Design and implementation of agents' motion by genetic algorithm

NAITO-Horizon

Kenji Ozoe[*1], Nobuhiro Ito[*2]

[*1]Department of Information engineering,
Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya City,
Aichi Prefecture 466-8555, JAPAN
agent-staff@phaser.elcom.nitech.ac.jp

[*2]Software Science Course,
Department of Applied Information Science,
Faculty of Management and Information Science,
Aichi Institute of Technology
1247 Yachigusa, Yakusa-Cho, Toyota City,
Aichi Prefecture 470-0392, JAPAN
n-ito@aitech.ac.jp

Abstract. It is difficult to implement motions of 3D soccer agents. Because it is necessary to control 20 joints to realize motions of agents, and we consider interaction between joints of the agent. We apply genetic algorithm to implement motions of agents.

1 Introduction

In Robocup Soccer Simulation League, Humanoid agent model is in use. And we must control more than 20 joints to realize motions of agents. So, it is difficult to realize motions because we consider interaction between joints of the agent.

We apply genetic algorithm to implement motions of agents. By using genetic algorithm, we can get suboptimal motions of agents efficiently. In the following, we describe the details about example of walking motion.

2 Walking motion

We realize the walking motion by the combination of 4 poses as follows.

```
int action = 1
while (walk == true) {
  switch (action){
    case 1: up right leg
      break
    case 2: down right leg
      break
    case 3: up left leg
      break
    case 4: down left leg
      break
  }
  acton = action + 1
  if(action == 5) action = 1
}
```

Then, it is necessary to consider the following points.

– Each Joint angle which is in each pose
– Change speed of each joint angle from a certain pose to a certain other pose

However, as for these problems, it is difficult to get optimal solution for that multiple joints interact complicatedly. Especially if we implement by hand coding, we require an enormous amount of time to get solution. In addition, there is no guarantee that we get optimal solution. Therefore we devise a method to get solution for efficiently by the learning method using genetic algorithm. In the following chapter, we describe the details of the method.

3 Implementation of motion using genetic algorithm

3.1 Implementation procedure of genetic algorithm

We show procedure of genetic algorithm as follows.

Step 1: Make N agents
Step 2: Make a new agent by crossover from N agents
Step 3: Mutate a new agent
Step 4: Evaluate the fitness of a new agent
Step 5: Delete a agent of the lowest fitness from N+1 agents
Step 6: Until termination go to step 2

We apply genetic algorithm to soccer agents as follows.

Step 1: Express motions of a agent by gene expression
Step 2: Make a agent with a gene expression
Step 3: Make 10 genotypes
Step 4: Make a new genotype with mutation and crossover from 10 genotypes
Step 5: Simulate in a new genotypical agent and calculate fitness
Step 6: Delete a agent of the lowest fitness from 11 agents
Step 7: Until termination go to step 4

And we must design next points for implement it in this way.

1. Gene expression
2. Crossover
3. Mutation
4. Fitness

In the following, we describe the details.

3.2 Gene expression

We design gene expression to apply genetic algorithm. We simplify by restrict moving joints only six. We describe moving joints in the following.

- Right articulatio coxae : rlj2
- Right knee joint : rlj4
- Right ankle joint : rlj5
- Left articulatio coxae : llj2
- Left knee joint : llj4
- Left ankle joint : llj5

We do not move other joints. "up right leg", "down right leg" poses and "up left leg", "down left leg" poses are symmetry. So, we reverse "up right leg", "down right leg" poses and decide "up left leg", "down left leg" poses. We assume walking motion the combination of two pauses of "up right leg" and "down right leg".

We make the genotype the angle of each joint of two pauses. The genotype consist of 12 elements in the following. Each value are real number.

- Leg up pose
  - up rlj2 : url2
  - up rlj4 : url4
  - up rlj5 : url5
  - up llj2 : ull2
  - up llj4 : ull4
  - up llj5 : ull5
- Leg down pose
  - down rlj2 : drl2
  - down rlj4 : drl4
  - down rlj5 : drl5
  - down llj2 : dll2
  - down llj4 : dll4
  - down llj5 : dll5

3.3 Crossover

We make the genotype of a new agent with average from each joint of two agents which we chose at random. The expression is as follows.

```
rand() = [1,10]

agent1 = agent[rand()]
agent2 = agent[rand()]

agentnew_url2 = ( agent1_url2 + agent2_url2 ) / 2
agentnew_url4 = ( agent1_url4 + agent2_url4 ) / 2
agentnew_url5 = ( agent1_url5 + agent2_url5 ) / 2
agentnew_ull2 = ( agent1_ull2 + agent2_ull2 ) / 2
agentnew_ull4 = ( agent1_ull4 + agent2_ull4 ) / 2
agentnew_ull5 = ( agent1_ull5 + agent2_ull5 ) / 2
agentnew_drl2 = ( agent1_drl2 + agent2_drl2 ) / 2
agentnew_drl4 = ( agent1_drl4 + agent2_drl4 ) / 2
agentnew_drl5 = ( agent1_drl5 + agent2_drl5 ) / 2
agentnew_dll2 = ( agent1_dll2 + agent2_dll2 ) / 2
agentnew_dll4 = ( agent1_dll4 + agent2_dll4 ) / 2
agentnew_dll5 = ( agent1_dll5 + agent2_dll5 ) / 2
```

3.4 Mutation

We add random value in the -1.0 to +1.0 range to all gene loci. The expression is as follows.

```
rand() = [-1.0,1.0]

new_url2 = new_url2 + rand()
new_url4 = new_url4 + rand()
new_url5 = new_url5 + rand()
new_ull2 = new_ull2 + rand()
new_ull4 = new_ull4 + rand()
new_ull5 = new_ull5 + rand()
new_drl2 = new_drl2 + rand()
new_drl4 = new_drl4 + rand()
new_drl5 = new_drl5 + rand()
new_dll2 = new_dll2 + rand()
new_dll4 = new_dll4 + rand()
new_dll5 = new_dll5 + rand()
```

3.5 Fitness

We operate a new agent with the simulator. The fitness is defined as the time that agent requires constant distance to walk. The shorter necessary time is, the better the fitness is.

3.6 Result

We describe the result of walking that uses genetic algorithm as follows. We measure migration time according to the rule of Walking Challenge in 2007. The agent starting from (-18, 0) and the ball is in the center of the field. We measure the time the agent takes to touch a ball. The following table shows the migration time.

Table 1. migration time

	before applying	after applying
time[sec]	40.64	15.84

As can be expected from Table 1, The application of genetic algorithm brings the quite positive result compared with before it applies. This shows that better motion can be achieved by the study method that we designed. And by this method, we can streamline motions such as kick and getting up.

4 Conclusion

We paid attention to genetic algorithm to implement motions of agents. We designed agent's motion by the gene expression, and implemented walk. As a result, we realized motions with high performance than before. In the future, we realize better motions by improving this learning method.

References

1. Azuma Ohuchi, Masahito Yamamoto, Hidenori Kawamura. Theory and Application of Multi-agent Systems.