Usage of ZCS Evolutionary Classifier System as a Rule Maker for Cleaning Robot Task

Tomáš Cádrik and Marian Mach

Department of Cybernetics and Artificial Intelligence, Technical University of Košice
tomas.cadrik@gmail.com, Marian.Mach@tuke.sk

Abstract. This paper introduces the Cleaning robot task which is a simulation of the cleaning of a room by a robot. The robot must collect all the junk in the room and put it into a container. It must take out the junk sequentially, because the amount of carried trash is limited. The actions of this robot are selected by using the Michigan style classifier system ZCS. This paper shows the capability of this system to select good rules for the robot to perform the cleaning task.

1 Introduction

Learning classifier systems were introduced by John Holland in 1986 [1]. It is a technique which creates classifiers using evolutionary algorithms [2]. Later, Wilson created two successful classifier systems. They were named ZCS [3] and XCS [4] [5]. In these systems, each individual (also called classifier) represents only one rule. This style is called the Michigan style. Another style is called the Pittsburgh style, where each individual contains all rules. The learning classifier systems also have close links to reinforcement learning because they use fitness calculating algorithms which are similar to reinforcement learning techniques.

Classifier systems were tested on many problems. Some of them were single-step problems like the k-multiplexor problem [4]. But these systems show good results especially on multi-step problems like Animat [4] or Corridor problem [6]. This paper focuses on ZCS (Zeroth level classifier system) and on a simple multi-step task which was mentioned in the cleaning robot task (CR) introduced in this paper. The CR task consists of a room which must be cleaned by the robot and all of the junk scattered in the room must be put into a container. The usage of ZCS to solve the CR task can show the ability of this method on a multi-step problem which is more difficult than the Animat problem, because the robot must pick up junk and simultaneously take out this junk periodically and put it into the container, because the robot can carry only a limited amount of it.

This paper is organized as follows: First, the ZCS classifier system is described. In section III the CR task is introduced. Section IV contains experiments which show the capability of ZCS to solve the CR task.

2 Description of ZCS

ZCS was introduced in [3]. Each ZCS individual contains a rule consisting of conditions and action parts. The condition part reflects a particular state(s) of the environment. It is coded with symbols {0, 1, #}, where # means ‘does not care’, it is equal to 0 and 1 at the same time. The action part can be coded with any symbols.

ZCS also contains a genetic algorithm that is used to discover new individuals and a covering operator that is employed when no condition part matches the input. The following figure shows the schematic illustration of ZCS.

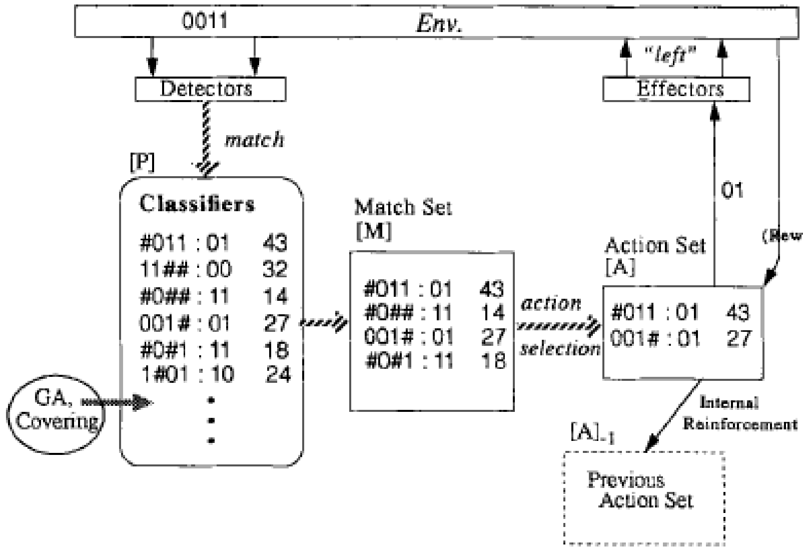

Fig. 1. Schematic illustration of ZCS by Wilson [2]

The ZCS starts with a randomly generated population [P]. Each individual starts with fitness equal to S0. First, the environment returns string corresponding to the current state. Then, ZCS makes a match set [M] to the input string. If [M] does not contain any classifier, the covering operator is activated. The covering operator creates a new classifier which matches the input and has a random action part. Then, each element in the condition part is changed with probability P # to # symbol. The initialization value of fitness is set to the average of fitness in the population. This classifier is inserted into the population and into [M]. Then, a classifier is deleted from the population according to the conversion of the fitness (1/fitness).

According to the fitness, a roulette wheel is used to choose an action from [M]. Then, each classifier from [M], which action is equal to the selected action, is copied to action set [A]. Each classifier in [M] that is not in [A] will update its fitness according to formula 1.

$$fitness_j = fitness_j - fitness_j * \tau \tag{1}$$

τ is a parameter that contains values from interval (0, 1>. Then, for each classifier in [A], a value will be calculated according to formula 2.

$$value_j = fitness_j * \beta \tag{2}$$

β can contain values from the same domain as τ. Each classifier in [A] will decrease its fitness with the calculated value. Then, the bucket B is calculated using the following formula.

$$B = \sum_{j=1}^{n} value_j \tag{3}$$

When the environment returns a reward, each classifier in [A] will update his fitness according to formula 4.

$$fitness_j = fitness_j + \beta * \frac{reward}{|A|} \tag{4}$$

Where |A| is the number of classifiers in [A]. Each classifier in the previous action set updates its fitness according to formula 5.

$$fitness_j = fitness_j + \gamma * \frac{B}{|Aprev|} \tag{5}$$

B is the value in the bucket, |Aprev| is the cardinality of the previous action set and γ is a parameter and has value from interval (0, 1>.

The last part of ZCS is the evolutionary algorithm. It is applied within each cycle with probability ψ. When it appears, two individuals are selected from the population by a roulette wheel according to their fitness. They are used as parents. Two new individuals are created using one point crossover (crossover starts with probability χ) and mutation (probability for each position in the string to mutate is μ). New individuals (children) start with fitness equal to the average fitness of their parents. Then, two individuals are deleted and these two new individuals are added to the population.

3 The Cleaning Robot Problem

The environment in the CR problem consists of a robot, junk and a container. The goal is to collect the junk and put it into the container. The robot can carry only a limited amount of junk. So when the robot carries the maximal amount of junk, it must go to the container and then it can continue collecting more junk on the map.

The robot can only see the surroundings and it knows how far the container is from it. The input from the environment is coded using 25 bits. The surrounding of the robot is coded in 16 bits. The first two bits represent the cell to the north from the robot. Each other pair of bits is the clockwise surrounding cells. The pair of bits "00" means that there is a wall. "01" means that there is an empty cell. "10" means that there is some junk in this cell. "11" means that there is a container in this cell. For example, the 16 bits of a robot in Figure 2 will be 0000011010000000.

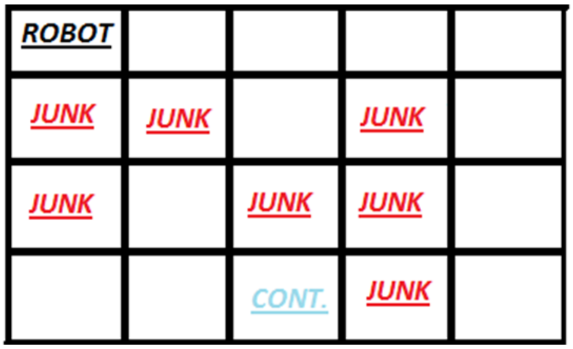

Fig. 2. Environment of a Cleaning robot problem

The distance of the robot from the container is coded in 8 bits. The first four bits are for the x distance and the second group represents the y distance. The first bit in these four bits means that the container is left or right from the robot (up or down if it is the y distance). The other three bits are the absolute value of the distance (if the maximal absolute distance is more than 7, we need more bits). In Figure 2, the distance of the robot from the container is 10101011 in bit representation. The last bit of the input string shows whether the robot can carry more junk.

When the robot steps on a cell with junk in it and it can take more junk, it will take it. Then the environment will return payoff 1000. When the robot steps on a cell with a container, it will put all junk it carries into it and the environment will return payoff 1000. If not, the returned payoff is 0.

The robot starts on a start cell (On Figure 2, the start cell is x=0, y=0). It can move in four directions (left, right, up, down). When the robot moves in a direction where there is a wall, it will stay on his position but he will lose one turn. When it puts all junk into the container, the environment will be restarted and the robot will start on its start cell. The goal of this problem is to take all the junk from the environment and put it into the container with usage of a minimum number of steps. The amount of junk doesn't always need to be the same in the environment after each cleaning. Figures show where the junk can appear but whether it appears on this position is random. So after restarting the environment, the amount of trash can be different. ZCS needs to learn how the environment looks and which action is the best for the input returned from the environment even if the environment is not the same for every cleaning cycle.

4 Experiments

The experiments were made on the environment displayed in Figure 2. Parameters of ZCS were set to values found in [7]. These values are shown in Tab 1.

Table 1. Parameter values of ZCS found in [7]

Parameter	S_0	τ	ψ	μ	χ	$P_\#$	β	γ
Value	20	0.1	0.25	0.02	0.5	0.33	0.2	0.4

In the first experiment, the maximum of carried junk was set to 1 (optimum is 31). Population was set to 10000. The number of cycles was 3000. The result while using the maximum amount of carried trash set to one is shown in Figure 3.

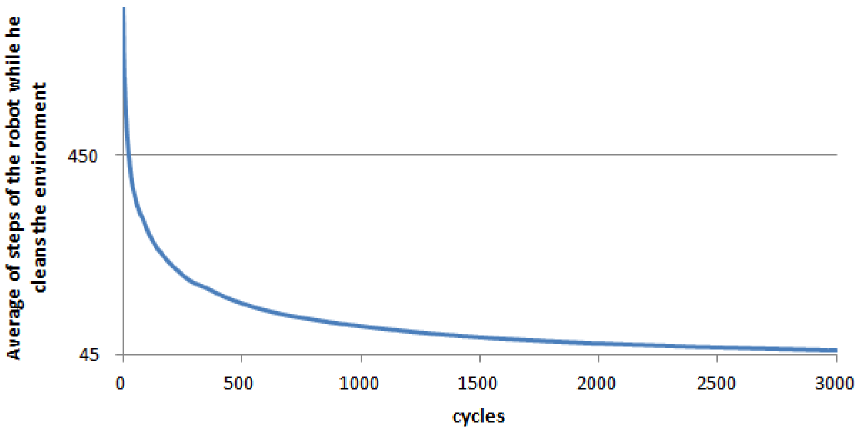

Fig. 3. Graph of dependence of the average number of steps of the robot while it cleans the environment and cycles. One cycle is one cleaning process, it ends when the environment contains no junk. The robot can carry only one piece of junk at a time.

In the first cycles the robot moves randomly. It can be seen in the graph, because these cycles have a big amount of steps while it cleans the environment. When the ZCS starts to learn how the environment looks like, the average of steps begins to decrease and the decreasing does not stop until the end of the last cycle.

The experiment shows the capability of ZCS to create rules in a more complicated problem than the animat problem is. It can gradually adapt to a changing environment (when the robot takes junk from a cell, the next time it visits that cell, it is clean) and choose good actions while the environment isn't clean (does not contain any junk). The number of steps is not optimal but the system shows progress in each cycle. Even if the cycles have different numbers of junk, it wasn't any problem for the ZCS.

The robot in the next experiment can carry three pieces of junk at once. The result while using the maximum amount of carried trash set to three (optimum is 15) is shown in Figure 4.

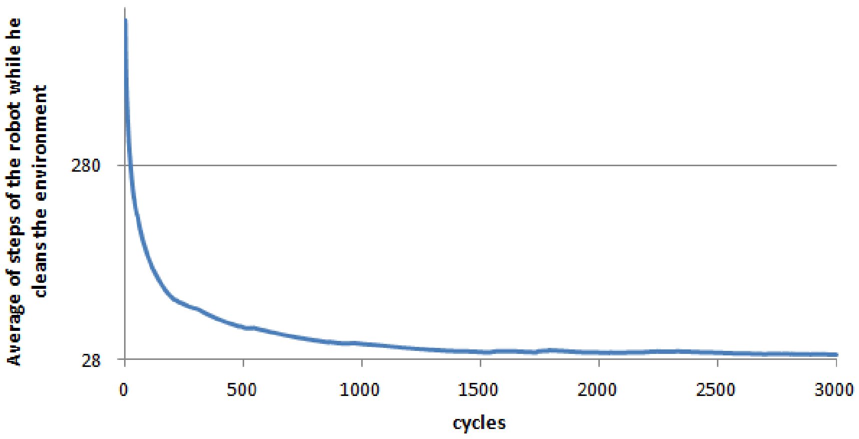

Fig. 4. Graph of dependence of the average number of steps of the robot while it cleans the environment and cycles. One cycle is one cleaning process, it ends when the environment contains no junk. The robot can carry only three pieces of junk at a time.

Figure 4 shows how ZCS adapts when the robot can carry more junk than in the first experiment. While the maximum pieces of carried junk was 1, the average number of steps was not less than 47. But if the maximum pieces of carried junk was three, the average number of steps was close to 28. This demonstrates how ZCS can utilize opportunities at the disposal.

5 Conclusion

This paper showed how ZCS can handle a multi-step problem named the Cleaning robot problem. Although this paper tested ZCS only on a simulation of a robot in a discrete world, it has a practical use. If we teach a cleaning robot where the most frequent places are, where junk or dirt can show up, we can use this robot to clean these places and it will clean only those where dirt is. Meanwhile, the robot will learn better rules. Some modifications can help ZCS to approve learning in larger and more complicated environments.

References

1. Holland, J.H.: Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In: Michalski, R.S., Carbonell, J.G. (eds.) Machine Learning, an Artifiial Intelligence Approach, vol. II, Morgan Kaufmann, Los Altos (1986)
2. Mach, M.: Evolutionary algorithms: Elements and principles. Elfa, Kosice (2009)
3. Wilson, S.W.: ZCS: A Zeroth Level Classifier System. Evolutionary Computation 2(1), 1–18 (1994)

4. Wilson, S.W.: Classifier Fitness Based on Accuracy. Evolutionary Computation 3(2), 149–175 (1995)
5. Butz, M.V., Wilson, S.W.: An Algorithmic Description of XCS. In: Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 2000. LNCS (LNAI), vol. 1996, pp. 253–272. Springer, Heidelberg (2001)
6. Tang, K.W., Jarvis, R.A.: Is XCS Suitable For Problems with Temporal Rewards? In: Proceedings 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 2006), vol. 2. IEEE (2006)
7. Computer Society, Washington, DC, pp. 258–264 (2005)
8. Cádrik, T.: Evolučné klasifikačné systémy, Diplomová práca. Technická univerzita, Košice, 74s (2013)