STRATEGY CREATION, DECOMPOSITION AND DISTRIBUTION IN PARTICLE NAVIGATION: MEMORY MODULE

Ulaş Beldek[1,3], Kemal Leblebicioğlu[2,3]

[1] *Department of Electronic and Communications Engineering, Çankaya University, Ankara, Turkey*
[2] *Electrical and Electronic Engineering Department, METU, Ankara, Turkey*
[3] *Computer Vision and Intelligent Systems Research Lab, METU, Ankara, Turkey*
beldek@metu.edu.tr, kleb@metu.edu.tr

Abstract: In particle navigation problem strategy development is crucial. The difficulties encountered by the particles during their navigation tasks require different approaches in problem solving. One way to overcome the difficulties is to divide the problem into simple modules and develop solutions for these modules separately. Basically, two different modules are sufficient in addition to the main body to develop a successful solver. The first module (conflict module), which is developed by genetic programming, is used when the particles are in conflict. The second module (memory module) helps the particles to escape from local regions.

Keywords: Genetic Algorithms, Agents, Optimization, Path Planning, Robot Navigation, Rule-Based Systems.

1. INTRODUCTION

There are lots of problems where a strategy has to be developed in order to produce a feasible solution. Strategy generation by computer is a difficult task where a lot of effort should be spent. Genetic programming (GP) by its probabilistic nature is among the techniques (Angelina and Kinnear, 1996; Koza, 1992) that one can create satisfactory solutions for problems involving strategical decision-making. GP depends on nature's dynamics. The rule is simple; the entity that adapts itself to its environment better has more chance to continue living in the future (Koza, 1992). GP is applicable to almost any problem (Gary, *et al*., 1996; Gray *et al*., 1998; Tackett, 1993), if it is possible to define the problem in terms of its determinants (actions, variables, constants, operators, processes) and relations between these determinants. Multi-agent structures can be used to accomplish a group of tasks (Fogarty, *et al*., 1995). Sometimes cooperation and communication have to be provided between agents to obtain successful solutions (Brazier, *et al*., 2001; Van Eijk, *et al*., 2000; Baeg, *et al*., 1995). Agents can act simultaneously or in a special order to resolve the problems they encounter. Strategy generation for particle navigation necessitates dealing with many different tasks simultaneously. GP is a suitable environment to generate these agents (Werner and Dyer, 1992; Luke and Spector, 1996; Iba, 1998; Beldek, *et al*., 2001). Iba (1998) used some mathematical relations to obtain agent structures. Depending on the breeding technique an agent (strategy) is capable of navigating one particle (heterogeneous breeding) or whole particles (homogeneous breeding). For the same problem the agents are created from if-then statements and action types by Beldek, *et al.,* (2001). Both of the studies (Iba, 1998; Beldek, *et al*., 2001) weren't totally effective in dealing with all the scenarios. Finding a compact strategy is very hard or sometimes impossible since long time duration will be needed to train agents on many scenarios. Another bad aspect is the robustness of the final strategy. That is, the best

strategy may be inadequate in some of the situations. An idea that comes to mind is not to try to develop a compact strategy but to divide the tasks into sub-tasks, find procedures to overcome these sub-tasks and then combine the results to obtain a suitable solution (Benaroch, 2001). What Beldek, *et al*., (2002) has done is exactly based on this idea. The conflicts between particles are solved in a separate module. The resultant agent was successful in solving the conflicts but the particles were still unable to overcome the tasks completely because wall structures may deter the particles while they are trying to reach their destinations. So a new module will be designed for a particle to be able to escape from a local region during its voyage towards its destination. This module is supposed to make the particle remember its previous actions and move accordingly. The uses of the conflict module, standard navigation module and this memory module will be examined in different scenarios.

2. PARTICLE NAVIGATION PROBLEM

The robot (particle) navigation problem is supposed to take place in a 10x10 grid area. Figure 1 shows a typical scenario. All the cells in the grid area can take one object. Either the cell is occupied by a particle or there could be an obstacle inside the cell. The particles are labeled as 1, 2, 3 and 4 whereas the obstacles are designated as darker regions in the grid area. The particles are able to move up, down, left and right as long as the obstacles and the boundaries of the scenario allow. It is aimed to move the particles to some pre-defined locations in the scenarios. Some conflicting situations are encountered especially when two particles try to pass in opposite directions in very narrow regions.

3. GENETIC PROGRAMMING

3.1. Terminals and Operations

Some terminals and operations are designed in order to construct the chromosomes (individuals) to be used in GP. The terminals are chosen from movement types and the operations are chosen from 'if-then' statements. The individuals are constructed by combining operations with the terminals meaningfully creating a hierarchical tree-like structure. These are the strategies that orient the particles. The strategies are capable of moving the particles depending on the destination and the local environment information. In this study, the symbols for the terminals and operations and their meaning are as follows:

Terminals:

- **a:** go one step in the direction of your orientation and keep your orientation the same.
- **b:** go one step backward but don't change your orientation.
- **c:** turn 90 degrees clockwise and go one step
- **d:** turn 90 degrees counter-clockwise and go one step.
- **e:** do not move and do not change your orientation.
- **f:** make a random movement and change your orientation in the movement direction.
- **g:** move in the direction of your destination.

Operations:

- + (Takes two arguments): it returns the first argument if there is an obstacle in front of the particle being controlled; otherwise it returns the second argument.
- **A** (Takes two arguments): It returns the first argument if there is another moving particle in front of the particle being controlled; otherwise it returns the second argument.
- **B** (Takes two arguments): It returns the first argument if the controlled particle is in a region where two opposite sides of the particle is occupied by obstacles and other two opposite sides are free cells; otherwise it returns its second argument.
- **F** (Takes two arguments): It returns its first argument if the controlled particle and the destination of the same particle are lying in same perpendicular or horizontal alignment; otherwise it returns its second argument.
- **R** (Takes two arguments): It returns its first argument if the cell in front of the controlled particle is obstacle-free; otherwise it returns its second argument.

The terminals symbolized by 'g' moves a particle in the direction of its destination, so they enable the particle to go in diagonal directions, if possible (if the obstacles allow them to move). The trees (individuals) are rule-based strategies. In Figure 2, a typical tree structure is shown.

The movement of a particle controlled by the tree structure in Figure 2 can be described as follows: First of all, it is checked if there is another particle in front of the controlled particle. If this is true then the controller checks if the particle and its destination lay in the same horizontal or vertical alignment. If the second statement is true the particle moves in backwards but still facing the original direction. If the statement is false the particle moves in forward direction. In case the first statement is false, the controller checks if the particle is in a place where there are obstacles at two opposite sides and the other two opposite sides are free cells. If this statement is true, the particle makes a movement in the direction of its destination. If the statement is false, it turns in clockwise direction and goes one step.

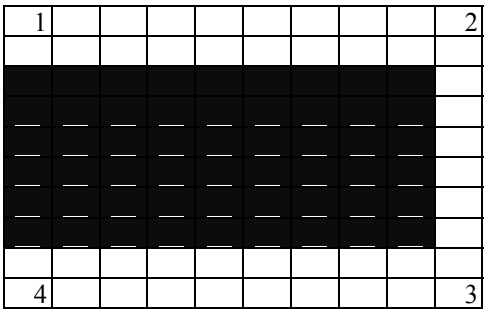

Fig. 1. A scenario example.

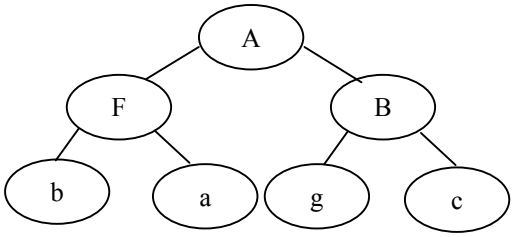

Fig. 2. A tree structure example.

This tree structure is represented as a string in MATLAB. The chain of command in the tree is from top to bottom and from left to right similar to LISP programming. E.g., the tree structure in Figure 2 can be represented as the string **'AFbaBgc'** in our simulations.

3.2 GP Application

Crossover, reproduction and regeneration are the genetic operations used in order to obtain a new generation. The population size is taken as 100 in our simulations. 80% of the new individuals in the next generation are produced by the crossover operation, 10% are reproduced and 10% are regenerated. The elitist method is also used in reproduction. In a crossover operation the probability of the crossover point being a terminal is adjusted to be between 15% and 25%.

Three different criteria are considered in order to assign a fitness for an individual. A large fitness value must be assigned to the individual that brings all the particles to their goals. Also a reward must be given according to the duration in which an individual completes all its tasks. Secondly, a respectable fitness value must be assigned to the individual that brings some of the particles to their destinations. Naturally this new fitness value must be smaller than that of an individual that brings all the particles to their destinations. Thirdly, an individual carrying a particle close to its destination must also be assigned a sufficient fitness value, but this fitness must be smaller than previously described fitness values. This fitness assignment has the same logic as the fitness assignment procedure used by Beldek, *et al.* (2001).

If there is more than one scenario, an average fitness must be assigned for an individual. So fitness values of independent scenarios are added and the sum is divided to number of scenarios in order to obtain an average fitness value for an individual

3.3. Homogeneous and Heterogeneous Breeding

One can use two different control approaches to orient the particles in a maze (Iba, 1998). First way is to use different strategies to control different particles. A GP chromosome with this strategy has distinct parts associated with each particle. This approach is called heterogeneous breeding. In the second approach all the particles are controlled by the same strategy (homogenous breeding).

4. SIMULATION STUDIES

4.1. Modular Approach

One way to overcome the difficulties in particle navigation problem is to use a modular approach. Dealing with the challenging parts in different modules, resolving these parts and combining the solutions obtained in these parts will be beneficial in solving the whole problem. Each module will consist of a single agent. Each agent will be effective on one type of challenging situation and particles will be controlled and moved by these agents when necessary. So strategies will depend on situations of particles, not on particles themselves. This task decomposition is supposed to provide efficiency and robustness to particle navigation.

Based on the information given by Beldek, *et al.* (2001) and Beldek, *et al.* (2002), modular approach has been applied for two different difficulties. The first difficulty arises when two particles try to pass in opposite directions in a narrow region where only one particle can pass at a time. This situation is called as the conflict. The simulation results for this type of difficulty are given in the study (Beldek, *et al.,* 2002), where some scenarios involving two particles in conflict are produced. A homogeneous agent, which is effective in eliminating the conflicts, is developed by using GP search. The tree structure of this conflict agent is shown in Figure 3. Secondly this conflict agent is used to develop a new agent, which is supposed to control the particles in standard situations. In order to develop this new agent (named as the standard navigation agent from now on), a number of scenarios involving four particles, are constructed, as well. These scenarios are such that the particles may encounter conflicts while they are moving to their destinations. The pre-developed conflict agent was used for controlling the movements of the particles when they are in conflict in the new scenarios. The GP search produced the standard navigation agent, which controls the particles when the particles are not in conflict. The

standard navigation agent developed is shown in Figure 4. Simulation results of the study (Beldek, *et al.,* 2002) show that all the conflicts are removed, but not all the navigation tasks are accomplished. Cooperation of two agents is not sufficient to complete all the navigation tasks. It is observed that the main problem was the blockage of the particles by the thick wall structures formed from uninterrupted placement of obstacles. Standard navigation agent sometimes becomes unsuccessful and the particles are blocked by wall structures through their way to destinations. As a result, they may be imprisoned in local areas. Constructing a new module to remove this blockage problem is the scope of our study.

The operations (if-then statements) are able to gather and process information from neighboring cells around the particles (operations '+', 'A', 'B', 'R') and from alignment of current position and destination of the particles (operation 'F'). So data about the cells, which are far away from the particles, is nearly impossible to be processed by the operations defined earlier. This is the main reason why the standard navigation agent is sometimes ineffective. It is not possible for the particles to escape from local regions in most of the cases. In order to overcome this difficulty, complicated operations might be proposed and another standard navigation agent could have been developed by GP. But we have preferred to utilize GP to construct strategies from simple operations. A simple idea is to process information from previous actions of particles. This necessitates a memory module. So a memory module is proposed in order to solve blockage problem.

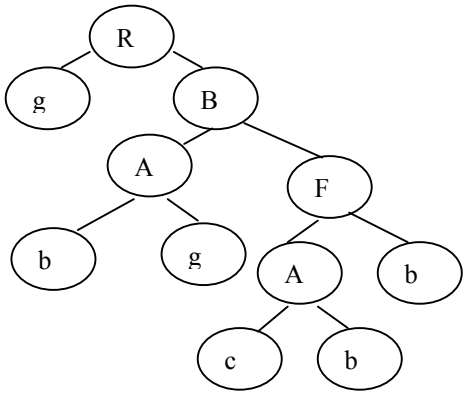

Fig. 3. Tree structure of conflict solving agent.

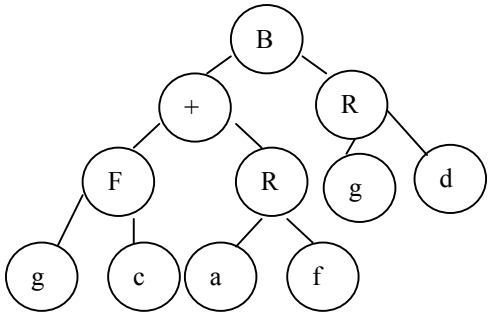

Fig. 4. Tree structure of standard navigation agent

4.2. Memory Module

Memory module is supposed to be used if a particle is not able to leave some region of the maze for a long time. Its control strategy depends on continuous application of intelligent movements depending on past observations and experiences of particle in order to rescue the particle from local regions. This is done as follows. For each particle, coordinates of previous (10) positions are kept in a vector. The average of these positions are compared with the current position of the particle. If the Euclidean distance between the current position and the average is smaller than a threshold (it is taken as 3), the particle is assumed trapped in a local region. Then the memory module takes control. Module provides a sequence of actions to enable the particle to escape the local region. A ε-neighborhood of the particle (designated as two square units towards each direction from the particle) is declared as a forbidden zone for the particle. The closest empty maze location out of this forbidden zone is established as a new temporary destination for the particle. The particle is supposed to move to this new destination until it escapes the forbidden zone. Whenever the particle arrives the temporary destination, standard navigation agent takes the control of the particle back. Standard navigation agent is not allowed to carry a particle to its forbidden zone again. But, if two particles are in conflict, then the conflict agent takes precedence and the particles are allowed to enter their forbidden zones if necessary. When the conflict is resolved, if one of the particles is in its forbidden zone, memory module takes the duty of moving the particle out of the forbidden zone. Otherwise, the standard navigation agent starts control ling the particle.

4.3. Memory Module Simulation Results

Two different simulations have been carried out in order to observe the effect of memory module on the overall performance. In the first simulation the scenarios are the same as the ones used to develop the standard navigation agent in the study (Beldek, *et al.*, 2002) (8 scenarios, each involving four particles). It is aimed to move the particles to their destinations by using standard navigation agent, conflict module and memory module, whenever they are necessary. Since memory module requires a long time to produce a decision, the number of time steps in the simulation is chosen as 100. It is allowed to use memory module after the first 20 steps in order to allow the agents to reach a solution without using the memory module. The forbidden zone for a particle is declared as permanent; so the particle is prohibited to enter the forbidden zones except during conflict module utilization. In Figure 5, first scenario is shown. In this scenario, it is aimed to interchange places of particles 1 with 2 and 3 with 4. These tasks are accomplished in 55 steps completely. Similar results are obtained in the other scenarios. All of the

particles have reached their destinations in all the eight scenarios.

In the second simulation, six different scenarios (each having two particles) are constructed. The scenarios contain trap regions to confuse the particles through their voyages to their destinations. It is vital for the particles to get out of these regions. The forbidden region of a particle is designated as a temporary forbidden zone. These zones are opened to the access of the associated particles in 40 steps after they are announced as forbidden zones. Entry of a particle in a trap region may cause a particle to assign all the free cells inside and outside of the trap region as belonging to the forbidden zone, which may result in closing the passages to destinations and imprisonment of a particle in that local area. Figures 6 and 7 show the first two scenarios. For the scenario in Figure 6 it is aimed to bring the particle 'A' to the initial position of the particle 'B' and the particle 'B' is supposed to go to cell 'H'. It is observed that both of the particles reach their destinations but the particle 'A' first enters the trap area and then escapes out of the region. For scenario shown in Figure 7, it is aimed to move the particle 'B' to the initial location of the particle 'A' and the particle 'A' is supposed to go to cell 'H'. Both of the tasks are accomplished at the end of the simulation. Additionally, in remaining four scenarios all the navigation tasks are also accomplished.

5. CONCLUSION AND FUTURE STUDIES

In this work, particle navigation problem is studied. It was previously observed that controlling every particle by a different strategy is insufficient for some complex situations (Beldek, *et al*., 2001). A new approach for removing the difficulties is proposed by Beldek, *et al*., (2002). There, two modules were developed for two different classes of difficulties (conflict agent and standard navigation agent). In this study, in addition, a memory module is proposed and utilization of three modules simultaneously in some scenarios is examined. It is observed that using these three modules whenever necessary helps removing difficulties and better results are obtained when compared to previous study (Beldek, *et al*., 2001.)

It should be remarked that the memory module is simply an application of a set of intelligent actions in order to rescue the particle from local regions. This is nothing to do with GP. On the other hand, standard navigation agent and conflict agent are developed by GP.

Switching of the control between modules is done automatically according to the difficulties encountered in the scenarios. This switching procedure may be developed as an agent by a GP search.

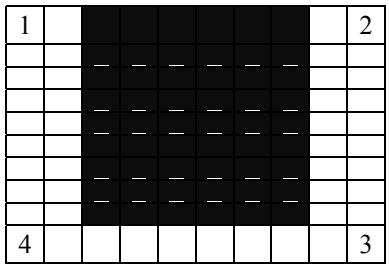

Fig. 5. First scenario in the first simulation.

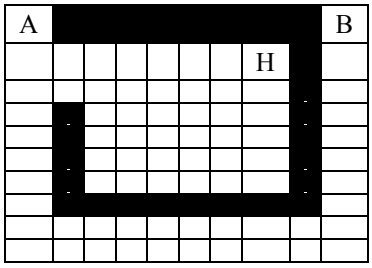

Fig. 6. First scenario in the second simulation.

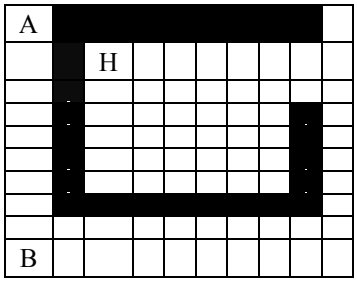

Fig. 7. Second scenario in the second simulation.

In the future, it is planned to develop a compact agent by GP, which will include all the necessary modules (normal navigation agent, conflict agent, memory module, switching between the modules).

REFERENCES

Angelina, P.J. and K.E. Kinnear (1996). *Advances in Genetic Programming II*, MIT Press, Cambridge MA.

Baeg, S.C., S.K. Park, J.M. Choi, M.W Jang and Y.H Lim (1995). Cooperation in Multi-agent Systems, In: *Intelligent Computer Communications (ICC '95)*, pp 1-12.

Beldek, U., K. Leblebicioğlu, U. Halıcı (2001). Genetic Programlama Yöntemiyle Robot Yönlendirme Probleminin Çözümlenmesi ve Matlab Uygulaması. In: *Otomatik Kontrol Ulusal Toplantısı, TOK 2001* (İ. Yüksel, M. Şengirgin, G. Şevkat, Ed)., pp 91-98, Endüstri ve Otomasyon-Eksen Yayıncılık, Bursa.

Beldek, U., K. Leblebicioğlu, U. Halıcı (2002). Robot Yönlendirme Probleminin Genetik

Programlama ile Çözümünde Yeni Yaklaşımlar. In: *10. Sinyal İşleme ve İletisim Uygulamaları Kurultayı, SİU 2002* (E. Panayırcı, Ed). Vol 1, pp 278-285, Ayna Reklam Tanıtım, İstanbul.

Benaroch, M. (2001) Declarative representation of strategic control knowledge. *International Journal of Human-Computer Studies*, **55 (6)**, 881-917.

Brazier F.M.T., C.M. Jonker, J. Treur, N. J.E. Wijngaards (2001). Deliberative Evolution in Multi-Agent Systems*, International Journal of Software Engineering and Knowledge Engineering*, **11 (5)**, 559-581.

Fogarty, T.C., L. Bull and B. Carse (1995). Evolving Multi Agent Systems. In: *Genetic Algorithms in Engineering and Computer Science* (G. Winter, J. Periaux, M. Galan, P. Cuesta, Ed.)., pp. 3-22. John Wiley and Sons, Chichester.

Gray, G.J., D.J.M. Smith, Y. Li, K.C. Sharman and T. Weinbrenner (1998). Non-linear Model Structure Identification Using Genetic Programming, *Control Engineering Practice*, **6**, 1341-1352.

Gray, G.J., D.J.M. Smith, Y. Li and K.C. Sherman (1996). Structural System Identification Using Genetic Programming and Block Diagram Oriented Simulation Tool, *Electronic Letters*, **32 (15)**, 1422-1424.

Iba, H. (1998) Evolutionary Learning of Communicating Agents, *Journal of Information Sciences*, **108**, 181-205.

Koza, J. (1992). *Genetic Programming, On the Programming of Computers by means of Natural Selection*, MIT Press, Cambridge MA.

Luke, S. and L. Spector (1996). Evolving Teamwork and Coordination with Genetic Programming, In: *Proceedings of the First Annual Conference on Genetic Programming (GP-96)* (J. Koza, Ed)., pp 150-156, MIT Press, Cambridge MA.

Tackett, W.A. (1993). Genetic Programming for Feature Discovery and Image Discrimination. In: *Proceedings of 5th International Conference on Genetic Algorithms ICGA-93* (S Forrest, Ed.)., pp. 303-319. Morgan Kaufmann, San Francisco CA.

Van Eijk, R.M., F.S. De Boer, W. Van Der Hoek J.J. Ch. Meyer (2000). Open Multi-Agent Systems: Agent Communication and Integration. In: *Intelligent Agents VI, Proceedings of 6th International Workshop on Agent Theories, Architectures, and Languages (ATAL'99)* (N.R. Jennings, Y. Lesperance, Ed.)., pp. 218-232, Springer-Verlag, Heilderberg.

Werner, G.M. and M.G. Dyer (1992). Evolution of communication in artificial organisms. In: *Artificial Life* (C.G. Langton, C. Taylor, J.D. Farmer, S. Rasmussen, Ed.). Vol 2, pp. 659-687, Addison-Wesley, New York.