Fuzzy GNP-TSRL (BS) has more adaptability . It is also confirmed from the Fig. 3.8 that the trajectory path of Fuzzy GNP-TSRL (BS) shows longer than that of Fuzzy GNP-RL before a sensor breaks.

After a sensor breaks at the time step 500th, the robot has problems to determine its behaviors which may hit the wall or move away from the wall so these behaviors make the reward smaller than in the normal situations as shown in Fig. 3.8. However, the average reward of Fuzzy GNP-TSRL (BS) is still higher than that of Fuzzy GNP-RL in each time step as shown in Fig. 3.7. As explained in sub-section 3.3.3, the recovery processes of the Two-Stage RL are done by changing functions and the connections adaptively when troubles occur. These recovery processes make Fuzzy GNP-TSRL (BS) can revise the robot behaviors faster and more appropriately than Fuzzy GNP-RL. As a result, the trajectory path of Fuzzy GNP-TSRL (BS) is longer to follow the wall than that of Fuzzy GNP-RL. Thus, the combination of the SS method and BS method works more efficiently and effectively to determine the appropriate behaviors in the implementation where inexperienced troubles occur.

Results of Simulation 3

The efficiency and effectiveness of the combination of the SS method and BS method are also studied when severe troubles occur, that is, when several number of sensors break in the implementation. A larger number of broken sensors make large troubles in the environments, which means that the recovery processes are hard, then the average reward decreased as shown in Fig. 3.9. However, Fuzzy GNP-TSRL (BS) could determine the behaviors more appropriately when severe troubles occur comparing with Fuzzy GNP-RL.

The trajectory path of Fuzzy GNP-TSRL (BS) is shown in Fig. 3.10. When severe troubles occur, the trajectory becomes shorter than when only one sensor breaks in the implementation, which means that the recovery processes are difficult to determine the appropriate behaviors. In addition, the trajectory of Fuzzy GNP-RL almost fail to follow the wall in this situation, because the recovery processes of One-Stage take longer time by only changing the functions, then good functions are hard to be

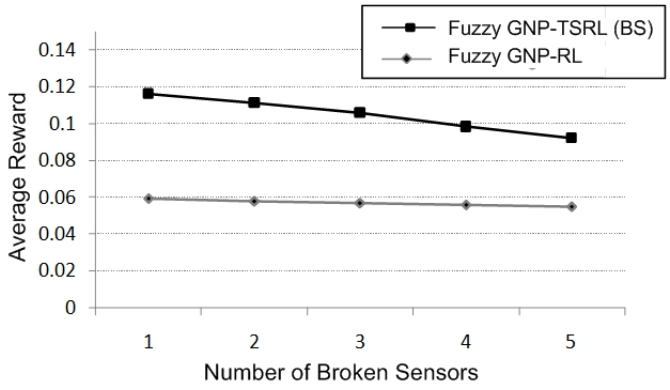


Figure 3.9: Average reward of Two-Stage RL and One-Stage RL after several sensors break in the implementation phase

learned.

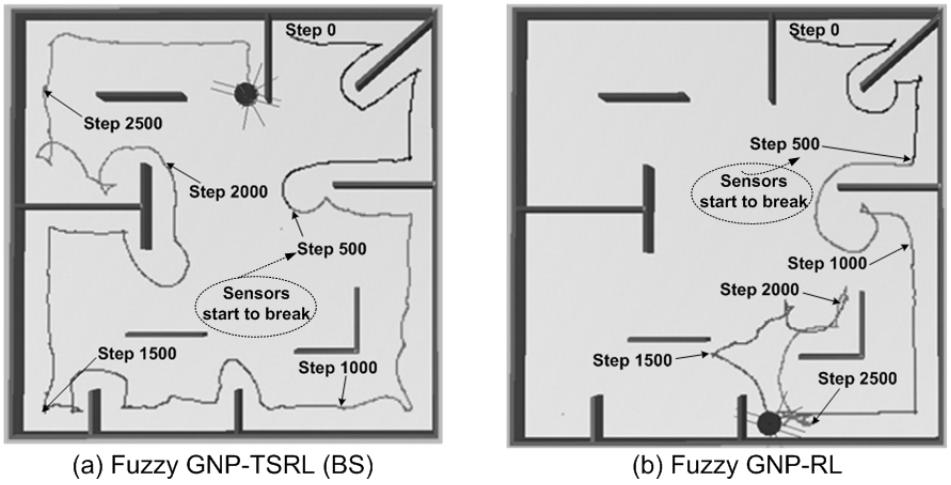


Figure 3.10: Trajectory path of Two-Stage RL and One-Stage RL when five sensors break in the implementation phase

3.6 Summary

Fuzzy GNP-TSRL (BS) has been proposed to improve the adaptability of Fuzzy GNP-RL which combines the SS method and BS method in the implementation with inexperienced troubles, such as broken sensors. Using the combination of the SS method and BS method, the learning process of Fuzzy GNP-TSRL (BS) works more efficiently and effectively than Fuzzy GNP-RL. Then, the recovery process of Fuzzy GNP-TSRL (BS) is faster.

Adding learning for the branch connection selection makes the size of Q tables of Fuzzy GNP-TSRL (BS) larger than that of Fuzzy GNP-RL, when the number of sub nodes is the same, then the Q values usually takes longer time to converges. In this

case, a method to make the balance of the exploration and exploitation is needed to improve the performance of Fuzzy GNP-TSRL. This method will be studied in the next chapter.

Chapter 4

Fuzzy GNP-TSRL (BS) with Changing ϵ-greedy Policy and Learning Rate α

4.1 Introduction

The objective of the agent is to maximize the reward accumulated using a policy to select an action for each state. In Reinforcement Learning (RL), the balance between the exploitation and exploration is an important issue [73] , where the environments must be sufficiently explored for action selections during learning. The basic exploration technique is realized by using the randomness of the exploration, that is, the actions are selected with a probability distribution. The basic policy for the action selection is ϵ-greedy policy [21] which allows a certain degree of random explorations with the probability of ϵ.

In chapter 3, Fuzzy GNP-TSRL (BS) has been proposed to improve the adaptability of Fuzzy GNP-RL using two-stage action selection of the SS method and BS method, that is, the SS method is carried out in the first stage and BS method in the second stage of RL. These action selections are carried out using fixed parameters of ϵ-greedy policy and learning rate α. These fixed parameters are used in general, where a small value of ϵ means that the agent makes small explorations, while a large value means that the agent will make large explorations and learn from them. However, large explorations make the agent not use what it has learned previously. On the other hand, the exploration of the environments impacts on the speed of the learning process, which may result in inefficient learning time. Then, balancing the exploration and exploitation should be considered carefully.

Generally, the agent knows nothing about the environment at the first step, therefore, larger random action selections are carried out to explore behaviors at the beginning. When the agent learns more about the environments, it becomes possible to estimate which actions are suitable. Then, decreasing ϵ value seems appropriate to estimate the good actions more. Changing ϵ value is motivated to control the balance of the exploration and exploitation in order to improve the performance of the agent.

4.2 Motivation of Changing Parameters of Fuzzy GNP-TSRL (BS)

The size of Q tables of Fuzzy GNP-TSRL (BS) is larger than Fuzzy GNP-RL for the same number of sub nodes, thus a policy for action selection for Fuzzy GNP-TSRL (BS) is to be studied furthermore to maximize the reward accumulated. Then, a method of changing ϵ-greedy for Fuzzy GNP-TSRL (BS) is considered in this chapter.

Fuzzy GNP-TSRL (BS) does the learning process in both of the training phase and implementation phase. Then, the changing method is studied as follows.

1. In the training phase, both of the Q_{SS} values and Q_{BS} values are initialized at zero in the first generation. Then, firstly Fuzzy GNP-TSRL (BS) determines the node transitions with larger random action selections to make large explorations. Because the size of Q tables of Fuzzy GNP-TSRL (BS) which is grouped as the Q_{SS} table and Q_{BS}table is larger than Fuzzy GNP-RL, larger explorations are needed to learn the good structures. On the other hand, the ability of estimating the action selections should be improved in later generations by decreasing the exploration. In addition, learning rate α determines to what extent the newly acquired information will override the old information. Then, when learning rate α is larger in early generations, the Q values of Fuzzy GNP-TSRL (BS) are updated considering the recent information more, when learning rate α is decreased in later generations, the Q values of Fuzzy GNP-TSRL (BS) are updated considering the old information more. Thus, changing ϵ-greedy policy and learning rate α generation by generation may improve the performance of Fuzzy GNP-TSRL (BS).

2. In the implementation phase, as Fuzzy GNP-TSRL (BS) has the ability to change the programs adaptively when troubles occur, increasing the exploration may recover the troubles quickly and may determine the behaviors more appropriately. However, the exploration should be decreased after Fuzzy GNP-TSRL (BS) recovers the troubles, which may increase the ability to estimate the good structures. Changing learning rate α also may stabilize the learning by updating Q values appropriately in order to improve the performance of Fuzzy GNP-TSRL (BS).

Thus, the objective of Fuzzy GNP-TSRL (BS) with changing ϵ-greedy policy and learning rate α is to improve the adaptability more when the changes of environments occur.

4.3 Mechanism of Changing Parameters

In chapter 3, Fuzzy GNP-TSRL (BS) enhanced the performances of Fuzzy GNP-RL when sudden changes occur in the environments. In order to improve the performances of Fuzzy GNP-TSRL (BS) more, the parameters of ϵ-greedy policy and learning rate α are changed during the training phase and these performances are confirmed in the implementation phase.

4.3.1 Changing ϵ-greedy Policy and Learning Rate α in the Training Phase

Fuzzy GNP-TSRL (BS) uses the SS method and BS method as explained in sub-section 3.3.2. In order to make an effective and efficient learning of the SS method and BS method, the exploration ability and exploitation ability should be controlled to make the balance, that is, the exploration should be carried out, on the other hand the good actions also should be exploited. Therefore, ϵ-greedy is set at a large value (ϵ_{ev}^{max}) at the first generation to make large explorations of the state space for learning the environments. After some generations, the random selection should be decreased until ϵ_{ev}^{min} generation by generation to reinforce good actions more. In addition, learning rate α is also set at a large value (α_{ev}^{max}) at the first generation to speed up the learning and is decreased until α_{ev}^{min} in later generations to stabilize the learning. The changes of ϵ-greedy policy and learning rate α are shown in Fig. 4.1, that is, they are controlled by Eq. 4.1 and Eq. 4.2.

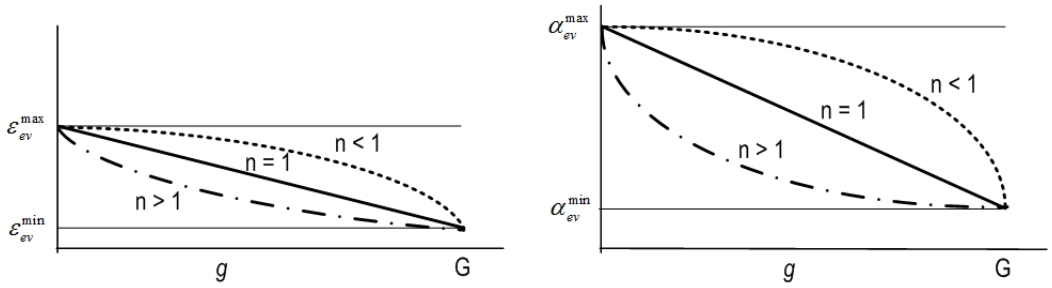


Figure 4.1: Changing of ϵ-greedy policy and learning rate α generation by generation

$$\epsilon_{ev} = \left(\epsilon_{ev}^{max} - \epsilon_{ev}^{min}\right)\left(1 - \frac{g}{G}\right)^{n} + \epsilon_{ev}^{min}, \tag{4.1}$$

$$\alpha_{ev} = \left(\alpha_{ev}^{max} - \alpha_{ev}^{min}\right)\left(1 - \frac{g}{G}\right)^{n} + \alpha_{ev}^{min}, \tag{4.2}$$

where, ϵ_{ev}^{max} and ϵ_{ev}^{min} are the upper and lower bound of ϵ-greedy policy in the evolution phase, respectively, while α_{ev}^{max} and α_{ev}^{min} are the upper and lower bound of learning rate α in the evolution phase, respectively. n is a constant for changing ϵ-greedy policy and learning rate α, where if $n < 1$ means that more explorations are carried out, while $n > 1$ means less explorations and $n = 1$ means the gradual decrease in the case of the changing ϵ-greedy policy. g is the current generation and G is the maximum number of generations. In this case, the parameter values are determined by experiments.

4.3.2 Changing ϵ-greedy Policy and Learning Rate α in the Implementation Phase

The adaptability of Fuzzy GNP-TSRL (BS) with changing ϵ-greedy policy and learning rate α in the training phase is studied by implementing the selected individuals in two cases.

1. In the first case, the individuals are implemented during their live time (T) using fixed ϵ-greedy policy and learning rate α. The aim is to study the impact of changing ϵ-greedy policy and learning rate α in the implementation phase.

2. In the second case, the individuals are implemented with changing ϵ-greedy policy and learning rate α during their live time (T). Here, the improvement of the on-line learning ability in the dynamical changing environments is studied, where the changes of the environments are carried out by setting sensors to break at time step t'. When the environments are suddenly changed, the individuals should explore the environments to find the better reward by increasing the exploration ability and reinforce the good actions after several time steps. The changing ϵ-greedy policy and learning rate α are controlled by the following

$$\epsilon_t = \left(\epsilon_t^{max} - \epsilon_t^{min}\right)\left(1 - \frac{t - t'}{T - t'}\right)^n + \epsilon_t^{min}, \tag{4.3}$$

$$\alpha_t = \left(\alpha_t^{max} - \alpha_t^{min}\right)\left(1 - \frac{t - t'}{T - t'}\right)^n + \alpha_t^{min}, \tag{4.4}$$

   where, ϵ_t^{max} and ϵ_t^{min} are the upper and lower bound of ϵ-greedy policy in each time step, respectively, while α_t^{max} and α_t^{min} are the upper and lower bound of learning rate α, respectively. t, t', T are the current time step, the time step when sensors break and the life time step during the implementation, respectively. n is a constant for changing ϵ-greedy policy and learning rate α, where if $n < 1$ means that more explorations are carried out, while $n > 1$ means less explorations and $n = 1$ means that the gradual decrease in the case of the changing ϵ-greedy policy. The parameter values are determined by experiments.

In the second case, changing ϵ-greedy policy and learning rate α is carried out when the changes of the environments occur [73]. In order to make the agent quickly adapts to the new situations caused by the changes of the environments, larger ϵ-greedy policy and learning rate α are used at the first and they are decreased after several time steps. The parameters of ϵ-greedy policy and learning rate α are carried out as Fig. 4.2.

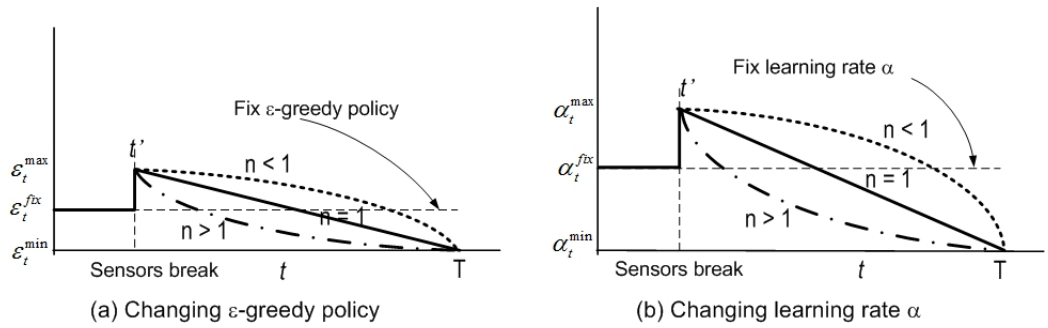


Figure 4.2: Changing of ϵ-greedy policy and learning rate α during the life time (T)

4.4 Comparison between Fuzzy GNP-TSRL (BS) with Changing and with Fixed ϵ-greedy Policy and Learning Rate α

Fuzzy GNP-TSRL (BS) is the Two-Stage RL combining the SS method and BS method, where the recovery process can be implemented for severe troubles. However, the size of Q tables of Fuzzy GNP-TSRL (BS) is larger than Fuzzy GNP-RL, then a mechanism to balance the exploration and exploitation improves the performance of the Two-Stage RL.

Fuzzy GNP-TSRL (BS) with changing ϵ-greedy policy and learning rate α makes larger explorations in early generations, but makes more exploitations in later generations. Thus, balancing the exploration and exploitation could enhance the performances of GNP-TSRL (BS) trained with fixed ϵ-greedy policy and learning rate α. In addition, the enhancement can also be done with changing ϵ-greedy policy and learning rate α in the implementation, which can quickly adapt to the changes of the environments.

On the other hand, Fuzzy GNP-TSRL (BS) with fixed ϵ-greedy policy and learning rate α is used, the exploration and exploitation are carried out by fixed parameters.

4.5 Simulations

Changing ϵ-greedy policy and learning rate α of Fuzzy GNP-TSRL (BS) is studied in the benchmark of the wall following behaviors of a Khepera robot using Webots simulator [71], where the specification of the robot and calculation of the reward have been introduced in section 2.5.1 and section 2.5.2, respectively.

4.5.1 Simulation Conditions

The effects of Fuzzy GNP-TSRL (BS) with changing ϵ-greedy policy and learning rate α are studied comparing with that with fixed parameters in two aspects, that is, in the training phase and implementation phase using the environments shown in Fig. 2.7. The best individual in the last generation is selected for the implementation.

Simulation in the Training Phase

In the training phase, 300 individuals are evolved at the end of each generation, 300 individuals are generated to form a new population for the next generation; 179 individuals are generated by mutation, 120 individuals are generated by crossover, and one individual is the elite. Each individual uses 61 nodes including 40 Fuzzy judgment nodes (5 for each kind), 20 processing nodes (10 for each kind) and one start node. Each of the Fuzzy judgment nodes and processing nodes of Fuzzy GNP-TSRL (BS) has 2 sub nodes, and each node connection of the sub nodes has 2 branch connections determined by the evolution.

In order to study the improvement of Fuzzy GNP-TSRL (BS) with changing parameters, 3 cases are done in the training phase, that is,

- **Case 1**. The individuals of Fuzzy GNP-TSRL (BS) are evaluated and evolved generation by generation using changing ϵ-greedy policy and learning rate α using the curves shown in Fig. 4.1 (it is defined as *TSRLch*). The individuals learn

the environments using changing ϵ-greedy policy and learning rate α which start from $\epsilon_{ev}^{max} = 0.15$ and $\alpha_{ev}^{max} = 0.70$ and gradually decrease to $\epsilon_{ev}^{min} = 0.01$ and $\alpha_{ev}^{min} = 0.1$. Here, parameter n is simulated to find the best performance.

- **Case 2**. The individuals of Fuzzy GNP-TSRL (BS) are evaluated and evolved generation by generation using fixed ϵ-greedy policy and learning rate α of the minimum values, that is, ϵ_{ev} = 0.01 and α_{ev} =0.10. (it is defined as $TSRLco(A)$).

- **Case 3**. The individuals of Fuzzy GNP-TSRL (BS) are evaluated and evolved generation by generation using fixed ϵ-greedy policy and learning rate α of the maximum values, that is, ϵ_{ev} = 0.15 and α_{ev} =0.70. (it is defined as $TSRLco(B)$).

The maximum and minimum values of ϵ-greedy policy and learning rate α are selected appropriately through the simulations as shown in Table 4.1.

Table 4.1: Simulation conditions of Fuzzy GNP-TSRL with changing parameters comparing with fixed parameters

	TSRLch	TSRLco (A)	TSRLco (B)
The number of individuals	(300) crossover: 120, mutation: 179, elite: 1		
The number of nodes	(61) 20 processing nodes, 40 Fuzzy judgment nodes, 1 start node		
The number of sub nodes	2 for each judgment and processing node		
The number of branch connections	2 for each connection		
Parameter of evolution	P_c = 0.1, P_m = 0.01, tournament sizes = 7		
(Training phase) Parameter of learning	γ = 0.9, ϵ_{ev}^{max} = 0.15, ϵ_{ev}^{min} = 0.01 α_{ev}^{max} = 0.70 , α_{ev}^{min} = 0.10	γ = 0.9, ϵ_{ev} = 0.01 α_{ev} = 0.10	γ = 0.9 ϵ_{ev} = 0.15, α_{ev} = 0.70,
(Implementation phase) Fixed Parameter of learning	γ = 0.9, ϵ_t^{fix} = 0.010 α_t^{fix} = 0.100	γ = 0.9, ϵ_t^{fix} = 0.010 α_t^{fix} = 0.100	γ = 0.9 ϵ_t^{fix} = 0.010, α_t^{fix} = 0.100,
(Implementation phase) Changing Parameter of learning	ϵ_t^{max} = 0.015, ϵ_t^{min} = 0.000 α_t^{max} = 0.150, α_t^{min} = 0.000		

Simulation in the Implementation Phase

The effects of changing ϵ-greedy policy and learning rate α in the training phase are also studied in the implementation phase in two cases, that is,

- **Case 1**. The implementation is carried out using fixed values of ϵ_t = 0.010 and α_t = 0.100 (*TSRL/TESTco*). In this case, the improvement of the adaptability us-

Table 4.2: Performances of *TSRLch* with various parameter n

n	Training Phase Fitness at the 1000th generation	Implementation Phase Average Reward of the time step 500th to 3000th
10	0.521	0.131
3	0.555	0.156
1	**0.593**	**0.158**
1/3	0.545	0.147
1/10	0.531	0.142

ing changing ϵ-greedy policy and learning rate α during the training phase is studied.

- **Case 2**. The implementation is carried out using changing values to study the adaptability of on-line learning. From the 1st to 500th time steps, the individuals are implemented with fixed parameters, i.e., $\epsilon_t^{fix} = 0.010$ and $\alpha_t^{fix} = 0.100$, and after the sudden changes occur, the values are set higher at $\epsilon_t^{max} = 0.015$ and $\alpha_t^{max} = 0.150$ and gradually decreased until the end of the life time to values of $\epsilon_t^{min} = 0$ and $\alpha_t^{min} = 0$ (*TSRL/TESTch*) as shown in Fig. 4.2. Here, parameter n is also simulated to find the best performance.

In order to study the adaptability in the implementation phase, the sudden changes are carried out by making sensors break at the 500th time step, where the life time of each individual is 3000 time steps. The average reward of each method is calculated after the sensors' break, where the implementations are done 3000 times, that is, 10 best individuals from 10 independent are implemented 300 times with 10 different start positions of the robot.

4.5.2 Simulation Results

Before studying the effects of changing ϵ-greedy policy and learning rate α on Fuzzy GNP-TSRL (BS), parameter n is studied to find the best performance. Table 4.2 shows the performance of Fuzzy GNP-TSRL (BS) with changing ϵ-greedy policy and learning rate α (*TSRLch*), which is simulated with various parameter n.

From Table 4.2, $n = 1$ shows the best performance in both of the training phase and implementation phase. Then, this result is used for studying the effectiveness of Fuzzy GNP-TSRL (BS) with changing parameters of ϵ-greedy policy and learning rate α.

Simulation Results in the Training Phase

Firstly, the individuals of GNP-TSRL (BS) are trained with changing ϵ_{ev}-greedy policy and learning rate α_{ev} (*TSRLch*) and compared with the individuals which are trained with fixed parameters, i.e., $TSRLco(A)$ and $TSRLco(B)$ as explained in section 4.5.1. The average fitnesses of these simulations are shown in Fig. 4.3, where each curve is averaged over 10 best individuals of 10 independent training simulations.

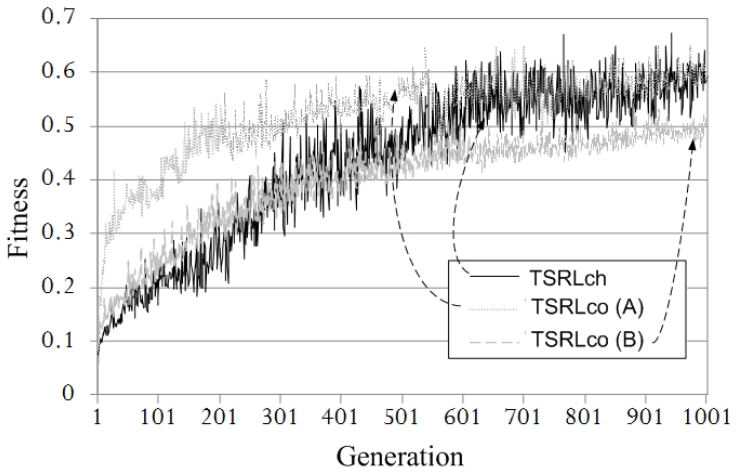


Figure 4.3: Average fitness of *TSRLch*, *TSRLco*(A) and *TSRLco*(B) in the training phase

The average fitness of *TSRLco*(A) converges faster and higher than that of *TSRLco*(B). Small explorations are carried out by *TSRLco*(A), then the good actions can be reinforced well. On the other hand, large explorations are carried out by *TSRLco*(B), then the good actions are more difficult to be reinforced. In other words, *TSRLco*(A) and *TSRLco*(B) carry out smaller and larger explorations, respectively. If the random action selection is carried out with higher probability, the good actions cannot be reinforced well, then the fitnesses are small. On the other hand, If the exploration of action selections is carried out with lower probability, the good actions can be reinforced well, but the alternative actions cannot be reinforced well.

In the case of *TSRLch*, the curve of the average fitness converges slower in earlier generations, because random action selections are carried out with the high probability. However, when the random action selection is decreased in later generations, the good actions can be reinforced well, so that the fitness becomes as large as *TSRLco*(A).

The simulation results confirm that *TSRLch* is trained more efficiently and effectively than *TSRLco*(A) and *TSRLco*(B).

Simulation Results in the Implementation Phase

In the implementation phase, the adaptability of Fuzzy GNP-TSRL (BS) is studied in two cases, i.e., (1) implementation with fixed parameters of ϵ_t-greedy policy and learning rate α_t (*TSRL/TESTco*); and (2) implementation with changing parameters (*TSRL/TESTch*) as shown in Table 4.1.

Case 1. The individuals which are trained by *TSRLco*(A), *TSRLco*(B) and *TSRLch*, are implemented in *TSRL/TESTco*, i.e., $\epsilon_t^{fix} = 0.01$ and $\alpha_t^{fix} = 0.10$. When the troubles occur in the implementation phase, the action selections are influenced by incorrect information, then the agent behaviors are determined inappropriately and the Q values are decreased, then the alternative actions are selected more frequently due to the changes of the environments. The average rewards of the implementation with fixed ϵ-greedy policy and learning rate α (TSRL/TESTco) are shown in Table 4.3.

1. *TSRLco*(A) has the lowest average reward as shown in Table 4.3, because it was trained with small explorations, the Q values of the alternative actions are not

Table 4.3: Average reward of *TSRLch*, *TSRLco*(A) and *TSRLco*(B) implemented under *TSRL/TESTco*

Individual	Average	Stdev	T-test one tail
TSRLch	0.158	0.036	-
TSRLco(A)	0.070	0.030	7.25E-06
TSRLco(B)	0.112	0.031	3.97E-03

explored well. Thus, under the changes of the environments in the implementation, the actions of *TSRLco*(A) cannot be selected appropriately.

2. *TSRLco*(B) was trained with large explorations, although its performance is worse than *TSRLco*(A) in the training phase, the average reward of *TSRLco*(B) is higher than *TSRLco*(A) in the implementation phase, because the alternative actions are explored more than *TSRLco*(A). Thus, under the changes of the environments in the implementation, the alternative actions of *TSRLco*(B) can be selected more appropriately.

3. *TSRLch* has the highest average reward, which means that changing ϵ_{ev}-greedy policy and learning rate α_{ev} can reinforce both of the good actions and alternative actions more efficiently and effectively comparing with *TSRLco*(A) and *TSRLco*(B) in the training phase. Thus, when troubles occur in the implementation, *TSRLch* can determine the actions more appropriately, thus the average reward is the highest.

Case 2. The individuals which are trained by *TSRLco*(A), *TSRLco*(B) and *TSRLch*, are implemented in *TSRL/TESTch* using parameters as shown in Table 4.1. In this case, after sudden changes occur, large random action selections are carried out in order to quickly adapt to the new environment and then, the action selections are gradually decreased in order to reinforce the good actions more appropriately. Using this mechanism, *TSRLco*(A), *TSRLco*(B) and *TSRLch* have higher average rewards when they are implemented in *TSRL/TESTch* compared with *TSRL/TESTco* as shown in Fig. 4.4. It shows that the on-line learning with changing ϵ_t-greedy policy and learning rate α_t improves the adaptability. Furthermore, *TSRLch* shows higher average reward than *TSRLco*(A) and *TSRLco*(B) as shown in Table 4.4. It is because *TSRLch* can reinforce both the good actions and alternative actions in the training phase and the on-line learning of *TSRL/TESTch* can make the balance between the exploration and exploitation to determine the good actions for the implementation.

When severe troubles occur, such as large number of sensors' break, then incorrect information becomes larger. Here, the appropriate behaviors are more difficult to be determined. Therefore, the average reward also becomes smaller. The average rewards of Fuzzy GNP-TSRL (BS) with changing ϵ-greedy and learning rate α show the superiority than those with fixed values as shown in Fig. 4.5. It confirms that changing ϵ-greedy and learning rate α in the training phase and implementation phase can improve the adaptability of Fuzzy GNP-TSRL (BS).

Table 4.4: Average reward of *TSRLch*, *TSRLco*(A), *TSRLco*(B) implemented under *TSRL/TESTch*

Individual	Average	Stdev	T-test one tail
TSRLch	0.170	0.035	-
TSRLco(A)	0.083	0.035	1.46E-05
TSRLco(B)	0.115	0.026	6.89E-04

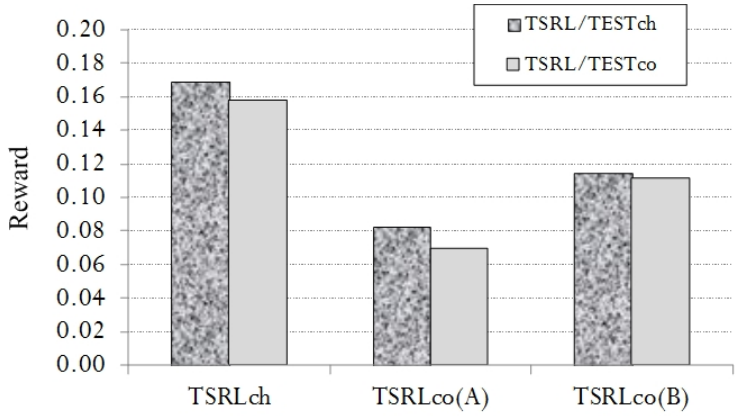


Figure 4.4: Average fitness of *TSRLch*, *TSRLco*(A) and *TSRLco*(B) in the implementation phase

4.6 Summary

Changing ϵ-greedy policy and learning rate α has been proposed to improve the adaptability of Fuzzy GNP-TSRL (BS).

1. In the training phase, the high explorations are carried out to determine the random action selections in early generations, while the explorations are decreased in later generations to reinforce the good actions more. The changing parameters make the balance between the exploration and exploitation, then the actions can be reinforced well to improve the adaptability of Fuzzy GNP-TSRL (BS).

2. In the implementation phase, the on-line learning with changing ϵ_t-greedy policy and learning rate α_t can improve the adaptability of Fuzzy GNP-TSRL (BS) more, when the changes of the environments occur.

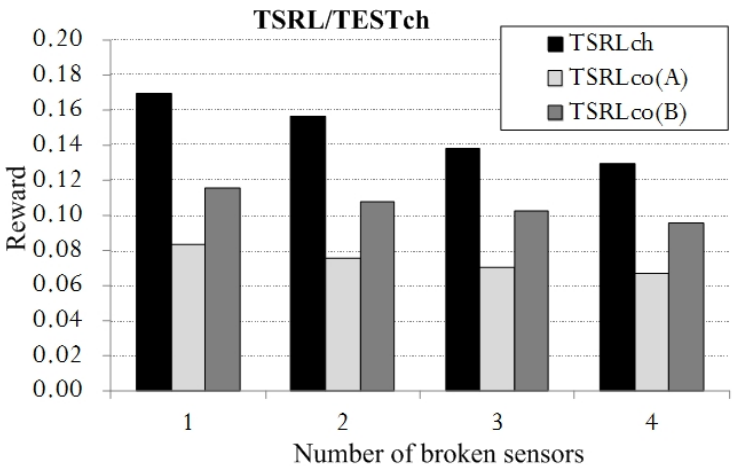


Figure 4.5: Average fitness of *TSRLch* with several broken sensors in the implementation phase

Chapter 5

Fuzzy GNP-TSRL (CS)

5.1 Introduction

In order to deal with severe problems caused by broken sensors in the implementation, Two-Stage Reinforcement Learning of Fuzzy GNP-TSRL (BS) shows better performance than One-Stage Reinforcement Learning of Fuzzy GNP-RL. Fuzzy GNP-TSRL (BS) has the ability to change the programs adaptively using alternative sub nodes/functions and also alternative connections. Fuzzy GNP-TSRL (BS) increases the search ability in the graph structures, where the next nodes of the node transitions are also learned, then, the recovery process of Fuzzy GNP-TSRL (BS) is faster than Fuzzy GNP-RL. Although Fuzzy GNP-TSRL (BS) has larger connections, which means that it can make various node transitions and may reuse the same nodes many times in the node transitions. Here, if the nodes which create inappropriate behaviors are executed many times, then the performance of Fuzzy GNP-TSRL (BS) would be decreased. On the other hand, increasing the branch connections make the size of Q values larger than Fuzzy GNP-RL if the same number of sub nodes is used. Then, the Q values take longer times to converge. Thus, instead of using the Branch connection Selection method (BS method), another method of the Credit branch Selection method (CS) of Credit-GNP is considered in this chapter.

Another version of enhancing GNP, that is, Credit-GNP has been proposed [69], where credit branch is implemented to GNP with rules [70] to generate the association rules and store in the rule pool during the training phase. Then, Credit-GNP can store more various rules in the rule pool and improves the generalization ability in the dynamic environments.

The idea of the CS method is to skip harmful nodes and it is combined with Fuzzy GNP-RL in this chapter. Thus, instead of using the Branch connection Selection method of Fuzzy GNP-TSRL (BS), skipping harmful nodes which create inappropriate behaviors enhances the problems of Fuzzy GNP-TSRL (BS) and improve the adaptability of Fuzzy GNP-RL.

5.2 Motivation of GNP-TSRL (CS)

Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL have advantages to change programs adaptively by changing sub nodes and/or selecting branch connections. In the chapter 3, Fuzzy GNP-TSRL (BS) can improve the adaptability of Fuzzy GNP-RL when inexpe-

rienced sudden changes occur in the implementation. While the conventional Fuzzy GNP-RL proposed a recovery method by changing sub nodes when the node is visited again by the node transitions. It takes a longer time, therefore, Fuzzy GNP-TSRL (BS) was proposed by changing connections beside of changing sub nodes. In this case, the next nodes of Fuzzy GNP-TSRL (BS) are also learned to determine the node transitions, therefore, Fuzzy GNP-TSRL (BS) has better performances than Fuzzy GNP-RL.

Both of Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL have advantages to recover the problems when the nodes are visited again in the next time step. However, this mechanism also creates problems when the nodes visited many times are harmful nodes. Thus, the performance may be decreased. In order to solve this problem, skipping harmful nodes is the simplest way for the recovering process. Thus, Fuzzy GNP-TSRL with Credit branch Selection (Fuzzy GNP-TSRL (CS)) is proposed to improve the adaptability of Fuzzy GNP-RL, when the changes of the environments occur. Fuzzy GNP-TSRL (CS) combines the Sub node Selection method (SS) of the Fuzzy GNP-RL and Credit branch Selection method (CS), where these methods are carried using Two-Stage Reinforcement Learning.

5.3 Algorithm of Fuzzy GNP-TSRL (CS)

In order to realize Fuzzy GNP-TSRL (CS), the judgment nodes and processing nodes are modified. Instead of using Branch connection Selection method (BS method) as described in chapter 3, Fuzzy GNP-TSRL (CS) combines the SS method to determine the appropriate functions and Credit branch Selection method (CS method) to skip harmful nodes. The node structures of the proposed method are described in sub-section 5.3.1. The mechanisms of determining the node transitions and learning process are described in sub-section 5.3.2. In addition, the recovery process is explained in sub section 5.3.3 to show the advantages of the proposed.

5.3.1 Structure Representation of GNP-TSRL (CS)

The basic structure of Fuzzy GNP-TSRL (CS) is the same as the previous chapter shown in Fig. 2.1 as explained in section 2.3.1. In order to realize the combination of the SS method and CS method, the node structures of Fuzzy GNP-TSRL (CS) are revised as Fig. 5.1, where two-kind Q tables are used, that is, Q_{SS}-table and Q_{CS}-table. Therefore, the gene of Fuzzy GNP-TSRL (CS) is modified as shown in Fig. 5.2.

It is supposed that node $i \in \{0, 1, ..., n-1\}$ has m sub nodes. The gene structure of node i is divided into macro node part, sub node part and branch part.

Macro Node Part

The macro node part of node i is defined by NT_i and d_i. Where NT_i represents a node type and d_i represents the time delay spent on executing node i, where NT_i and d_i are shown in Table 2.1.

Sub Node Part

The sub node part describes the function of the sub nodes. The function of sub node $p \in \{1, ..., m\}$ of node i is defined by ID_{ip}, a_{ip} and $Q_{SS}(i, ip)$, as explained in sub-section

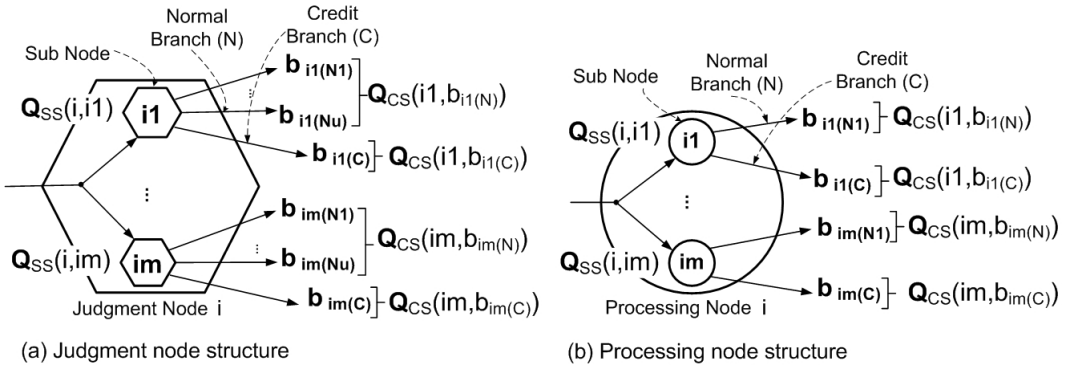


Figure 5.1: Node structures of GNP-TSRL (CS)

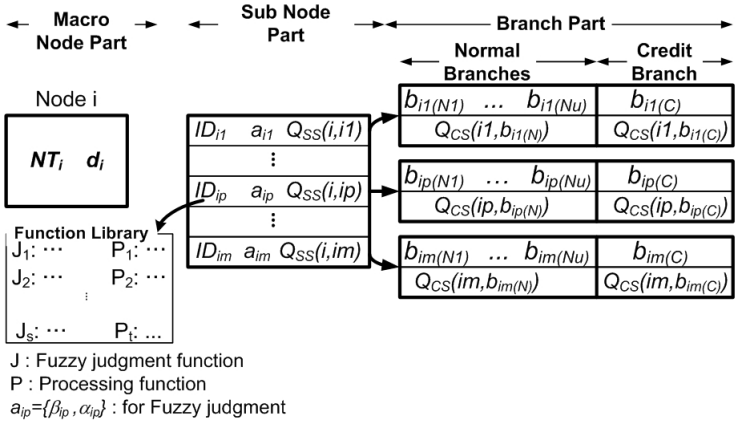


Figure 5.2: Gene of GNP-TSRL (CS)

2.3.1. In Fuzzy GNP-TSRL (CS), the function which should be executed is determined by ϵ-greedy policy according to $Q_{SS}(i, ip)$ value by the SS method at the first stage of RL. Here, the state is the current node i and the action is sub node selection ip, that is, sub node p of node i.

Branch Part

The different point of Fuzzy GNP-TSRL (CS) is the branch part. While in Fuzzy GNP-TSRL (BS), each connection of sub node p of the nodes has several branch connections, the connections of Fuzzy GNP-TSRL (CS) are grouped into the normal branches and credit branch as follows.

- **Normal branches.** When the node is judgment node i, sub node p has u normal branches, i.e., $\{b_{ip(N1)}, ..., b_{ip(Nu)}\}$. In this chapter, the number of the normal branches is two to represent branch connections as described in Fuzzy GNP-RL, where a branch of the normal is selected by the judgment result like Fuzzy GNP-RL explained in section 2.3.2. However, when the node is a processing node, the sub node has only one normal branch, i.e., $b_{ip(N1)}$. The normal branches of sub node p of node i are grouped by a Q value, i.e., $Q_{CS}(ip, b_{ip(N)})$
- **Credit branch.** Each sub node of node i has a credit branch, i.e., $b_{ip(C)}$ and the Q value of the credit branch is represented by $Q_{CS}(ip, b_{ip(C)})$.

The selection of the normal branches or credit branch is determined by ϵ-greedy policy according to $Q_{CS}(ip, b_{ip(N)})$ and $Q_{CS}(ip, b_{ip(C)})$. If $Q_{CS}(ip, b_{ip(N)})$ is selected, then the function of sub node p of node i is executed as a normal node and the next node is determined by a normal branch $\in \{b_{ip(N1)}, ..., b_{ip(Nu)}\}$ as Fuzzy GNP-RL. However, while $Q_{CS}(ip, b_{ip(C)})$ is selected, then the function of the node is not executed and it is considered as a harmful node, that is, the node is skipped and the node transition is transfered to next node shown by $b_{ip(C)}$.

5.3.2 Learning Process of Fuzzy GNP-TSRL (CS)

Fuzzy GNP-TSRL (CS) is proposed to improve the adaptability of Fuzzy GNP-RL when sudden changes occur, where the Two-Stage Reinforcement Learning is used, that is, the Sub node Selection method (SS method) and Credit branch Selection method (CS method) are learned at the first stage and second stage, respectively. Here, two kinds of Q tables are used, i.e., Q_{SS} table and Q_{CS} table.

SS Method. The SS method is carried out at the first stage of Fuzzy GNP-TSRL (CS) and done like Fuzzy GNP-RL . The state is represented by the current node i, while the action is the selection of sub node p of node i which is determined based on ϵ-greedy policy according to the Q_{SS} value.

CS Method. The CS method is carried out at the second stage of Fuzzy GNP-TSRL (CS) to select a group of normal branches or a credit branch. The CS method is done based on ϵ-greedy policy according to the Q_{CS} values. In this case, the state is represented by sub node p of node i, which is selected at the first stage, while the action is a group of normal branches of $\{b_{ip(N1)}, ..., b_{ip(Nu)}\}$ or credit branch $b_{ip(C)}$ which is selected by ϵ-greedy policy according to the $Q_{CS}(ip, b_{ip(N)})$ value or $Q_{CS}(ip, b_{ip(C)})$ value.

Node Transitions of Fuzzy GNP-TSRL (CS)

The mechanism of skipping harmful nodes is explained using an example of node transitions shown in Fig. 5.3.

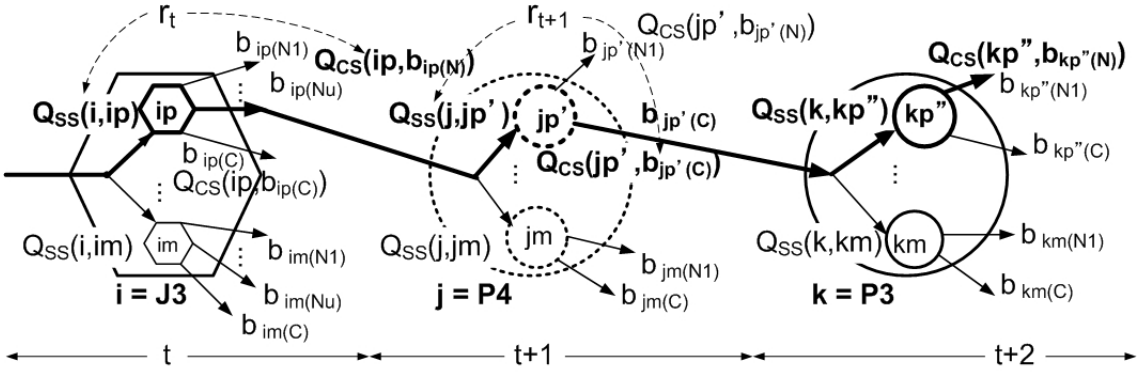


Figure 5.3: An example of a node transition of GNP-TSRL (CS)

- It is supposed that the node transitions are $J3 \rightarrow P4 \rightarrow P3$. If the current node i is judgment node $J3$, this node judges sensor ip selected by ϵ-greedy policy according to $Q_{SS}(i, ip)$.
- Then, it is supposed that a group of normal branches are also selected by ϵ-greedy policy according to $Q_{CS}(ip, b_{ip(N)})$, then $J3$ works like Fuzzy GNP-RL, that is, J3 is executed as a normal judgment node and returns the judgment result, for example normal branch $b_{ip(Nu)}$ is selected.
- After that, the current node is transferred to $P4$, where ϵ-greedy policy selects sub node jp' according to $Q_{SS}(j, jp')$ and credit branch $b_{jp'(C)}$ is selected according to $Q_{CS}(jp', b_{jp'(C)})$. In this case, $P4$ is skipped which means that the agent behavior is not changed and the current node is transferred to $P3$.
- If sub node kp'' of $P3$ and normal branch $b_{kp''(N1)}$ are selected, then the agent behavior is changed by executing function kp''.

Sarsa Learning of Fuzzy GNP-TSRL (CS)

The procedure of updating Fuzzy GNP-TSRL (CS) using Sarsa learning is explained as follows.

1. At time t, it is supposed that the current state is node i, and Fuzzy GNP-TSRL (CS) refers to all Q_{SS} values for the SS method, i.e., $\{Q_{SS}(i, i1), ..., Q_{SS}(i, im)\}$, and selects one of them based on ϵ-greedy policy, that is, the sub node which has the max Q value is selected by the probability of $1 - \epsilon$, while another sub node can be selected randomly by the probability of ϵ. It is supposed that GNP selects $Q_{SS}(i, ip) \in \{Q_{SS}(i, i1), ..., Q_{SS}(i, im)\}$, and the corresponding node function ID_{ip} and parameter a_{ip} are determined.

2. At the next step, Fuzzy GNP-TSRL (CS) checks the conditions of the next node to be connected considering the Q values of the normal branches, i.e., $Q_{CS}(ip, b_{ip(N)})$ and credit branch, i.e., $Q_{CS}(ip, b_{ip(C)})$, which are selected based on ϵ-greedy policy as the second stage of Fuzzy GNP-TSRL (CS). Max Q value between $Q_{CS}(ip, b_{ip(N)})$ and $Q_{CS}(ip, b_{ip(C)})$ is selected by the probability of $1 - \epsilon$, or random one by the probability of ϵ.

   (a) if the normal branches are selected according to $Q_{CS}(ip, b_{ip(N)})$, then Fuzzy GNP-TSRL (CS) executes function ID_{ip} and determines the next node connection by $b_{ip(N)} \in \{b_{ip(N1)}, ..., b_{ip(Nu)}\}$.
   (b) else, the credit branch is selected, then Fuzzy GNP-TSRL (CS) considers that current node i is harmful, so its function is not executed and the next connection is $b_{ip(C)}$.

   At this step, it is supposed that normal branch $b_{ip(Nu)}$ is selected, then the next node is node j as shown in Fig. 5.3

3. After taking the action at step 2(a), the reward r_t is given. However, when the credit branch (step 2(b))is selected as the action, the reward r_t should be 0 because the node is skipped.

4. At time $t + 1$, Fuzzy GNP-TSRL (CS) repeats step 1 to 3 for the next node j. Here, it is supposed that the SS method determines $Q_{SS}(j, jp')$ and the CS method determines $Q_{CS}(jp', b_{jp'(C)})$.

5. Then, the Q value of the SS method in the first stage of RL is updated by the following procedure,

$$\begin{aligned} &Q_{SS}(i, ip) \leftarrow Q_{SS}(i, ip) \\ &+\alpha(r_t + \gamma Q_{SS}(j, jp') - Q_{SS}(i, ip)). \end{aligned} \tag{5.1}$$

   The Q value of the CS method in the second stage of RL is updated as follows.

   When a normal branch is selected in step (2.a),

$$\begin{aligned} &Q_{CS}(ip, b_{ip(N)}) \leftarrow Q_{CS}(ip, b_{ip(N)}) + \\ &\alpha(r_{t+1} + \gamma Q_{CS}(jp', b_{jp'(N)}) - \\ &Q_{CS}(ip, b_{(ip,N)})). \end{aligned} \tag{5.2}$$

   When a credit branch is selected in step (2.b),

$$\begin{aligned} &Q_{CS}(ip, b_{ip(C)}) \leftarrow Q_{CS}(ip, b_{ip(C)}) + \\ &\alpha(\gamma Q_{CS}(jp', b_{jp'(N)}) - \\ &Q_{CS}(ip, b_{ip(C)})). \end{aligned} \tag{5.3}$$

   Where, α is a learning rate ($0 < \alpha \leq 1$) and γ is discount rate ($0 \leq \gamma \leq 1$).

6. $t \leftarrow t + 1$, $i \leftarrow j$ and $p \leftarrow p'$. Then, return to step 1.

5.3.3 Recovery Process of Fuzzy GNP-TSRL (CS)

Fuzzy GNP-TSRL (BS) enhanced the recovery process of Fuzzy GNP-RL by selecting the appropriate next node which may change the agent behavior appropriately. However, this process still have a problem such as explained in section 5.2.

It is supposed that sub node i_1 in Fig. 5.4 is a harmful node and creates inappropriate behaviors. Then, executing sub node i_1 give a small average reward. In order to recover this problem, skipping sub node i_1 may increase the total average reward as shown in Fig. 5.4.

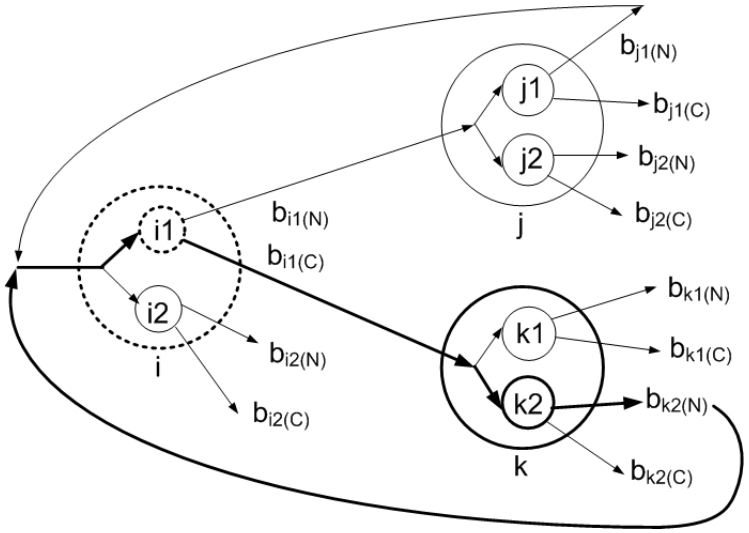


Figure 5.4: An example of recovery process of Fuzzy GNP-TSRL (CS)

Recovery Process by Changing the Function

If the current node is node i, sub node $i1$ is determined based on ϵ-greedy policy according to $Q_{SS}(i, i1)$. As the conventional method, when the function creates inappropriate behavior, alternative function $i2$ may be selected in the next time when node i is visited again.

Recovery Process by Skipping the Node

The new point in Fuzzy GNP-TSRL (CS) is to skip harmful nodes by using credit branch. As shown in Fig. 5.1, each sub node has the normal branches and credit branch, which are selected based on ϵ-greedy policy. It is supposed that a node transition is determined as shown in Fig. 5.4 and sub node $i1$ is selected. The recovery process can be described as follows.

- When normal branch $b_{i1(N)}$ is selected according to $Q_{CS}(i1, b_{i1(N)})$, then function $i1$ is executed as a normal normal node. However, if inappropriate behavior is created after executing this function, the $Q_{CS}(i1, b_{i1(N)})$ is updated smaller.
- When sub node $i1$ is visited again where $Q_{CS}(i1, b_{i1(C)})$ value is larger than $Q_{CS}(i1, b_{i1(N)})$, the credit branch is selected by the large probability. If credit branch $b_{i1(C)}$ is se-

lected according to $Q_{CS}(i1, b_{i1(C)})$, node i is considered as a harmful node and function $i1$ is skipped. Then, the agent behavior may be revised by the executing node k.

5.4 Comparison between Fuzzy GNP-TSRL (CS), Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL

Fuzzy GNP-TSRL (CS) is an enhancement of Fuzzy GNP-RL and revise the Fuzzy GNP-TSRL (BS), while Fuzzy GNP-RL uses One-Stage RL for the SS method, Fuzzy GNP-TSRL (BS) and Fuzzy GNP-TSRL (CS) uses Two-Stage RL, here Fuzzy GNP-TSRL (BS) combines the SS method and BS method to select branch connections, while GNP-TSRL (CS) combines the SS method and CS method to skip harmful nodes. When harmful nodes are executed many times in the node transitions, Fuzzy GNP-RL and Fuzzy GNP-TSRL (BS) cannot avoid these nodes, while Fuzzy GNP-TSRL (CS) has the ability to skip these nodes. Thus, the recovery process of GNP-TSRL (CS) can improve better than Fuzzy GNP-RL and Fuzzy GNP-TSRL (BS), when severe troubles occur, that is, Fuzzy GNP-TSRL (CS) can adaptively change the programs by changing the functions like Fuzzy GNP-RL or skip the functions when they are considered to create inappropriate behaviors. As a result, the adaptability of Fuzzy GNP-RL can be improved, especially when sensors break in the implementation.

5.5 Simulations

The effectiveness of the recovery process of Fuzzy GNP-TSRL (CS) is studied in the benchmark of the wall following behaviors of a Khepera robot using Webots simulator [71], where the specification of the robot and calculation of the reward have been introduced in section 2.5.1 and section 2.5.2, respectively.

5.5.1 Simulation Environments

The effectiveness of Fuzzy GNP-TSRL (CS) is studied in the training phase and in the implementation phase comparing with Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL by using the simulation environments as shown in Fig. 2.7. The simulations are carried out as follows.

1. **Simulation 1.** In the training phase, the individuals of Fuzzy GNP-TSRL (CS), Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL evolved generation by generation and are evaluated, where the functions, parameters and branches are changed by crossover and mutation. The life time for evaluation of each individual is 2000 time steps in each generation. In this simulation, Fuzzy logic is integrated to the judgment nodes and Gaussian noises ($\mu = 0, \sigma = 50$) are introduced to the sensor values as was done in Chapter 2. The generalization ability is confirmed by using 10 different start positions of the robot in the environment shown in Fig. 2.7(a), where the start positions are selected randomly by the individuals. Each individual is evaluated for 2000 time steps and the start position of the individual is changed randomly every 500 time steps. The individuals are evolved for 1000 generations and the fitness is evaluated generation by generation.

2. **Simulation 2.** In order to study the effectiveness of the recovery process of Fuzzy GNP-TSRL (CS), the best individuals from the training phase are implemented in the environments shown in Fig. 2.7(b), where the life time for the implementation of the individuals is 3000 time steps which are separated in two cases.
   - **Case 1.** The individuals are implemented in the normal situations, here, no sensor breaks in the implementation. In order to study the generalization ability of Fuzzy GNP-TSRL (CS), the individuals are implemented with 10 different start positions of the robot and repeated 300 times for each individual.
   - **Case 2.** The individuals are implemented with inexperienced changes of the environments which are carried out by making a sensor breaks at the 500th time step. The changes are set as follows, from the 1st time step to the 500th time step, the individuals are implemented in the normal situations (no sensor breaks); a sensor breaks at the 500th time step, after that the adaptability of Fuzzy GNP-TSRL (CS) to recover the troubles is analyzed.
3. **Simulation 3.** Furthermore, severe inexperienced troubles are set. In this case, the simulation is done like simulation 2, however, inexperienced sudden changes occur by making several sensors break at the 500th time step, that is, 2, 3, 4 and 5 sensors break. The performances of the proposed method for recovering the troubles are analyzed after the 500th time step.

5.5.2 Simulation Conditions

In order to confirm the advantages of Fuzzy GNP-TSRL (CS), the performance of Fuzzy GNP-TSRL (CS) is compared with Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL using parameter settings as shown in Table 5.1.

Firstly, the individuals are initialized to form a population, where the number of individuals is 300. Each individual has one start node and a fixed number of the judgment nodes and processing nodes, that is, 61 nodes including 40 Fuzzy judgment nodes (5 for each kind), 20 processing nodes (10 for each kind) and one start node. Each node has 2 sub nodes representing different functions, where each sub node of the judgment node has 2 kinds of connections, that is, normal branches and a credit branch, while that of the processing node has 1 normal branch and 1 credit branch. The function of node (ID_{ip}) is assigned by an unique number which is shown in the function library in Table 2.3. The parameter of node (a_{ip}) is set at random integers. When the node is a judgment node, its parameter is $a_{ip} = \{\beta_{ip}, \alpha_{ip}\}$, where α_{ip} is larger than β_{ip}; that is, α_{ip} is set between 0 and 1023, and β_{ip} is set between 0 and α_{ip}, however, when the node is a processing node, its parameter is set between -10 and 10. The initial connections between nodes are determined randomly in the graph structure. All Q values (Q_{SS} and Q_{CS}) are set at zero initially.

The learning parameters, i.e., learning rate $\alpha = 0.1$, discount rate $\gamma = 0.9$, and ϵ greedy policy parameters, i.e., $\epsilon_{SS} = 0.1$ and $\epsilon_{CS} = 0.1$ are given to consider the future rewards and to keep the balance between the exploration and exploitation. Gaussian noises ($\mu = 0, \sigma = 50$) are added to the sensor values in the training phase and implementation phase to improve the generalization ability of GNP in noisy environments.

Table 5.1: Simulation conditions

Fuzzy GNP	TSRL (CS)	TSRL (BS)	RL
The number of individuals	(300) mutation: 179, crossover: 120, elite: 1		
The number of nodes	(61) 20 processing nodes, 40 Fuzzy judgment nodes, and 1 start node		
The number of sub nodes	2	2	4
The number of branch connection for each connection of a sub node	-	2	-
The number of credit branch for each sub node of judgment nodes for each sub node of processing nodes	1 1	- -	- -
The kind of Q tables	Q_{SS} and Q_{CS}	Q_{SS} and Q_{BS}	Q_{SS}
Parameters of evolution	$P_c = 0.1$, $P_m = 0.01$, tournament sizes = 7		
Parameters of learning	$\alpha = 0.1$, $\gamma = 0.9$, $\epsilon_{SS} = 0.1$, $\epsilon_{CS} = 0.1$, $\epsilon_{BS} = 0.1$ (ϵ of greedy policy)		

5.5.3 Simulation Results

The effectiveness of Fuzzy GNP-TSRL (CS) is confirmed comparing with Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL in these simulation results.

Results of Simulation 1

Fig. 5.5 shows the average fitness of Fuzzy GNP-TSRL (CS) comparing with Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL, where each of the curves is averaged over 10 best individuals from 10 independent training simulations. The curve of Fuzzy GNP-TSRL (CS) converges faster and obtains higher fitness values than Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL. It shows that updating Q values of GNP-TSRL (CS) is done more efficiently and effectively than that of the other methods.

Fuzzy GNP-TSRL (CS) uses 2 kinds of Q tables and the actions in the nodes are grouped into two kinds, then the actions are determined more efficiently and effectively than only using one Q table like Fuzzy GNP-RL. Thus, the average fitness of Fuzzy GNP-TSRL (CS) and Fuzzy GNP-TSRL (BS) is higher than that of Fuzzy GNP-RL. Moreover, the average fitness of Fuzzy GNP-TSRL (CS) is still higher than that of Fuzzy GNP-TSRL (BS), even both of them use two Q tables. In this case, the CS method of Fuzzy GNP-TSRL (CS) is carried out more efficiently and effectively, because the normal branches are grouped into one Q_{CS}, while each branch of Fuzzy GNP-TSRL (BS) has their own Q_{BS}. In addition, when all of the branches in the nodes of Fuzzy

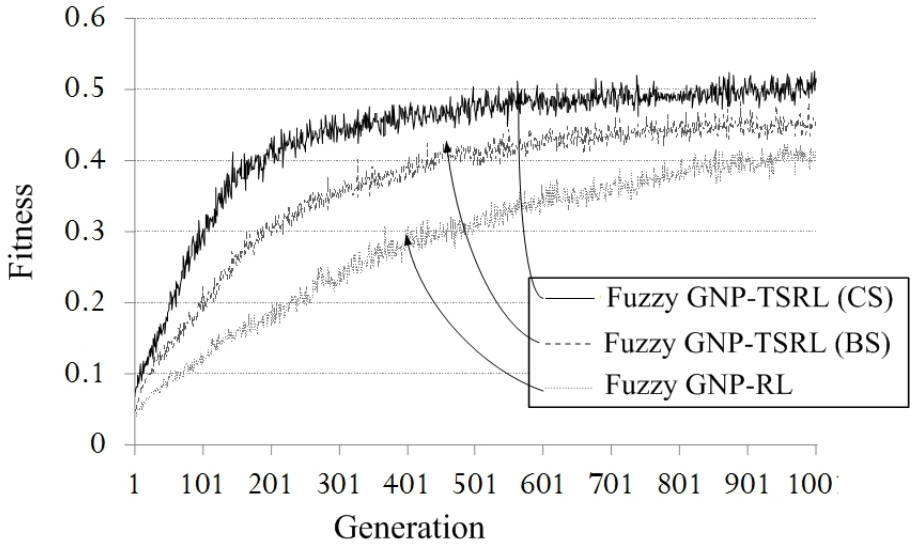


Figure 5.5: Average fitness in the training phase

GNP-TSRL (BS) fail to determine good connections, the fitness of Fuzzy GNP-TSRL (BS) is decreased, while Fuzzy GNP-TSRL (CS) can skip this harmful nodes by selecting the credit branch. Thus, the average fitness of Fuzzy GNP-TSRL (CS) shows the highest value among these three methods.

Results of Simulation 2

The adaptability of Fuzzy GNP-TSRL (CS) is studied in this section, when inexperienced sudden changes occur, where the efficiency and effectiveness of the adaptation mechanisms of Fuzzy GNP-TSRL (CS) are compared with those of Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL. In this case, 10 best individuals of 10 independent training simulations are implemented in a new environment shown in Fig. 2.7(b). Each individual is implemented 300 times with 10 different start positions in order to study the adaptability in unknown environments. The sudden changes are carried out by making one sensor breaks at the 500th time step of the 3000 life time steps for each implementation.

The average rewards in the environments with normal situation without sudden changes (case 1) are compared with that in the environments with sudden changes (case 2) as shown in Fig. 5.6.

- In case 1, Fuzzy GNP-TSRL (CS) shows the highest average reward in each time step, which means that Fuzzy GNP-TSRL (CS) has the highest generalization ability in unknown environments among the other methods.
- In case 2, when the sudden changes occur by a broken sensor at the 500th time step, the average rewards of all the methods decreased after the 500th time step. It is because the broken sensor damages wall-following behaviors of the robot so much. However, Fuzzy GNP-TSRL (CS) still has the highest average reward, which means the adaptability mechanism of Fuzzy GNP-TSRL (CS) works fairly well.

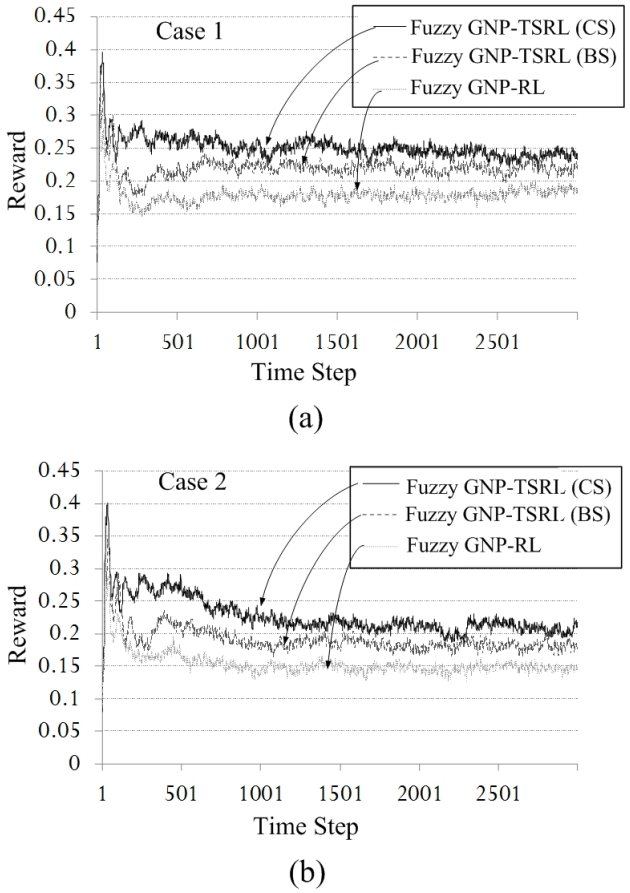


Figure 5.6: Average reward of Fuzzy GNP-TSRL (CS), Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL during the life time in the implementation phase

Results of Simulation 3

The effectiveness and efficiency of the adaptation mechanism of Fuzzy GNP-TSRL (CS) is also shown when several sensors break simultaneously, which means that the recovery processes are more difficult. In this case, the average reward of Fuzzy GNP-TSRL (CS) is still higher than that of Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL as shown in Fig. 5.7.

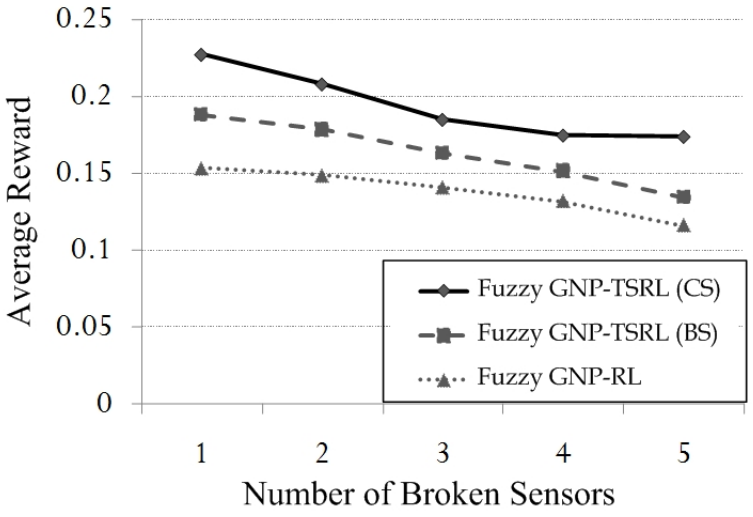


Figure 5.7: Average reward in the implementation phase with several broken sensors

5.6 Summary

Fuzzy GNP-TSRL (CS) has been proposed, which combines the Sub node Selection method (SS method) and Credit branch Selection method (CS method). Fuzzy GNP-TSRL (CS) determines the node transitions more appropriately to adapt to the changes of the environments comparing with Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL, where Fuzzy GNP-TSRL (CS) has the unique mechanism to avoid harmful nodes by skipping these nodes in the node transitions. As a result, the adaptability of Fuzzy GNP-TSRL (CS) shows the excellent average rewards in the implementation, especially when inexperienced changes of the environments occur.

Chapter 6

Conclusions

The primary aim of this study is to improve the robustness in uncertain environments and improve the adaptability of Genetic Network Programming with Reinforcement Learning (GNP-RL) when inexperienced changes occur in the environments for a mobile robot.

For evaluating the performances, a benchmark method of the wall following behaviors is used. The performances are studied in the training phase and in the implementation phase.

Improving the robustness of GNP-RL is studied by integrating Fuzzy logic to the judgment nodes, i.e., Fuzzy GNP-RL is proposed. Fuzzy GNP-RL improves the exploration ability to determine the node transitions more appropriately. In addition, introducing Gaussian noises to the sensor values also improved the exploration ability. As a result, the robustness of Fuzzy GNP-RL is improved in uncertain environments.

Improving the adaptability of GNP-RL is studied by proposing the mechanisms to recover the troubles. Here, Two-Stage Reinforcement Learning methods are proposed.

- Fuzzy GNP-TSRL (BS) combines Sub node Selection method (SS method) like Fuzzy GNP-RL and Branch connections Selection method (BS method). Fuzzy GNP-TSRL (BS) has the abilities to change the programs adaptively by changing functions and connections to adapt to the changes of the environments.

- Fuzzy GNP-TSRL (BS) with changing ϵ-greedy policy and learning rate α is studied to improve the balance between the exploration and exploitation when determining the action selections. Using this method, the on-line learning can determine the node transitions more appropriately, then the adaptability can be improved.

- Fuzzy GNP-TSRL (CS) combines Sub node Selection method (SS method) like Fuzzy GNP-RL and Credit branch Selection method (CS method) of Credit-GNP. This method proposed a method to skip harmful nodes. Then, the node transitions can determine the agent behaviors more appropriately.

Fuzzy GNP-TSRL (CS) shows the superior performance compared with Fuzzy GNP-TSRL (BS) and Fuzzy GNP-RL, because Fuzzy GNP-TSRL (CS) can skip harmful nodes, while the other methods can not avoid harmful nodes. Thus, the adaptability of Fuzzy GNP-TSRL (CS) has the best performance.

Bibliography

[1] H. Rahmandad and J. Sterman, "Heterogeneity and Network Structure in the Dynamics of Diffusion: Comparing Agent-Based and Differential Equation Models", *Management Science*, Vol. 54, No. 5, pp. 9981014, 2008.

[2] C. C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Part I", *IEEE Transactions on System and Cybernetics*, Vol. 20, No. 2, pp. 404-418, 1990.

[3] E. I. Papageorgiou, C. Stylios and P. P. Groumpos, "Unsupervised Learning Techniques for Find-Tuning Fuzzy Cognitive Map Causal Links", *Int. J. Human-Computer Studies*, Vol. 64, pp. 727-743, 2006.

[4] K. C. Ng and M. M. Trivedi, "A Neuro-Fuzzy Controller for Mobile Robot Navigation and Multirobot Convoying", *IEEE Transactions on System and Cybernetics*, Vol. 28, No. 6, pp. 829-840, 1998.

[5] K. O. Stanley and R. Miikkulainen, "Evolving Neural Networks through Augmenting Topologies", *Evolutionary Computation*, Vol. 10, No. 2, pp. 99-127, 2002.

[6] Y. Kassahun and G. Sommer, "Automatic Neural Robot Controller Design Using Evolutionary Acquisition of Neural Topologies", *In Proc. of the FACHGESPRCH AUTONOME MOBILE SYSTEME*, pp. 315–321, 2005.

[7] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge, MA 1992.

[8] K. Hirasawa, M. Okubo, H. Katagiri, J. Hu and J. Murata, "Comparison Between Genetic Network Programming (GNP) and Genetic Programming (GP), *In Proc. of the IEEE Congress on Evolutionary Computation*, Vol. 2, pp. 1276-1282, 2001.

[9] C.Chen and T. Xiao, "Probabilistic Fuzzy Control of Mobile Robots for Range Sensor Based Reactive Navigation", *Intelligent Control and Automation*, Vol. 2, No.2, pp. 77-85, 2011.

[10] J. M. Toibero, F. Roberti and R. Carelli, "Stable Contour-Following Control of Wheeled Mobile Robots", *Robotica*, Vol. 27, pp 1-12, 2009.

[11] I. Ayari and A. Chatti, "Reactive Control Using Behavior Modeling of a Mobile Robot", *International Journal of Computers, Communications and Control*, Vol. II, No. 3, pp. 217-228, 2007.

[12] O. Motlagh, S.H. Tang, N. Ismail and A.R. Ramli, "An Expert Fuzzy Cognitive Map for Reactive Navigation of Mobile Robots", *Fuzzy Sets and Systems*, Vol. 201, pp. 105-121, 2012.

[13] A. C. Murillo, P. Abad, J. J. Guerrero and C. Sagues, "Improving Topological Maps for Safer and Robust Navigation", *In Proc. of the IEEE Intelligent Robots and Systems 2009*, pp. 3609-3614, 2009.

[14] H. Kretzschmar, C. Stachniss and G. Grisetti, "Efficient Information-Theoretic Graph Pruning for Graph-Based SLAM with Laser Range Finders", *In Proc. of the IEEE Intelligent Robots and Systems 2011*, pp. 865-871, 2011.

[15] T. M. Gureckis and B. C. Love, "Learning in Noise: Dynamic Decision-Making in a variable Environment", *Jornal of Mathematical Psychology*, Vol. 53, pp. 180-193, 2009.

[16] P. Reigner, V. hansen and J. L. Crowly, "Incremental Supervised Learning for Mobile Robot Reactive Control", *Robotics and Autonomous Systems*, Vol 19, pp. 247-257, 1997.

[17] W. E. Dixon, D. M. Darren, E. Zergeroglu and A. Behal, "Adaptive Tracking Control of a Wheeled Mobile Robot via an Uncalibrated Camera System", *IEEE Transactions on Systems, Man and Cybernetics-PartB: Cybernetics*, Vol.31, No. 3, pp.341-352, 2001.

[18] K. T. Song and L. H. Sheen, "Heuristic Fuzzy-Neuro Network and Its Application to Reactive Navigation of a Mobile Robot", *Fuzzy Sets and Systems*, Vol. 110, pp. 331-340, 2000.

[19] E. Zalama, P. Gaudiano and J. L. Coronado, "A Real-Time, Unsupervised Neural Network for the Low-Level Control of a Mobile Robot in a Nonstationary Environment", *Neural Networks*, Vol. 8, No. 1, pp. 103-123, 1995.

[20] S. Yamada and M. Murota, "Unsupervised Learning to Recognize Environments from Behavior Sequences in a Mobile Robot", *In Proc. of the IEEE International Conference on Robotic and Automation*, pp. 1871-1876, 1998.

[21] R. S. Sutton and A. G. Barto, *Reinforcement Learning - An Introduction*, Cambridge, Massachusetts, London, England, MIT Press, 1998.

[22] L. P. Kaebling, M. L. Littman and A. W Moore, "Reinforcement Learning: A Survey", *Journal of Artificial Intteligent Research*, Vol. 4, pp. 237-285, 1996.

[23] W. D. Smart and L. P. Kaebling, "Effective Reinforcement Learning for Mobile Robots", In *Proc. of the IEEE on Robotics and Automation*, May 2002.

[24] L. Khriji, F. Touati, K. Benhmed and A. Al-Yahmedi, "Mobile Robot Navigation Based on Q-Learning Technique", *International Journal of Advanced Robotic Systems*, Vol. 8, No. 1, pp. 45-51, 2011.

[25] M. Menegaz and P. M. Engel, "Using the GTSOM Network for Mobile Robot Navigation with Reinforcement Learning", *In Proc. of the International Joint Conference on Neural Networks*, pp. 2073-2077, 2009.

[26] D. Tamilselvi, Dr. S. Mercy Shalinie and G. Nirmala, "Q Learning for Mobile Robot Navigation in Indoor Environment", *In Proc. of the IEEE International Conference on Recent Trends in Information Technology*, pp. 324-329, 2011.

[27] K. macek, I. Petrovic and N. Peric, "A Reinforcement Learning Approach to Obstacle Avoidance of Mobile Robots", *In Proc. of the IEEE Advanced Motion Control*, pp. 462-466, 2002.

[28] M. J. Er and Y. Zhou, "Automatic Generation of Fuzzy Inference Systems Via Unsupervised Learning", *Neural Networks*, Vol. 21, Issue 10, pp. 1556-1566, 2008.

[29] B. Q. Huang, G. Y. Cao and M. Guo, "Reinforcement Learning Neural Network to the Problem of Autonomous Mobile Robot Obstacle Avoidance", *In Proc. of the Fourth International Conference on Machine Learning and Cybernetics*, pp. 85-89, 2005.

[30] J. Holland, *Adaptation in Natural and Artificial System*, Univ. of Michigan Press, Ann Arbor, 1975; MIT Press, Cambridge, MA, 1992.

[31] X. Yao, "Evolving Artificial Neural Networks", *In Proc. of the IEEE*, Vol. 87, No. 9, pp. 1423-1447, 1999.

[32] O. Obe and I. Dumitrache, "Adaptive Neuro-Fuzzy Controller With Genetic Training For Mobile Robot Control", *International Journal of Computers, Communications and Control*, Vol. VII, No. 1, pp. 135-146, 2012.

[33] W. K. Wong, H. Y. Chen, C. Y. Hsu and T. K. Chao, "Reinforcement Learning of Robotic Motion with Genetic Programming, Simulated Annealing and Self-Organizing Map", *In Proc. of Conf. on Technologies and Applications of Artificial Intelligence*, pp. 292-298, 2011.

[34] S. Kamio and H. Iba, "Adaptation Technique for Integrating Genetic Programming and Reinforcement Learning for Real Robots", *IEEE trans. on Evolutionary Computation*, Vol. 9, No. 3, pp. 318-333, 2005.

[35] S. Mabu, K. Hirasawa and J. Hu, "A Graph-Based Evolutionary Algorithm: Genetic Network Programming (GNP) and Its Extension Using Reinforcement Learning", *Evolutionary Computation*, Vol. 15, No. 3, pp. 369-398, 2007.

[36] M. L. Littman and A. R. Cassandra, "Learning Policies for Partially Observable Environments: Salling Up", *In Proc. the 12th Int. Conf. on Machine Learning*, San Francisco, CA, pp. 1-59, 1995.

[37] T. Soule, J. A. Foster and J. Dickinson, "Code growth in Genetic Programming", *In Proc. of the First Annual Conference on Genetic Programming*, pp. 215-223, 1996.

[38] W. Banzhaf and W. B. Langdon, "Some Considerations on Reason for Bloat", *Genetic Programming and Evolvable Machines*, Vol. 3, pp. 81-91, 2002.

[39] Y. Lu, S. Mabu and K. Hirasawa, "Multicar Elevator Group Supervisory Control System using Genetic Network Programming", *IEEJ Transaction on Electronics, Information and System*, Vol. 6 (SI), pp. 65-73, 2011.

[40] Y. Chen, S. Mabu and K. Hirasawa, "Trading Rules on Stock Markets using Genetic network Programming with Sarsa Learning", *Journal of Advanced Computational Intelligence and intelligent Informatics*, Vol. 12, No. 4, pp. 383-392, 2008.

[41] L.Wang, S. Mabu and K. Hirasawa, "Genetic Network Programming with Rule Accumulation and Its Application to Tile-World Problem", *Journal of Advanced Computational Intelligence and intelligent Informatics*, Vol. 13, No. 5, pp. 551-560, 2009.

[42] C. Yue, S. Mabu and K. Hirasawa, "A Bidding Strategy for Continuous Double Auctions Based on Genetic Network Programming with Generalized Judgments", *In Proc. of the IEEE Systems, Man, and Cybernetics (SMC)*, pp. 144-151, 2011.

[43] S. Mabu, H. Hatakeyama, M. T. Thu, K. Hirasawa and J. Hu, "Genetic Network Programming with Reinforcement Learning and Its Application to Making Mobile Robot Behavior", *Trans. IEE Japan*, Vol. 126. No. 8, pp. 1009-1015, 2006.

[44] S. Mabu, A. Tjahjadi and K. Hirasawa, "Adaptability Analysis of Genetic Network Programming with Reinforcement Learning in Dynamically Changing Environments", *Expert Systems with Application*, 2012 (In Press, Corrected Proof)

[45] S. Y. Yi and M. J. Chung, "A Robust Fuzzy Logic Controller for Robot Manipulators with Uncertainties", *IEEE Trans. on Systems, Man and Cybernetics Part-B: Cybernetics, Vol. 27, No. 4, pp. 706-713, 1997*

[46] H. Du and N. Zhang, "Application of Evolving Takagi-Sugeno Fuzzy Model to Nonlinear System Identification", *Applied Soft Computing*, Vol. 8, pp. 676686, 2008.

[47] W. M. Hinojosa, S. Nefti and U. Kaymak, "Systems Control With Generalized Probabilistic Fuzzy-Reinforcement Learning", *IEEE Trans. on Fuzzy Systems*, Vol. 19, No. 1, pp. 51-64, 2011.

[48] S. Chen and C. Chen, "Probabilistic Fuzzy System for Uncertain Localization and Map Building of Mobile Robots", IEEE Trans. on Instrumentation and Measurement, Vol. 61, No. 6, pp. 1546-1560, 2012.

[49] L. Jiangrong, L. Junmin and X Zhile, "T-S Fuzzy Stochastic Bilinear Model and Fuzzy Controller Design Based on Switching Piecewise Lyapunov Functions", *In Proc. of the 30th Chinese Control Conference*, pp. 2825-2829, 2011.

[50] C. J. C. H. Watkins and P. Dayan, "Technical Note Q-Learning", *Machine Learning*, Vol. 8, pp. 279-292, 1992

[51] G. A. Rummery and M. Nirajan, "On-line Q-learning Using Connectionist Systems", *Cambridge University Engineering Dept.*, Tech. Rep. CUED/F-INFENG/TR 166, 1994.

[52] K. P. Murphy, "A Survey of POMDP Solution Techniques", *Technical Report*, U. C. Berkeley, 2000.

[53] K. Hsiao, L. P. Kaelbling and T. L. Perez, "Grasping POMDPs", *In Proc. of the IEEE Int. Conf. on Robotics & Automation*, pp. 44854692, 2007.

[54] H. Kurniawati, D. Hsu Wee and S. Lee, "SARSOP: Effcient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces", *In Proc. of the Robotics: Science & Systems* 2008.

[55] J. Loch and S. Singh, "Using Eligibility Traces to Find the Best Memoryless Policy in Partially Observable Markov Decision Processes", *In Proc. of the Intl. Conf. on Machine Learning*, 1998.

[56] E. Santos and X. Zhong, "Genetic Algorithms and Reinforcement Learning for the Tactical Fixed Interval Scheduling Problem", *International Journal on Artificial Intelligence Tools*, Vol. 10, No. 1-2, pp. 1-16, 2001.

[57] Y. Sakurai, K. Takada, T. Kawabe and S. Tsuruta, "A Method to Control Parameters of Evolutionary Algorithms by using Reinforcement Learning", *In Proc. of the IEEE Sixth International Conference on Signal-Image Technology and Internet Based Systems*, pp. 64-79, 2010.

[58] S. Goschin, E. Franti, M. Dascalu and S. Osiceanu, "Combine and Compare Evolutionary Robotics and Reinforcement Learning as Methods of Designing Autonomous Robots", *in Proc. of the IEEE Congress on Evolutionary Computation*, pp. 1511-1516, 2007.

[59] K. L. Downing, "Adaptive Genetic Programs via Reinforcement Learning", *In Proc. of the Genetic and Evolutionary Computation Conference*, pp. 19-26, 2001.

[60] A. Nemra and H. Rezine, "Genetic Reinforcement Learning Algorithms for Online Fuzzy Inference System tuning Application to Mobile Robotic", *Robotics, Automation and Control*, I-Tech, pp. 227-257, 2008.

[61] P. J. Angeline, G. M. Saunders and J. B. Pollack, "An Evolutionary Algorithm that Constructs Recurrent Neural Networks", *IEEE Trans. on Neural Network*, Vol. 5, No. 1, pp. 54-65, 1994.

[62] J. Zhong, X. Hu, J. Zhang and M. Gu, "Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms", *In Proc. of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, Vol. 02, pp. 1115-1121.

[63] M. Srinivas and L. M. Patnaik, "Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms", *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 24, No. 4, pp. 656-667, 1994.

[64] Z. Michalewicz, "Genetic Algorithms + Data Structures", Springer-Verlag Berlin Heidelberg New York, 1996.

[65] A. Neubauer, "A Theoretical Analysis of the Non-Uniform Mutation Operator for the Modified Genetic Algorithm", *In Proc. of the IEEE International Conference on Evolutionary Computation*, pp. 93-96, 1997.

[66] T. Braunl, "Embedded Robotics Mobile Robot Design and Applications with Embedded Systems", *Springer-Verlag Berlin Heidelberg*, 2006.

[67] K. Taboada, S. Mabu, E. Gonzales, K. Shimada and K. Hirasawa, "Mining Fuzzy Association Rules: A General Model Based on Genetic Network Programming and Its Applications", *IEEJ Transaction on Electrical and Electronic Engineering*, Vol. 128, No. 5, pp. 343-354, 2010.

[68] Y. Yang, S. Mabu, K. Shimada and K. Hirasawa, "Fuzzy Intertransaction Class Association Rule Mining Using Genetic Network Programming for Stock Market Prediction", *IEEJ Transaction on Electrical and Electronic Engineering*, Vol. 129, No. 1, pp. 1-8, 2011.

[69] L. Wang, W. Xu, S. Mabu and K. Hirasawa, "Rule Accumulation Method Based on Credit Genetic Network Programming", *In Proc. of the IEEE World Congress on Computational Intelligence*, pp. 3651-3658, 2012.

[70] F. Ye, L. Yu, S. Mabu, K. Shimada and K. Hirasawa, "Genetic Network Programming with Rules", *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol. 10, pp: 16-24, 2009.

[71] "Cyberbotics", [Online]. Available: www.cyberbotics.com/

[72] L. F. Wang, K. C. Tan and V. Prahlad, "Developing Khepera Robot Applications in a Webots Environment", *in Proc of IEEE 2000 International Symposium on Micromechatronics and Human Science*, pp. 71-76, 2000.

[73] S. Ishii, W. Yoshida and J. Yoshimoto, "Control of Exploitation-Exploration Meta-Parameter in Reinforcement Learning", *Neural Networks*, Vol 15, pp. 665-687, 2002.

[74] D. A. White, D. A. Sofge and S. B. Thrun,"The Role of Exploration in Learning Control", *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, pp. 1-27, Florence, Kentucky 41022, 1992.

Acknowledgments

My deepest gratitude and support of many people who are gratefully acknowledged here.

My deepest gratitude goes first and foremost to Professor Hirasawa for encouragement, patient guidance, ideas, and constructive comments to improve my research experiences. His patience and support helped me finish this dissertation. I hope that one day I would become as good an advisor to my students as he has been to me.

Secondly, I also would like to acknowledge great respect to the committee members, Prof. Jinglu Hu, Prof. Yoshie Osamu and Prof. Shigeru Fujimura, for their careful reviews, constructive advices and helpful suggestions in my work.

Thirdly, I also grateful to Dr. Mabu, whose kindness helped me a lot and gave invaluable suggestion during discussion in laboratory.

Thanks also to my friends in our laboratory, Ms. Yang Yang, Ms. Nanan Lu, Mr. Lutao Wang and Mr. Xianeng Li, who gave me encouragement and suggestions.

My heartfelt gratitude especially to my families, my father, my mother, and my sisters and brothers in Indonesia. Their love and support provided me the energy to attain my study.

Last my thanks would go to my beloved family, my husband, my daughter and my son, for love and support in my life.

This work is supported by the Directorate General of Higher Education, Indonesia.

List of Publications

Journals

J1 S. Sendari, S. Mabu and K. Hirasawa, "Two-Stage Reinforcement Learning on Credit Branch Genetic Network Programming for Mobile Robots, *IEEJ Transaction on Electronics, Information and Systems*. (Accepted)

J2 S. Sendari, S. Mabu, A. Tjahjadi and K. Hirasawa, "Fuzzy Genetic Network Programming with Noises for Mobile Robot Navigation, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol. 15, No. 7, pp. 767-776, 2011.

International Conferences (with Review Process)

C1 S. Sendari, S. Mabu and K. Hirasawa, "Two-Stage Reinforcement Learning Based on Genetic Network Programming for Mobile Robot, *In Proc. of the SICE International Annual Conference*, pp. 95-100, Akita, Japan, 2012/8.

C2 S. Sendari, S. Mabu and K. Hirasawa, "Fuzzy Genetic Network Programming with Reinforcement Learning for Mobile Robot Navigation, *In Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 2243-2248, Anchorage, USA, 2011/10.