12

Evolutionary Adaptive Behavior in Noisy Multi-Agent System

Takamasa Iio, Ivan Tanev, Katsunori Shimohara and Mitsunori Miki
Doshisha University
Japan

1. Introduction

Multi-agent systems have become more and more important in many aspects of computer science such as distributed artificial intelligence, distributed computing systems, robotics, artificial life, etc. Based on the belief that any complex system is more than the sum of its individual elements (Holand, 1999, Morgan, 1923, Morowitz, 2002), multi-agent systems introduce the issue of the emergence of behavior through interactions between agents (Forrest, 1991). Accordingly, a coordinated behavior needed to archive complex tasks might emerge in multi-agent systems from relatively simple defined interactions between agents. An agent is a virtual entity that can act, perceive the proximity of its environment and communicate with others; it is autonomous and has the ability to achieve its goals. Multi-agent systems contain a world (environment), entities (agents), relations between the entities, a way the world is perceived by the entities, a set of operations that can be performed by the entities and changes in the world as a result of these actions.
The main application areas of multi-agent systems are problem solving, simulation, collective robotics, software engineering, and construction of synthetic worlds (Ferber, 1999). Considering the latter application area and focusing on the autonomy of agents and the interactions that link them together (Parunak, Van, Brueckner, Fleischer & Odell, 2002), the following important issues can be raised: What is the minimum amount of perception information needed by agents in order to perceive the world? How can agents cooperate? What are the methods, and what are the lower bounds of communications, required for them to coordinate their actions? What architecture should they feature so that they can achieve their goals? What approaches can be applied to automatically construct their functionality, with the quality of such a design being competitive with human-handcrafted design? These issues are of special interest, since the aim is to create multi-agent systems, which are scalable, robust, flexible, and able to adapt to changes automatically. These features of multi-agent systems are believed to be particularly important in real world applications where the approaches to construct synthetic worlds can be viewed as practical methods, techniques towards creating complex “situation-aware” multi-computer, multi-vehicle, or multi-robot systems based on the concepts of agents, communication, cooperation and coordination of actions.
The purpose of our research is an automatic design of the coordinated behavior among agents. Particularly, we are interested in the robustness of the coordinated behavior to some

uncertainty derived from a mechanical or electrical noise of real-world applications. In this document we intend to highlight the following issues:

- Applying the genetic programming paradigm for evolving the coordinated behavior among agents, which interact with each other according to implicit interaction in an ideal noiseless environment,
- Testing the capability of the above behavior in a noisy environment, and
- Evolving the behavior again in two different environments; noiseless environment and noisy environment, for a comparative investigation of the robustness of the two types of evolved behavior.

We employ the predator prey pursuit problem (Benda, 1986) to verify the hypothesis of emergence of surrounding behavior in multi-agent systems from simply defined interactions between the agents. The noisy environment is implemented as the perceptual noise of predator agents; that is to say, in noisy environment the predator agents get uncertain perception information with some noise.

The remaining of the document is organized as follows. Section 2 introduces an instance of the general, well defined yet difficult to solve predator-prey pursuit problem as the task which we use in this work. Section 3 elaborates on the strongly typed genetic programming, employed as an algorithmic paradigm to evolve the functionality of agents. Section 4 explains the model of the perceptual noise of the predator agents. In Section 5, the comparative empirical results of evolution of surrounding behavior in a noiseless environment, execution of the surrounding behavior in a noisy environment and re-evolution of the surrounding behavior in each of the noiseless and the noisy environment are presented. Our conclusions are drawn in Section 6.

2. Configuration of a multi-agent system

2.1 Instance of Predator prey pursuit problem

In order to investigate relationships between a coordinated behavior among agents and uncertainty of their perception, we address predator prey pursuit problem (Benda, 1986), which is a game that some predator agents chase and catch a prey agent on a two dimensional field. The problem is general, well-defined and well-studied in multi-agent systems yet difficult to solve because the predator agents cannot capture the prey agent without a harmonious teamwork.

In our work, there are four predator agents and a prey agent on a two dimensional torus field. Considering a more realistic instance of the problem than the previous works (Haynes & Sen, 1996, Haynes, Wainwright, Sen & Schoenefeld, 1997, Luke & Spector, 1996), the field is a simulated continuous torus instead of coarse grid. The snapshot of our software is shown in Figure 1. All agents have moving and perceptual abilities. Their moving abilities are also continuous; they can turn to any angle from their current heading and can run with speed equal to 0, 0.25, 0.5, 0.75 and 1.0 of their maximum speed. We introduce a proximity perception model in that the predator agents can see the prey agent and only the closest predator agent, only when these agents are within the limited range of visibility of their simulated sensors. The prey employs random wandering if there is no predator in sight and an a priori handcrafted optimal escaping strategy as soon as predator(s) become "visible." In order to make a situation where the predator agents cannot capture the prey unless they collaborate with each other, we made the maximum speed of the predator agents lower than

that of the prey agent but the range of visibility of the predator agents wider than that of the prey to allow the predators to stalk the prey.
The situation requires a relatively complex behavior that the predator agents surround the prey agent on all sides in the world, and the behavior should be emerged from simple, local, implicit and proximity-defined interactions between the predator agents.

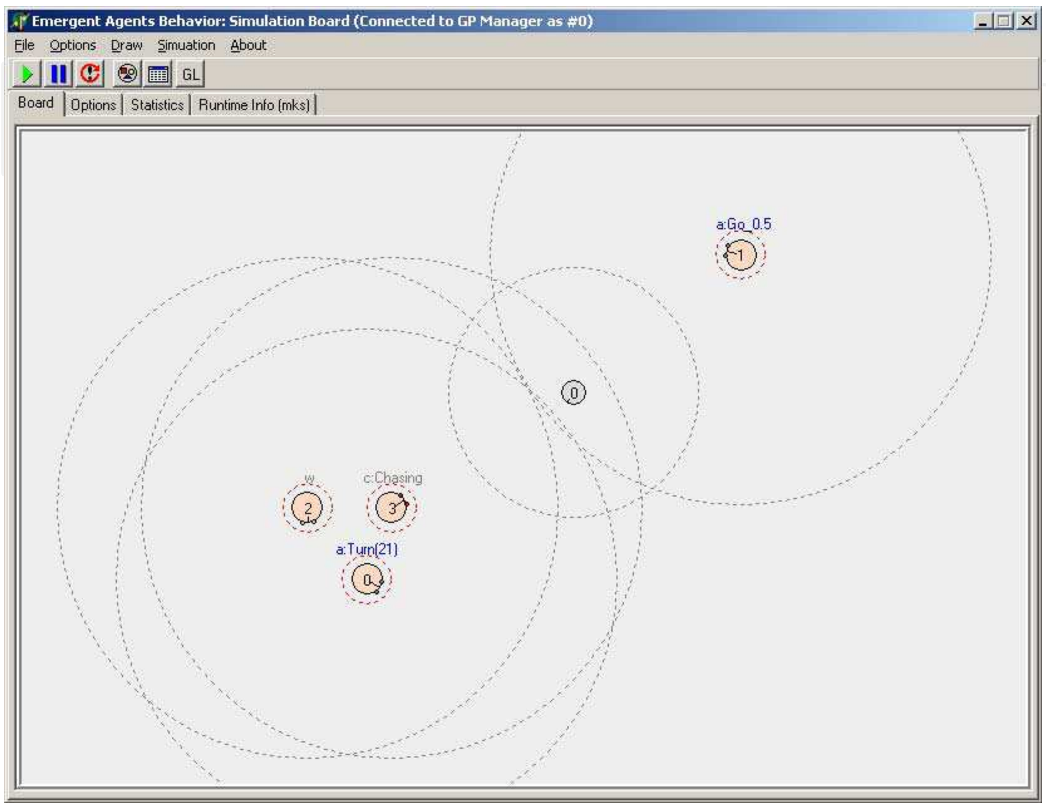

Emergent Agents Behavior: Simulation Board (Connected to GP Manager as #0)
File Options Draw Simuation About
GL
Board Options Statistics Runtime Info (mks)
a:Go_0.5
1
0
w
2
c:Chasing
3
a:Turn(21)
0

Fig. 1. A snapshot of the instance of predator prey pursuit problem.

2.2 Architecture of the agents

We adopted a subsumption architecture (Brooks, 1986) as the architecture of the predator agents; it was comprised of functional modules separated in three levels: wandering, greedy chase and surrounding (Figure 2(a)). Wandering module makes the predator agents walk

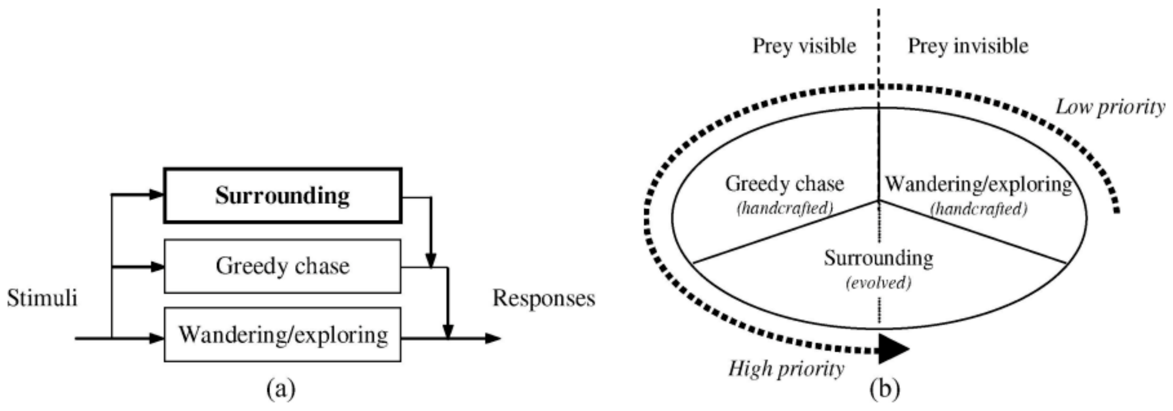


Fig. 2. Subsumption architecture of the agents: functional structure (a) and states (b).

around randomly when other agents are not within their sight, and greedy chase module makes them chase the prey agent while it is within their sight. These modules are handcrafted. Surrounding module makes the predator agents run evolved behavior program when another predator agent is in their sight. The program is designed automatically via simulated evolution. The highest priority module is surrounding, the next is greedy chase, and the lowest priority is wandering. The priority allows the following two things to be evolved simultaneously; (i) the capability of agents to resolve social dilemmas, determined by the way surrounding behavior overrides greedy chase when the prey is in sight, and (ii) the capability to resolve the exploration-exploitation dilemma, determined by the ability of surrounding behavior to override wandering when the prey is invisible.

3. Genetic programming to automatically design surrounding behavior

We employ genetic programming to automatically design surrounding behavior represented as a set of stimulus-response rules of the predator agents. Genetic programming is a domain-independent problem solving approach in which a population of computer programs (individuals' genotypes, in this case surrounding behavior programs) is evolved to solve problems (Koza, 1992).

The simulated evolution in genetic programming is based on the Darwinian principle of reproduction and survival of the fittest. The fitness of each individual is based on the quality with which the phenotype of the simulated indivdual is performing in a given environment; that is to say, predator agents that can capture the prey agent successfully and quickly have a higher fitness value, and their surrounding behavior program are more likely to remain in the next generation.

In the remaining of this section, we elaborate strongly-typed genetic programming for limiting the search space of genetic programming and the major attributes of genetic programming; function and terminal set, genetic representation, genetic operations and fitness evaluation.

3.1 Strongly-typed genetic programming with exception handling

Genetic programming can automatically evolve a set of stimulus-response rules of arbitrary complexity without the need to a priori specify the extent of such complexity; however, that might often cause an enormous computational effort needed to discover a huge search space while looking for potential solutions to the problem. In that respect the introduction of "pruning algorithms" is a significant towards an efficient search for a solution in huge and multidimensional search spaces (Morowitz, 2002). We impose a restriction on the syntax of evolved genetic programs based on some a priori known semantics. The approach is known as strongly typed genetic programming and its advantage over canonical GP in achieving better computational effort is well proven (Montana, 1995).

Considering the sample rule shown in Figure 3, which express a reactive behavior that if each predator agent gets the stimulus of its own speed being less than 20 mm/s, it turns to the bearing of the peer agent (i.e. the closest predator agent that is visible) plus 10 degrees, it is noticeable that both the return values of functions and their operands are associated with data types such as Boolean (the return value of Boolean expression (Speed < 20)), speed (e.g. variable Speed and constant 20), and angle of visibility (bearing) (e.g. variable Peer a and constant 10). An eventual arbitrary creation or modification of a genetic program

semantically would make little sense: indeed, it is unfeasible to maintain Boolean expressions comparing operands of different data types, because they have different physical units.
Moreover, since we introduce sensor range limits, there is a clear possibility of maintaining phenotypically inactive, genotypically neutral code in genetic programs; for example, if a Boolean expression, that compares a perception variable of a certain data type with a constant value beyond that data type's limits (e.g. Peer d > 1000, if that sensors' range is only 400) evaluates always as a constant True or False. Analogically, the semantics of action Turn( ) imply a parameter of data type angle. And allowing only addition and subtraction as arithmetic operations implies that each operand involved in the expression that defines the parameter (the resulting turning angle) should have the same data type angle. Addressing these concerns, the grammar of strongly-typed genetic programming establishes generic data types of visible angle, distance, speed, and Boolean with corresponding allowed ranges of values for their respective instances (variables and ephemeral constants). In addition, it stipulates the data type of the results of arithmetic and logical expressions, and the allowed data type of operands (perception variables and ephemeral constants) involved in these expressions.
We would like to emphasize that proposed approach is not based on domain-specific knowledge, and therefore the proposed strongly-typed genetic programming cannot be considered a "stronger" approach compromising the domain-neutrality of the very GP paradigm itself. The limitations imposed on the syntax of genetic programs are solely based (i) on the natural presumption that the predator agents are fully aware of their physically reasonable limits of their perception- and moving abilities; and (ii) on the common rule in strongly-typed 3G algorithmic languages that all the operands in addition, subtraction and comparison operations should have the same data types. These limitations do not incorporate a priori obtained knowledge, specific for the domain nor the external world where the agents are situated.

```
IF (Speed<20) THEN Turn(Peer_a+10)
```

Fig. 3. Sample stimulus-response rule.

3.1.1 Function set and terminal set

Genetic programs can be represented as parsing trees whose nodes are functions, variables or constants. The nodes that have sub-trees are non-terminals; they represent functions to which the sub-trees represent the arguments. Variables and constants are terminals; they take no arguments and they always are leaves in the parsing trees.
The set of those terminals includes the perceptions (stimuli) and actions (responses) the predator agents are able to perform as summarized in Table 1.
The function set comprises the arithmetic and logical operators, and the IF-THEN function which establish the relationship between current perceptions and corresponding actions. The terminal set comprises the sensory abilities, state variable, ephemeral constants and moving abilities. The sensory abilities and state variable are variable numbers; these numbers can be renewed by the perceptions of each predator agents, while the ephemeral constants and moving abilities are constant numbers. The detail of those sets is described in Table 1.

Category	Designation	Explanation

Function set	IF-THEN
	IF-THEN-NA,
	LE, GE, WI, EQ, NE, + , −
		IF-THEN
		IF-THEN with exception handling
		≤ , ≥ , Within, = , Not = , + , -

Terminal set		
Sensory abilities	Prey_d, Peer_d	Distance to the prey and to the closest agent, mm.
	Prey_a, Peer_a	Bearing of the prey and of the closest agent, degrees
	PreyVisible, PeerVisible	True if prey/agent is “visible,” false otherwise
State variable	Speed	Speed of the agent, mm/s
Ephemeral constants	Integer	
Moving abilities	Turn(a)	Turns relatively to a degrees (a > 0: clockwise)
	Stop, Go_1.0	Sets speed to 0, or to maximum, respectively
	Go_0.25, Go_0.5, Go_0.75	Sets speed to 25%, 50%, 75% of maximum

Table 1. Function set and terminal set of strongly-typed genetic programming.

3.1.2 Representation of genotype

Inspired by its flexibility, and the recently emerged widespread adoption of document object model (DOM) and extensible mark-up language (XML), we represent evolved genotypes of simulated the predator agents as DOM-parse trees featuring equivalent flat XML-text in a way as first implemented in [DOM/XML]. Our additional motivation stems from the fact that despite the recently reported use of DOM/XML for representing computer architectures, source code, and agents’ communication languages, we are not aware of any attempts to employ XML technology for representing evolvable structures such as genetic programs in a generic, standard, and portable way. Our approach implies that the genetic operations are performed on DOM-parse trees using off-the shelf, platform- and language neutral DOM-parsers. The corresponding XML-text representation (rather than S-expression) is used as a flat file format, feasible for migration of genetic programs among the computational nodes in an eventual distributed implementation of the genetic programming. A fragment of XML representing of the above sample stimulus-response rule (refer to Figure 3) is shown in Figure 4. The benefits of using DOM/XML-based representations of genetic programs, as documented in (Tanev, 2003) can be briefly summarized as follows:

i. XML tags offer a generic support for maintaining data types in genetic programming (strongly typed genetic programming);
ii. W3C-standard XML schemas offer a generic way to represent the grammar of genetic programming;
iii. Fast prototyping of genetic programming by using the standard built-in API of DOM-parsers for maintaining and manipulating genetic programs;

iv. OS neutrality of parsers;
v. Algorithmic language neutrality of DOM-parsers;
vi. Inherent Web-compliance of an eventual parallel distributed implementation of genetic programming.

```
<IF-THEN-NA>
   <COND-THEN-NA>
     <COND_TDistance>
       <VAR_TDistance>Peer_d</VAR_TDistance >
       <OPER TDistance>LE</OPER_TDistance >
       <CONST TDistance>20</CONST_TDistance >
     </COND_TDistance >
   </COND-THEN-NA>
   <THEN> ... </THEN>
   <NA>   ... </NA>
</IF-THEN-NA>
```

Fig. 4. Fragment of XML representation of sample stimulus-response rule.

3.1.3 Genetic operations

Binary tournament selection is employed; a robust, commonly used selection mechanism, which has proved to be efficient and simple to code. Crossover operation is defined in a strongly typed way in that only the nodes (and corresponding sub-trees) of the same data type (i.e. labelled with the same tag) from parents can be swapped. Sub-tree mutation is also allowed in a strongly typed way in that syntactically correct sub-tree replaces a random node in a genetic program. The routine refers to the type of node it is going to currently alter and applies a randomly chosen rule from the set of applicable rules as defined in the grammar of strongly-typed genetic programming. The transposition mutation also operates on a single genetic program by swapping two random nodes having the same data type.

3.1.4 Breeding strategy

We adopted a homogeneous breeding strategy in which the performance of a single genetic program cloned to all the predator agents is evaluated. Anticipating that the symmetrical nature of a world populated with identical predator agents is unlikely to promote any specialization in their behavior, we consider the features of such a homogeneous multi-agent society as (i) adequate to the world and (ii) consistent with our previously declared intention to create a robust and scalable multi-agent system.

3.1.5 Fitness functions

The fitness F measured for the trial starting with a particular initial situation is evaluated as the length of the radius vector of the derived agents' behavior in the virtual energy-distance-time space as:

$$F = \sqrt{dE_A^{\,2} + D_A^{\,2} + T} + K_c \times C \qquad (1)$$

where dE_A is the average energy consumption during the trial, D_A is the average distance to the prey by the end of the trial, and T is the elapsed time of the trial. C is the complexity of the agents' genetic representation in tree nodes, and K_C (equal to 0.1) is the scaling coefficient of the penalty imposed for complex genetic representations of the agents. Actually, T is an especially prime term in Eq.1, and therefore we may regard the fitness function as the function of the time needed for the predators to capture the prey. The trial is limited to 300s of "real" time or to the time the prey is captured; and with a sampling rate of 500ms it is simulated with up to 600 time steps. Smaller fitness values correspond to better performing predator agents.
The selection pressure, which favours more parsimonious agents' representations, is introduced as a measure to reduce the bloat in GP. The bloat (or the uncontrolled growth of genotypic representations during an evolutionary run) drastically reduces the computational performance of the implementation. The quantities dE_A and D_A are averaged over all predator agents. The energy consumption estimation dE for each predator agent takes into account both the basal metabolic rate and the energy consumption for motion as follows:

$$dE = E_{BMR} \times T + E_M \times D \tag{2}$$

where E_{BMR} is the basal metabolic rate, equal to 0.05 units per second, and E_M is the energy consumption for moving activities equal to 0.01 units per mm traversed during the trial. The trial is limited to 300 s of "real" time or to the time the prey is captured; and with a sampling rate of 500 ms it is simulated with up to 600 time steps. Smaller fitness values correspond to better performing predator agents. Notice that the agents are explicitly rewarded for capturing the prey (for minimizing the elapsed time of the trial) rather than for demonstrating surrounding behavior, which might eventually be needed to capture the prey. Surrounding, being discovered through simulated evolution, should emerge from the simply defined perception and moving abilities of the agents.
In order to obtain more general solutions to the problem the fitness of each genetic program is evaluated as an average of the fitness measured over 10 different initial situations. However, based on empirically proven data that in the initial stages of evolution agents are hardly able to successfully resolve more than a few (out of 10) initial situations, in order to enhance the computational performance of strongly-typed genetic programming, we applied an evaluation of the fitness function (Miller & Goldberg, 1995). The number of initial situations used to evaluate genetic programs in a population gradually increases with the evolution of the population. Starting from 4 for the first generation of each run, the number of situations is revised (until it reaches the value of 10 initial situations) on completion of each generation and it is set to exceed 2 the number of situations successfully solved by the best-of-generation genetic program. Given that with additional initial situation(s) they have to resolve, the agents would perform either better or, more likely worse, the fitness of the best-of-current generation could be occasionally somewhat worse than fitness of the best genetic program of the previous generation. Therefore, it is reasonable to anticipate non-monotonous fitness convergence characteristics of strongly-typed genetic programming.

4. Perceptual noise model

Since the purpose of our work is to investigate the relationship between the robustness of evolved surrounding behavior and the uncertainty of the predator agents, we introduce a perceptual noise model to their sensory abilities; distance to the prey and to the closest agent (i.e. Prey_d and Peer_d) and bearing of the prey and of the closest agent (i.e. Prey_a and Peer_a). In our model the perceptual noise term is added to the variable of sensory abilities of the predator agents as follows:

$$Peer\_d_{noise} = Peer\_d \pm Random\left(Peer\_d \times n\right) \tag{3}$$

$$Peer\_a_{noise} = Peer\_a \pm Random\left(Peer\_d \times n\right) \tag{4}$$

where *Peer_d*$_{noise}$ and *Peer_a*$_{noise}$ represent perceived distance and angle to the closest predator agent in its sight, and *Peer_d* and *Peer_a* mean exact distance and angle between these agents. In distance and angle to the prey, *Peer_d*$_{noise}$, *Peer_d*, *Peer_a*$_{noise}$ and *Peer_a* are replaced to *Pray_d*$_{noise}$, *Pray_d*, *Pray_a* and *Prey_a*$_{noise}$ respectively.
The second term *n* of each expression represents perceptual noise levels; the addition of the term makes the perception of predator agents uncertain. The noise increases in proportional to the distance between agents (i.e. *Peer_d* and *Prey_d*). In the other words, the further the peer predator agent and the prey agent are from a certain predator agent, more ambiguous the distance and the bearing from itself to the peer predator agent are; of course they are invisible if they move outside of its sight. This model that the location of a far-away object is perceived uncertainly is simple and natural. The perceptual noise reflects simple and usual supposition that it is hard to identify the exact location of a far-away object. The perceptual model of the predator agents is visualized in Figure 5.

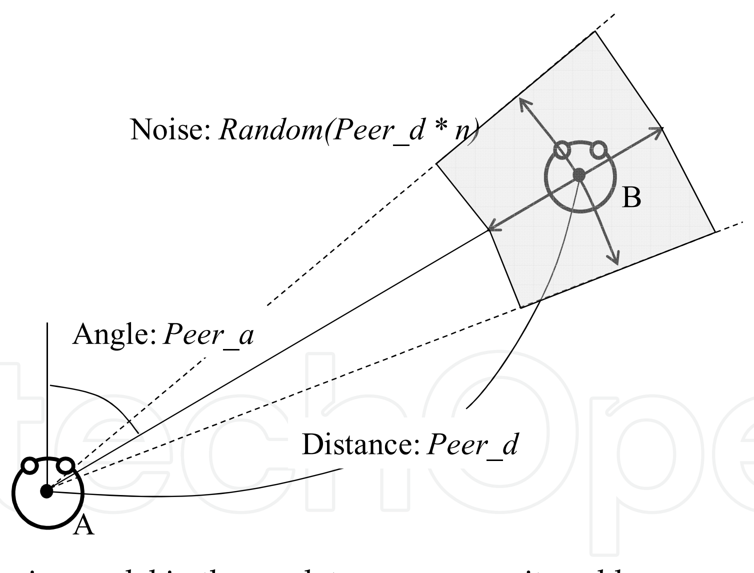


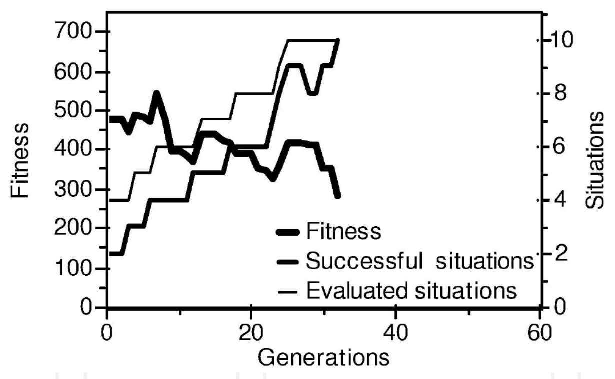
Fig. 5. Perceptual noise model in the predator prey pursuit problem.

Figure 5 illustrates a situation in which the predator agent A perceives a location of the peer predator agent B. If the agent A has a noiseless sensor, it can perceive the exact location information. However, because of the perceptual noise the predator agent A cannot precisely determine the exact location of the peer predator agent B. Therefore, the predator agent A randomly perceives that the peer predator agent B is located somewhere in the gray zone in Figure 3. For example if the noise level is 2.0%, the second term N of the formula takes the value under plus or minus 8; therefore, if the distance between these agents is 400mm (Peer_d = 400) and the angle between them is 30 degree (Peer_a = 30), the distance that the predator agent perceives results in the value from 392mm through 408mm, and also the angle results in the value from 22 degree through 38 degree. The perceptual noise model make the communication of predator agents instable, and therefore, it might result in inadequate surrounding behavior.

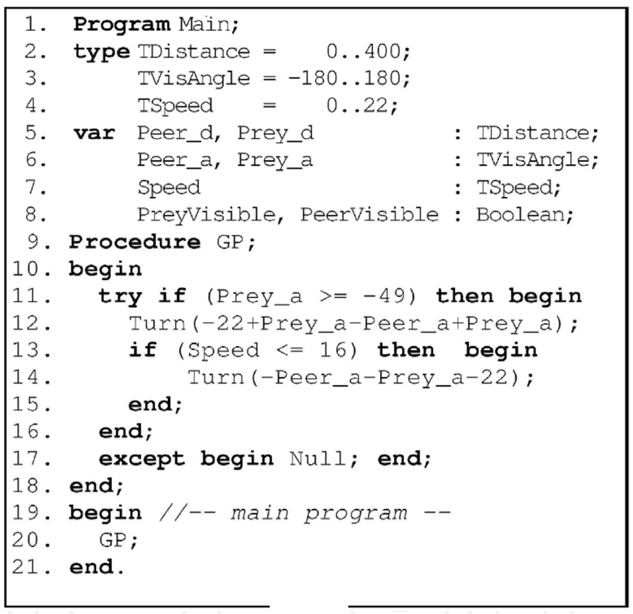
5. Empirical results and discussion

5.1 Evolution of the surrounding behavior of the predator agent

The values of parameters of strongly-typed genetic programming used in our simulation are summarized as follows: Population size was 600, Selection ration was 10%, Elitism was 1%, Mutation ratio was 2% and Trial interval was 600 steps. The fitness value of 300, employed as a termination criterion roughly corresponds to a successful team of predator agents that capture the prey by the middle of the trial of 600 steps.
The result, shown in Figure 6(a) indicates typical fitness convergence characteristic. Note that smaller fitness values correspond to better performing predator agents, since the fitness value is strongly affected by elapsed time of the trial as mentioned above (see Section 3.1.5). We consider these empirical results as an evidence of the very feasibility of applying a genetic programming paradigm for automatic design of autonomous agents capable of accomplishing complex tasks through local, implicit and proximity-defined interactions.



(a)



```
1.  Program Main;
2.  type TDistance =    0..400;
3.       TVisAngle = -180..180;
4.       TSpeed    =    0..22;
5.  var  Peer_d, Prey_d            : TDistance;
6.       Peer_a, Prey_a            : TVisAngle;
7.       Speed                     : TSpeed;
8.       PreyVisible, PeerVisible  : Boolean;
9.  Procedure GP;
10. begin
11.   try if (Prey_a >= -49) then begin
12.     Turn(-22+Prey_a-Peer_a+Prey_a);
13.     if (Speed <= 16) then  begin
14.         Turn(-Peer_a-Prey_a-22);
15.     end;
16.   end;
17.   except begin Null; end;
18. end;
19. begin //-- main program --
20.   GP;
21. end.
```

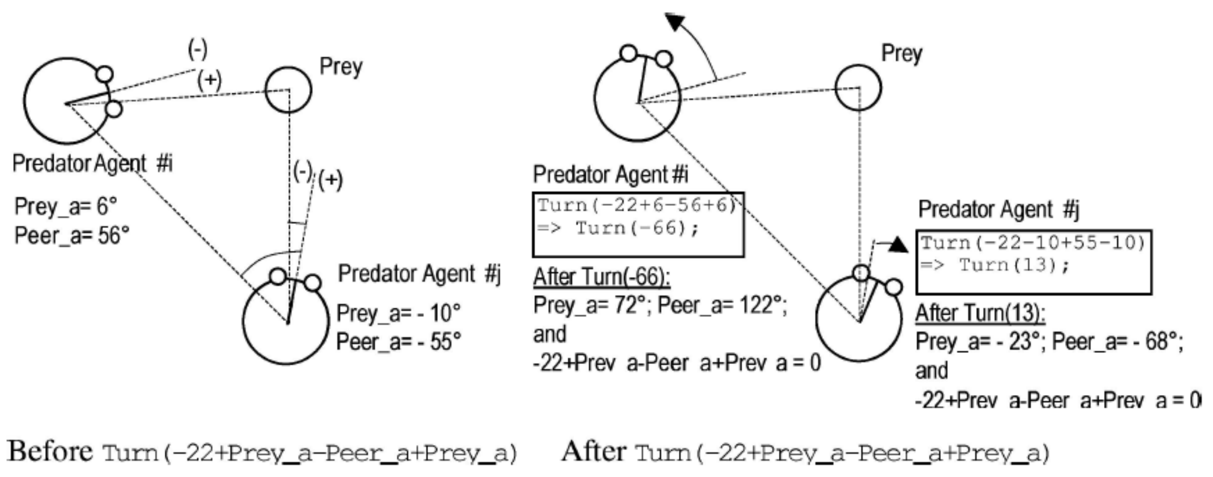
(b)

Fig. 6. Typical fitness convergence characteristic (a) and human-readable representation of sample best-of-run genetic program (b).

A human-readable representation of a sample best-of-run genetic program is shown in Figure 6(b). Figure 7 illustrates the execution of Turn(- 22 + Prey a - Peer a + Prey a) which is the most often executed command of the evolved solution (Figure 6(b), Line 12). The sensory feedback involved in computing the turning angles implies that agents orient themselves towards the directions which ensure that the perception variables Prey_a and Peer_a comply with the equation "- 22 + Prey_a – Peer_a + Prey_a = 0." Moving in these directions tends to separate the closest agents away and yields a characteristic chase of the

prey from the two opposite sides of the world when only two agents are involved. When more than two agents simultaneously execute the same command (a situation which is not elaborated in the figure) their team perform a surrounding approach to the prey.
The traces of the entities in the world for one of the 10 initial situations are shown in Figure 8. Agents employ a basic model of implicit interactions—only the distance and the bearing of the closest agent (and the prey) are perceived. The prey is captured in 118 simulated time steps (top). Large white and small black circles denote the predator agents in their initial and final position, respectively. The small white circle indicates the prey, initially situated in the center of the world. The numbers in rectangles show the timestamp information. The emergence of the following behavioral traits of predator agents is noticeable (each agent is governed by the sample best-of-run genetic program):

- Switch from greedy chase into surrounding approach (Agent #2, time step 65);
- Zigzag movement, which results in a lower chasing speed indicating "intention" to trap the prey (Agent #1, following time step 40), and
- Surrounding approach (agents #0, #2 and #3) at the final stages of the trial.


(-)
(+)
Prey
Predator Agent #i
Prey_a= 6°
Peer_a= 56°
(-) (+)
Predator Agent #j
Prey_a= - 10°
Peer_a= - 55°
Prey
Predator Agent #i
After Turn(-66):
Prey_a= 72°; Peer_a= 122°;
and
-22+Prev a-Peer a+Prev a = 0
Predator Agent #j
After Turn(13):
Prey_a= - 23°; Peer_a= - 68°;
and
-22+Prev a-Peer a+Prev a = 0
Before Turn(−22+Prey_a−Peer_a+Prey_a)
After Turn(−22+Prey_a−Peer_a+Prey_a)

```
Turn(-22+6-56+6)
=> Turn(-66);
```

```
Turn(-22-10+55-10)
=> Turn(13);
```

Fig. 7. Orientation of the Predator Agents before (left) and after the execution of command Turn(- 22 + Prey a - Peer a + Prey a) (right), respectively.

Figure 9 explains the zigzag movement as demonstrated by Agent #1 illustrated in Figures 8. Agent #1 periodically turns towards the alternatively becoming "visible" closest peers Agent #0 (left) and Agent #3 (right), which results in the characteristic zigzag movement. Black circles inside and below the Agent #1 indicates the position of the agent at the most recent consecutive moments.
Although such basic model offers the benefits of simplicity and scalability, the following issues related to the feasibility of applying the basic model in real-world applications remain still open: How much does the perceptual noise affect the evolved surrounding behavior? Can the predator agents capture the prey well even in noisy environment? Taking into consideration that the real-world applications indeed suffer from some mechanical and electrical noises, our model should involve some kind of countermeasures against these noises. Does the model acquire the robustness to the noises via the evolutionary approach? We focus our attention on the evolution of the surrounding behavior in a noisy multi-agent system.

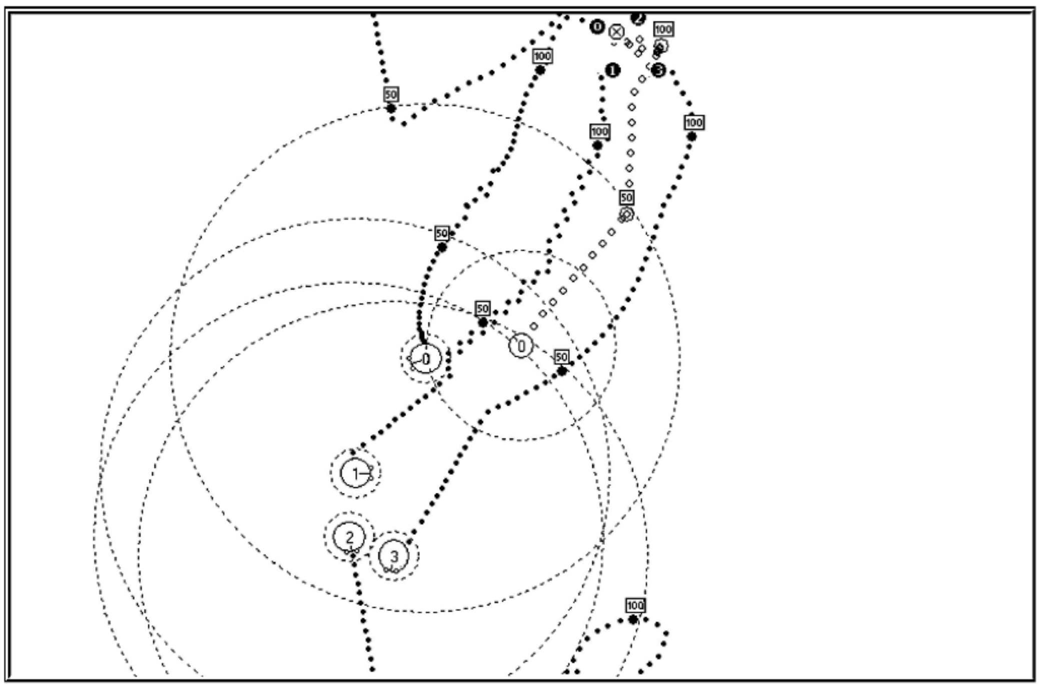
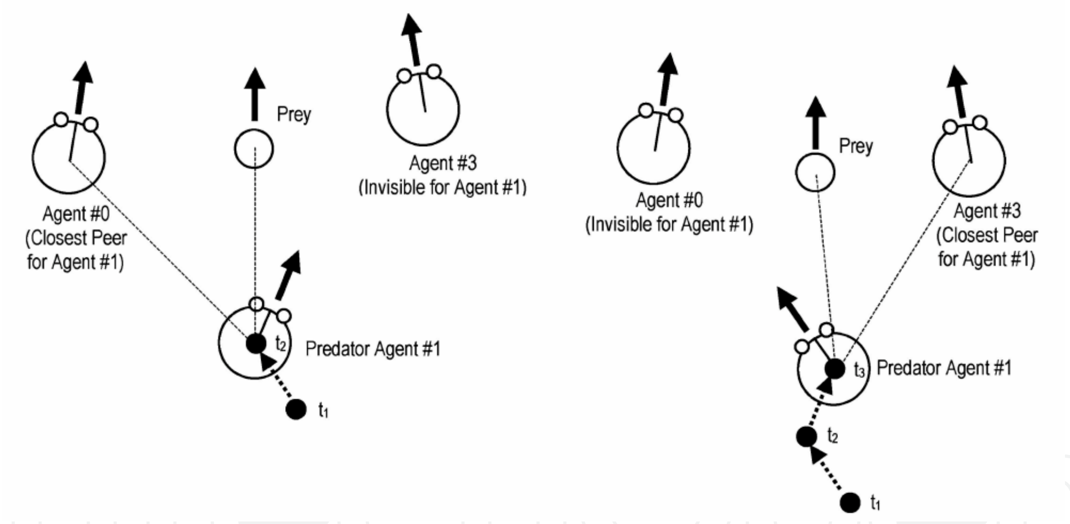


Fig. 8. Traces of the entities with predator agents governed by the sample best-of-run genetic program.



Prey
Agent #3 (Invisible for Agent #1)
Agent #0 (Closest Peer for Agent #1)
Predator Agent #1
t2
t1
Agent #0 (Invisible for Agent #1)
Prey
Agent #3 (Closest Peer for Agent #1)
t3
Predator Agent #1
t2
t1

Fig. 9. Explanation of the zigzag movement, which results in a lower chasing speed of the predator Agent#1 as illustrated in Figure 8.

5.2 Evolution in noisy multi-agent system

Since the predator agents with perceptual noise cannot perceive the exact location information of other agents, they might not be cooperated well each other; that is to say, their coordinated surrounding behavior would be poor. The quantitative effect of the perceptual noise on the coordinated surrounding behavior is still unknown. Therefore, we investigate the relationship between the fitness of the predator agents and the perceptual noise levels, and moreover we attempt to verify the supposition that the robustness of the predator agents behavior is related to an environment in which the evolutionary process by genetic programming runs.

5.2.1 The evolved surrounding behavior of the predator agents suffering from the perceptual noise

We evolved a surrounding behavior of predator agents with noiseless perception, and then evaluated the evolved surrounding behavior to the predator agents suffering from the perceptual noise, in order to investigate how much the surrounding behavior evolved in noiseless environment is affected by the perceptual noise. The levels of perceptual noise were between 0% and 3.0% in incremental of 0.5%. The fitness of the evolved surrounding behavior was different with every evaluation because of the randomness of perceptual noise. We conducted the evaluation 50 times. Figure 10 shows the average of the results.

The obtained results indicate that the fitness was worse almost linearly with the increase of perceptual noise levels, and also the success situations, which is the average number of (total 10) initial situations in which the predator agents successfully captured the prey, decreased with the increase of perceptual noise levels.

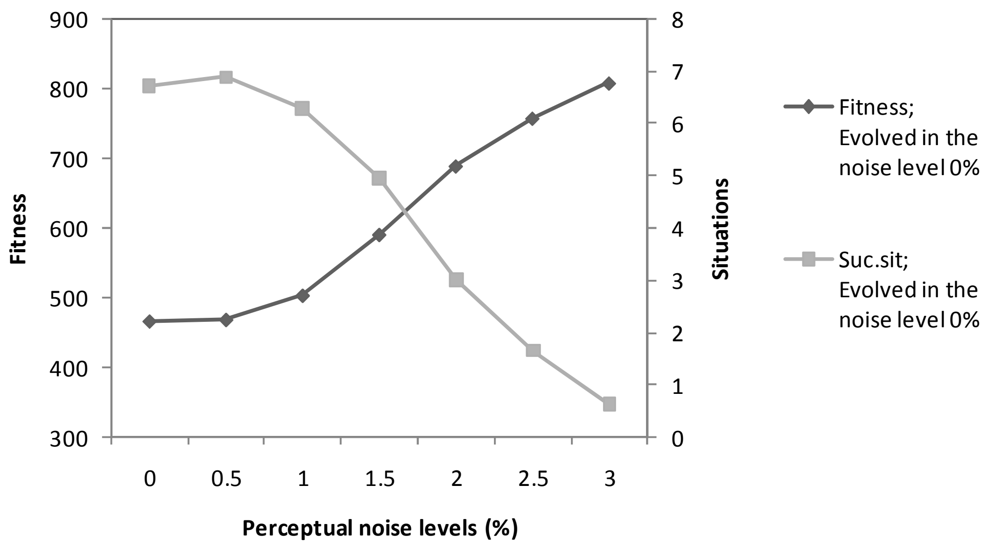


Fig. 10. Changes in the fitness and the success situations of the surrounding behavior in each noisy environment. Note that the surrounding behavior has been already evolved in noiseless environment.

This detrimental effect observed in the behavior of predator agents was most pronounced at the final stages of each trial as shown in Figure 11. The predator agents closed in on the prey at least to some extent but when they eventually enclose the pray, their erratic moving derived from perceptual noise made capturing the pray difficult.

5.2.2 Incremental evolution of the surrounding behavior in noisy environment

The surrounding behavior evolved in noiseless environment through genetic programming did not worked well in each noisy environment. Taking into account that genetic programming is a technique to automatically design agents' behavior without providing explicit domain-specific knowledge about how to achieve a task (Angleine, 1994), we might develop more robust surrounding behavior to noisy environment; in other words, the surrounding behavior involving the solution to uncertainty in noisy environment might be acquired through the interaction between genetic programming and noisy environment without incorporating the explicit knowledge of perceptual noise into the predator agents.

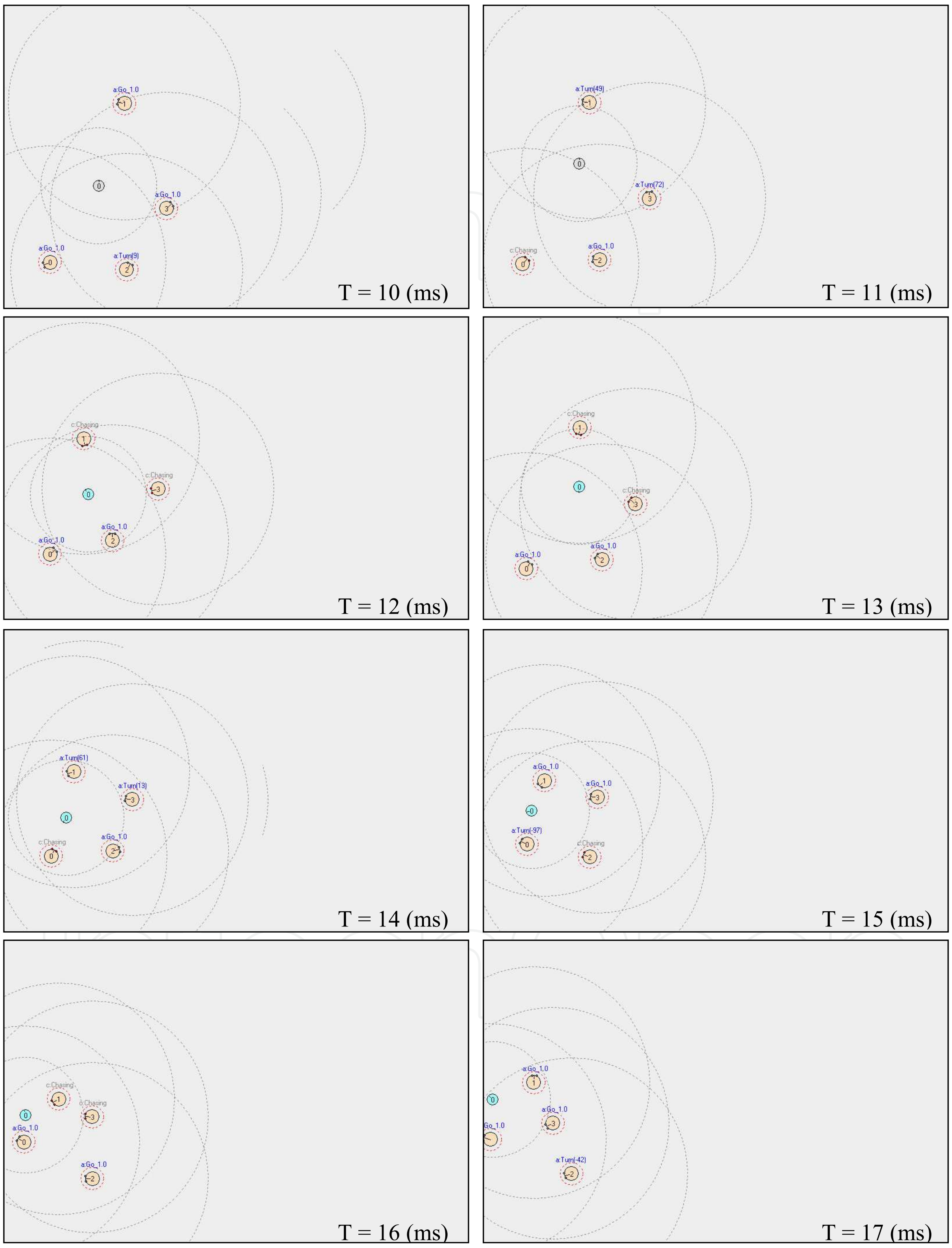


Fig. 11. Typical failure of the evolved surrounding behavior in noisy environment.

In order to investigate the above assumption, we evolved again the evolved behavior in noisy environment with noise rate of 0% and 2%, respectively. The initial population includes some of the best-of-run behavior program evolved in noiseless environment, because the randomly created initial population can hardly adapt in complicated and uncertain noisy environment.

As Figure 12 shows, the results verify that both the fitness and the success situations of the re-evolved surrounding behavior in noisy environment were better than those of the re-evolved in noiseless environment. The behavior evolved in the noise level 2% features a moderate degradation when applied in environments with up to the perceptual noise level 1.5%, and eventually, both the fitness and capture rate converge at similar value of the noise level 2.5%. The shape of graph shows that while the fitness of the surrounding behavior re-evolved in noiseless environment (i.e. the perceptual noise level 0%) seems linear with respect to the perceptual noise level, that of the surrounding behavior program evolved in noisy environment (i.e. the perceptual noise level 2%) seems to draw a sigmoid curve. This indicates that the fitness tend to keep well until a certain threshold (in the behavior evolved in the perceptual noise level 2%, the threshold is 1.5%), hence, the surrounding behavior is more robust on a specific noise level range.

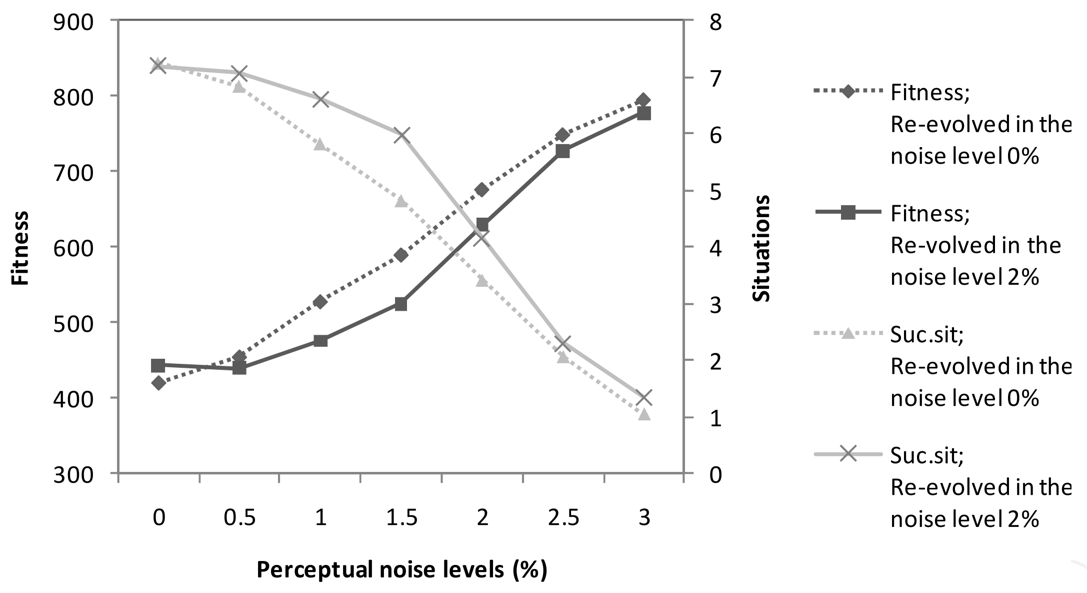
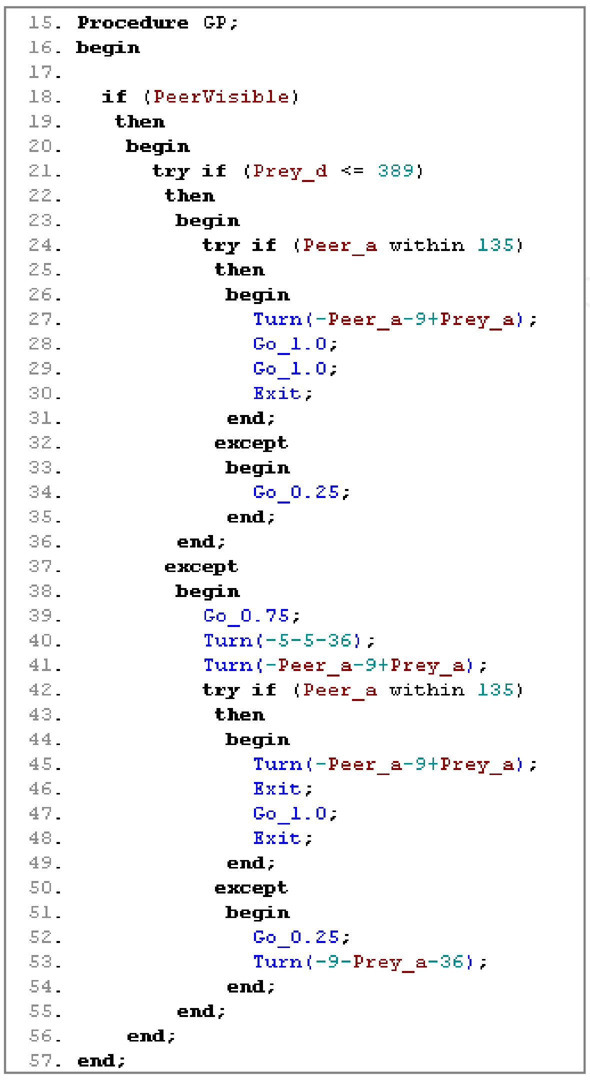


Fig. 12. Changes in the fitness and the success situations of the re-evolved surrounding behavior with the perceptual noise 0% and 2%.

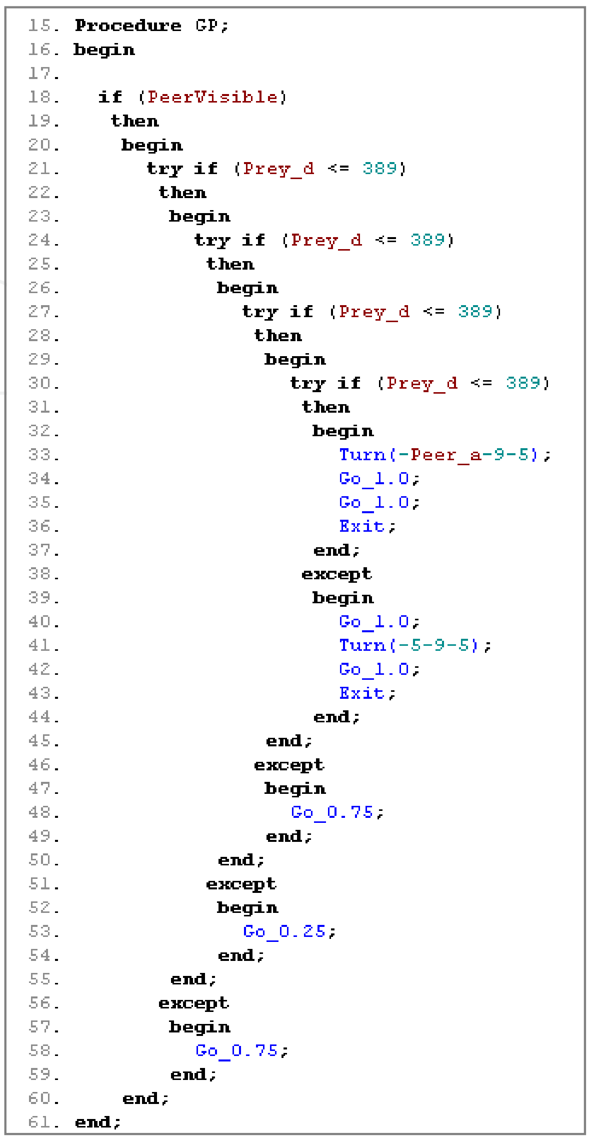
Through the evolution in noisy environment, we could generate a behavior whose fitness is stable until a certain threshold. It seems to be natural conclusion, but what improves the robustness of the behavior? We suppose one reason is that the behavior program obtained a specialized structure to be robust to the perceptual noise. Figure 13 show a comparison of the genetic programs evolved in noiseless environment and in noisy environment.

Unfortunately, we were unable to discover such a structure in a program because the logic of automatically evolved code is hardly understandable by human. However, some interesting points can be discussed as follows: The instruction Turn() is present the program evolved in noisy environment (Figure 8, left) not that often compared to the program evolved in noiseless environment (Figure 8, right); and the variables for perceptual



```
Procedure GP;
begin

 if (PeerVisible)
  then
   begin
     try if (Prey_d <= 389)
      then
       begin
         try if (Peer_a within 135)
          then
           begin
             Turn(-Peer_a-9+Prey_a);
             Go_1.0;
             Go_1.0;
             Exit;
           end;
          except
           begin
             Go_0.25;
           end;
       end;
      except
       begin
         Go_0.75;
         Turn(-5-5-36);
         Turn(-Peer_a-9+Prey_a);
         try if (Peer_a within 135)
          then
           begin
             Turn(-Peer_a-9+Prey_a);
             Exit;
             Go_1.0;
             Exit;
           end;
          except
           begin
             Go_0.25;
             Turn(-9-Prey_a-36);
           end;
       end;
   end;
end;
```



```
Procedure GP;
begin

  if (PeerVisible)
   then
    begin
      try if (Prey_d <= 389)
       then
        begin
          try if (Prey_d <= 389)
           then
            begin
              try if (Prey_d <= 389)
               then
                begin
                  try if (Prey_d <= 389)
                   then
                    begin
                      Turn(-Peer_a-9-5);
                      Go_1.0;
                      Go_1.0;
                      Exit;
                    end;
                   except
                    begin
                      Go_1.0;
                      Turn(-5-9-5);
                      Go_1.0;
                      Exit;
                    end;
                end;
               except
                begin
                  Go_0.75;
                end;
            end;
           except
            begin
              Go_0.25;
            end;
        end;
       except
        begin
          Go_0.75;
        end;
    end;
end;
```

Fig. 13. The comparison of genetic programs evolved in noiseless (left) and noisy (right) environments, respectively.

information (i.e. Peer_a, Peer_d, Prey_a and Prey_d) rarely appear in the left-, conversely to the right program shown in Figure 8. In our mode, such variables are perturbed directly by the perceptual noise. Consequently, in the case of high noise level, it is considered that a program involving many such variables is more difficult. As a result, a program evolved in noisy environment might be evolved to limit the reliance on such variables.

6. Conclusion

We presented the result of our work on the use of genetic programming for evolving surrounding behavior of agents situated in inherently cooperative environment. We use the predator-prey pursuit problem to verify our hypothesis that relatively complex surrounding behavior may emerge from simple, implicit, locally defined, and therefore - scalable interactions between the predator agents. Proposing perceptual noise model of the predator agents we investigated the relationship between the evolved surrounding behavior and the perceptual noise. We demonstrated that relatively complex, surrounding behavior emerges even from the simple, basic model of implicit, proximity defined interactions among the agents. We observed the relatively simple motion of the predator agents in the direction away from the closest predator agents yields emergent collective behavioral traits of predator agents such as (i) a characteristics zigzag movement, which results in a lower chasing speed indicating "intention" to trap the prey, and ultimately, (ii) a surrounding of the prey. Although the above surrounding behavior was performed efficiently in noiseless environment, the performance of it was worse as the perceptual noise level increased. We evolved the behavior again in each of two different environment; noiseless environment and noisy environment, and compared the performance of these types of behavior. The behavior evolved in noisy environment get better performance than that evolved in noiseless environment.

In the future we are planning to incorporate evolvable rather than handcrafted escaping strategy of the prey as used in our current approach. We are also interested in enhancing the currently used perception and communication models into a model, which allows for predator agents to analyze the effects of their own actions on the reaction of the other agents in the world. We are planning to investigate both the survival value of such reflection and the robustness of the team of predator agents.

7. References

Angeline, P. J. (1994). Genetic programming and emergent intelligence, In: *Advances in Genetic Programming*, Kinnear, K. E. Jr. (Ed.), pp. 75-98, MIT Press, 0-262-11188-8, Cambridge, MA, USA

Benda, M.; Jagannathan, B. & Dodhiawala, R. (1986). On optimal cooperation of knowledge sources," In: *Technical Report BCS-G2010-28*, Boeing AI Center, Boeing Computer Services, Bellevue,WA.

Brooks, R. A. (1986). A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation*, Vol. 2, No. 1, pp. 14-23, 0882-4967.

Ferber, J. (1999). *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison-Wesley Longman: Harlow, 0201360489,

Forrest, S. (1991). Emergent computation: Self-organising, collective, and cooperative phenomena in natural and artificial computing networks, In: *Emergent Computation*, Forrest, S. (Ed.), pp. 1-11, MIT Press, 0-262-56057-7, Cambridge, MA, USA.

Haynes, T. & Sen, S. (1996). Evolving Behavioral Strategies in Predators and Prey, *Adaptation and leaning in multiagent systems*, pp. 113-126, Springer Verlag.

Haynes, T.; Wainwright, R.; Sen, S. & Schoenefeld, D. (1997). Strongly Typed Genetic Programming in Evolving Cooperation Strategies, *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 271-278, Morgan Kaufmann Publishers, Inc.

Holand, J. H. (1999). *Emergence: From Chaos to Order*, Perseus Books, 0738201421, Cambridge, 1999.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 0262111705, Cambridge, MA, USA.

Luke, S. & Spector L. (1996). Evolving Teamwork and Coordination with Genetic Programming, In Genetic Programming 1996: Proceedings of the First Annual Conference. Koza, J. et al (Ed.), pp. 141-149. MIT Press, 10884750, Cambridge, MA, USA.

Miller, B. L. & Goldberg, D. E. (1995). Genetic Algorithms, Tournament Selection, and the Effects of Noise, *Complex Systems*, Vol. 9, pp. 193-212, .

Montana, D. (1995). Strongly typed genetic programming, *Evolutionary Computation*, Vol. 3, No. 2 , pp. 199-230, MIT Press, 1063-6560.

Morgan, C. (1923). *Emergent Evolution*, Henry Holt and Co, 0404604684.

Morowitz, H. J. (2002) *The Emergence of Everything: How the World Became Complex*, Oxford University Press, 019513513X , New York, USA.

Parunak, H.; Van, D.; Brueckner, S.; Fleischer, M. & Odell, J. (2002) Co-X: Defining what agents do together, *Proceedings of the AAMAS 2002 Workshop on Teamwork and Coalition Formation*, Shehory, O.; Ioerger, T. R.; Vassileva, J. & Yen, J. (Eds.), pp. 62-69.

Tanev, I. (2003). DOM/XML-based portable genetic representation of morphology, behavior and communication abilities of evolvable agents, *Proceedings of the 8th International Symposium on Artificial Life and Robotics*, pp. 185-188.