Simultaneous Development of Antagonistic/Cooperative Paths Using an Ant Algorithm

*Chris Rook** *D.A. Harrison**

* School of Computing and Maths, University of Northumbria,
Newcastle-upon-Tyne, UK. NE12 7AE
Email: {c.rook,d.a.harrison}@unn.ac.uk

1 Introduction

Ant algorithms were first used for the travelling salesman problem [1], [2]. They have since been applied to a number of other problems including the graph colouring problem [3], the quadratic assignment problem [4] and vehicle routing [5]. In all these situations, the algorithm is used to develop a single solution in a problem where there are no changing parameters. This paper discusses the use of ant algorithms to develop a path for an agent when the agent must take into account the actions of other, similar, agents. In some situations, such as in a computer game or a competitive financial market, these agents will be working against each other. In other situations, such as crowd movement or air traffic control, the agents may be working co-operatively for the benefit of all.

This paper will first consider the antagonistic situation, as applied to a simple maze-based computer game. A co-operative situation is then examined, using a number of agents attempting to perform a co-operative task in a maze. For both of these situations, an ant algorithm is used to make decisions for each agent. It will be demonstrated that ant algorithms have some potential for these situations. The advantages of using ant algorithms for these purposes will be considered and the results of experiments exploring the strengths of the algorithm will be given. Some problems with the algorithm used will be discussed, as well as a number of improvements that could be made.

2 Cooperative/Antagonistic situations

For this paper, problems involved in finding paths through mazes will be considered. It should be remembered that processes applied to pathfinding in a maze can also be applied to any decision-making problem that can be expressed as a transition through a number of connected nodes. For all the maze problems considered, there will be a number of agents moving through the maze, attempting to achieve some goal. Each agent will have to take into account the anticipated decisions of all the others. There are four different situations in which these agents may have to consider each other.

- Antagonistic - the agent's main goal is to act to the detriment of other agents. This has obvious application to computer games, in which computer-controlled agents attempt to anticipate the actions of the human player in order to prevent the human player from winning.

- Competitive - the agent's prime aim is to achieve some goal either more quickly or more effectively than the other agents. In this situation, each agent will be aiming at the main goal, but will also have to consider the actions of other agents in order to impede them or to prevent being impeded by them. This may have some applications in financial market, where companies make financial decisions in order to secure the strongest possible control of the market, but may need to act to prevent rival companies from achieving the same goal.
- Co-operative - the agents share the same goal and wish to work together to achieve the goal as quickly or as efficiently as possible. They may need to take into account the actions of the other agents to avoid duplicating work or impeding each other. This may have application to the use of autonomous machines that are working together to perform a task, or to the efficient movement of crowds.
- Avoidance - the agents are pursuing their own goals and are not interested in the success or failure of the other agents. They may need to take into account the anticipated movement of other agents in order to avoid being impeded by them. This may have application to the movement of vehicles through traffic.

For the work discussed in this paper it has been assumed that the agents do not communicate. This means that each agent has to anticipate the actions of the other agents and cannot rely on this information being provided.

3 An antagonistic ant algorithm applied to a simple game

To produce an artificial intelligence that can deal with an antagonistic situations experiments have been conducted with an ant algorithm which will decide on the moves for each player in a maze game. Ant algorithms have already been used in path-finding situations [1] [5]. The innovation here is that a number of paths (one for each agent) will be developed simultaneously, and the paths developed by each agent will affect the suitability of the paths chosen for other agents.

Each computer-controlled agent will start to decide on a move by creating a copy of the current state of the game. Each agent in the game, including the agent currently deciding its move, will be represented by an ant on the copy of the board. Ants can mark the copy of the board with pheromone. Each ant has a distinct type of pheromone and is only guided by its own pheromone. The ants will play a game, making moves at random from node to node along legal edges. These moves can be represented as a single path for each ant on the copy of the board. After all the ants have made a set number of moves the algorithm will assign a score to each ant, based on how good its chosen moves were. This score will be used to adjust the pheromone on the edges along the path used by that ant. If the score was high because the ant's moves were good the pheromone will be increased. If the ant made poor moves the pheromone will be decreased. The ants will then be returned to their starting positions, to make another run through the game. This time, the ants' choices will be guided by their pheromone using a biased random system. A number of iterations will be performed, at the end of which the agent will choose to follow the most strongly marked edge from the starting position of the ant that represents itself.

This entire process must be repeated by each agent in the game. As the agents do not communicate they must each run through this process in which they represent both themselves and all other agents as ants in an imaginary game. It will be quite possible, then, for agents to make incorrect predictions about the actions of opposing agents. However, since both its own path and the path of the imaginary opponents has been optimised by the use of an ant algorithm, it is likely that the agent will have based its predicted route on good routes for the opposing agents.

This has been applied to a game in which up to eight players move around a maze attempting to destroy each other by picking up "power cells" that allow them to shoot or crash into other players. This game

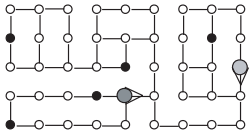

Figure 1: Diagram of the game maze. Circles are nodes. Black circles are nodes with power-ups. The larger, grey circles are agents. The direction they are pointing in is indicated.

has been chosen in preference for games traditionally used for studies in artificial intelligence studies for a number of reasons. Firstly, the game involves finding paths through a maze, and ant algorithms are most suitable for problems which can be easily treated as searching for an optimal path [6]. While many traditional games can be treated as a search through some sort of graph [7], it seemed easiest to apply the system to a game in which the path metaphor was more obvious. Secondly, the game allows for a number of players, rather than just two. The system suggested works for a number of players who all need to anticipate each others' actions. Finally, the game has some resemblance to a number of commercial computer games currently on the market.

In brief, the game board is a maze constructed of a grid of nodes connected by edges to adjacent nodes. Each node can have two, three or four edges, heading north, south, east or west. Some nodes are marked to indicate that they contain power cells. Players are initially placed on starting nodes and can move along edges, but cannot return along an edge used in the last move. Players also cannot remain stationary. If a player has reached a node with a power cell, the power cell is removed and the player temporarily has the ability to do damage to other players (for a set number of moves). Damage can be done by shooting, which requires line-of-sight along existing edges, or by colliding with other players. Once a player has received too much damage his piece is removed from play. A small sample board is shown in $figure1$.

In order to decide on the agents' moves, each agent uses its own execution of the ant algorithm suggested. For each agent, a copy of the game board is made, and all pieces on the copy are replaced with pieces controlled by ants. Each ant has its own copy of the edges in the maze, which it uses to mark with trail. The trail on all edges is set to some starting value (5 was used here.) All pieces then make biased random moves through the game influenced by trail for a set number of moves. For this game, the number of moves made is identical to the width of the board - usually 12. Note that this is the number of moves made by each ant, so for a game with four ants this would represent a 48-ply search. This may seem like a long way to look ahead, but in this game the number of choices at each move is actually quite small. The probability of moving to each node is given by:

$$P_{kgh} = \begin{cases} \frac{\tau_{kgh}}{\sum_{f \in LegalNodes} \tau_{kgf}} & if h \in LegalNodes \\ 0 & otherwise \end{cases} \quad (1)$$

where P_{kgh} is the probability of the $k-th$ ant moving from the current node g to the node h, τ_{kgh} is the trail left by ant k between nodes g and h, and $LegalNodes$ are nodes that are connected to g by an edge, and that were not occupied by the $k-th$ ant on the previous turn.

Since the pheromone on all edges is initially the same, the first iteration will produce purely random paths as there is no heuristic applied here. During the iteration ants will cause and receive damage automatically. A score will be calculated for each ant based on these events. Events early on in the iteration have more effect on the score than events later on, because earlier events are more likely to be accurately predicted. The score ranges from -1 (immediate defeat - destruction of self) to +1 (immediate victory - destruction of all opponents).

The formula for this score is:

$$S_k = \sum_{t=0}^{MaxTurns} \Big(\frac{\Delta D_{at}}{D_{at}} - \frac{\Delta D_{kt}}{D_{kt}}\Big).\Big(\frac{MaxTurns}{t + MaxTurns}\Big) \quad (2)$$

where t is the number of the turn, $MaxTurns$ is the number of turns to be run each iteration, S_k is the score for ant k, ΔD_{at} is the total damage taken by all ants during turn t, D_{at}, is the total damage all ants can receive before being removed at the start of turn t, ΔD_{kt} is the damage taken by ant k in turn t and D_{kt} is the damage ant k can receive before being removed at the start of turn t.

The score for each ant is used to update that ant's pheromone trail. A negative score will cause the trail on the edges used by that ant to decay down towards zero. A positive score will make the trail tend towards an asymptote at ten. (Any number could have been used.) The magnitude of the score controls how much the trail moves towards ten or zero. This update is given by:

$$\tau_{kij} = \begin{cases} (l - s_k).\tau_{kij} + s_k.(5 + 5\frac{S_k}{|S_k|}) & \text{if edge } (i,j) \text{ is on } k^{th} \text{ ant's tour} \\ t_{kij} & \text{otherwise} \end{cases} \quad (3)$$

where τ_{kij} is the trail left by ant k on edge (i, j), and s_k is the score for the $k - th$ ant's tour.

The ants are then reset to their starting conditions, and the process is repeated. A number of iterations are performed. Experiments with up to 5000 iterations have been carried out. 250 iterations seems to be a good number, balancing decision time with quality of paths chosen. When the iterations are complete the algorithm chooses the edge with the strongest trail as the best edge to take.

It is worth noting that the ant algorithm makes no use of any game-specific heuristic. The ants are initially guided by random choice, then start to follow the trail marked by previous ants. As has been demonstrated in [2], the use of a heuristic function greatly improves the Ant Colony System. As will be discussed below, one of the strengths of this system for computer games is its flexibility. Since the method works only by playing imaginary versions of the game, it automatically adapts when the nature of the game changes. For many modern computer games and real situations, the strategies required can change drastically in the middle of play. The use of a heuristic function would limit this flexibility unless it were, itself, able to adapt to large changes in the strategies required.

3.1 Advantages and disadvantages

Observing the game indicates that the system provides a reasonable artificial intelligence for the computer-controlled players. Given a thinking time of around a second the game is capable of beating a human player thinking for the same length of time. For very short thinking times, the algorithm can make sound decisions far faster than a human. Experiments described in the paper demonstrate that the algorithm easily defeats random movement, and performs better than a game-specific algorithm for this simple game.

The algorithm can adapt easily to the time available for thinking by adjusting the number of iterations it performs. If the processor is busy with other tasks the algorithm can use small numbers of iterations to adapt. The number of iterations can be set in advance, or the algorithm could make iterations until interrupted.

The algorithm is fairly easy to write. Since it has a natural metaphor, this makes it easy to explain. Also, most of the functions needed for the artificial intelligence must already exist in the rest of the program. This is because the approach works by running imaginary games.

The algorithm is general and can be applied to any situation where the decisions can be represented by a maze. The system does not need to be provided with any understanding of tactics as it finds the best path by running an imaginary game.

The algorithm is extremely flexible. Any change in the rules of the game is automatically handled. In early versions of the game, damage could only be done by achieving line-of-sight on the opponent. When the game was modified to allow damage by collision, no alteration to the artificial intelligence was needed, as the modification to the game also modified the imaginary games used by the algorithm. Similar modifications tested include addition of nodes that grant temporary invulnerability, or the ability to drop mines. For many complex games or real situations, changes such as these could happen in the middle of a game, and the artificial intelligence would need to adapt. Also, many game users like to customise games by adding modifications, but do not have the ability to adjust the artificial intelligence. This algorithm would automatically adjust to any such changes.

The algorithm may make decisions in a similar way to a human player confronted with an unfamiliar game. It thinks through a series of moves to see what happens. If the results of that trial are not favourable, it tries a different path. If the results are favourable, it tries the path again to see what happens if the opponents do something different. This process is repeated until the algorithm runs out of time.

The most obvious disadvantage is the slow speed. Experiments in which agents are allowed different number of iterations as they play against each other indicate that, for this game, little improvement is to be gained after 200 iterations. For a game with four agents, each agent is capable of deciding on a move in around 100ms. Since the game has been written in Java, and little attempt has been made to optimise the code, it seems likely that considerable improvements can be made.

The algorithm has problems when an ant returns to a node it has previously used in the same iteration. The ant will mark an edge with pheromone on each occasion. This pheromone cannot indicate which occasion the pheromone refers to.

When none of the options available to the computer-controlled player are good, the algorithm has difficulty deciding which of the undesirable options is the least undesirable. This is because if there are several undesirable moves the most undesirable will lose the most pheromone each time it is explored, but will be explored less often. This means that undesirable edges, on some occasions, may all lose approximately the same amount of pheromone. Inspection of the game tested reveals occasions when this seems to be happening, as players occasionally make poor choices when in a bad situation.

The system assumes that opposing players will follow the good path found for them.This means that the computer can perform badly when opposing players have a choice of good paths to take.

3.2 Potential improvements

There are a number of improvements that could be made, depending on the situation:

At the moment, the system does not include any heuristic other than guidance by pheromone. On the first iteration ants move randomly and on subsequent iterations they are guided only by pheromone. A game-specific heuristic to guide the ants in their initial choice of path should improve performance. As discussed above, this would reduce the flexibility of the system.

All the computer-controlled agents make use of their own independent execution of the algorithm. In some situations, the agents could all share the results of a single execution of algorithm. This may be cheating slightly, since computer-controlled agents will always accurately predict the actions of other computer-controlled agents, but it will greatly reduce the time needed for all computer-controlled agents to make a decision. For the agents to co-operate in this way would provide a degree of teamwork between the computer-controlled agents working against a human player.

For each move the agent makes, the algorithm is repeated with the pheromone set to the starting level. There may be some benefit from using the pheromone left over from the previous move, perhaps with some decay. This would mean that the algorithm would not be starting from scratch, but would be

guided by the paths previously developed. It seems likely that this could improve the quality of the algorithm. In the extreme case, if a computer had no time to develop a new path, the agent could simply follow the path previously developed.

4 A cooperative ant algorithm applied to a maze problem

Some progress has been made in testing ant algorithms in a co-operative situation. For these experiments, the game used above was modified, so that the aim of the game was for the agents to collect as many power cells as possible between them in a set amount of time. All shooting elements of the game were removed and the agents operated purely co-operatively - it did not matter which agent picked up the power cells, as long as someone did.

This study is still in the early stages and there are, as yet, no statistical comparisons between the ant algorithm, random movement and a greedy heuristic. Inspection of the actions of agents, however, shows that they distribute their efforts effectively - the agents will split up and attempt to collect power cells from different parts of the board.

5 Conclusions

It has been demonstrated that ant algorithms can be applied to game-playing. The system suggested in this paper produced reasonable results and it may be that ant algorithms could be useful for action games in which the AI does not need to be perfect. No attempt has yet been made to compare ant algorithms with other game AI methods, or to tune the method. The system does have some useful qualities, as it is flexible, versatile and easy to use. There are also a number of problems with that may need to be dealt with before ant algorithms can be used in a more complex game. It has also been demonstrated that ant algorithms can be used to allow agents to co-operate in a path-finding situation without communicating.

References

[1] Dorigo, M. Maniezzo, V. Colorni, A. *Ant System: An autocatalytic optimizing process.* Technical Report No 91-016 Revised, Politecnico di Milano, Italy. 1991.

[2] Dorigo, M. Maniezzo, V. Colorni, A. *The Ant System: Optimization by a colony of cooperating agents.* IEEE Transactions on Systems, Man and Cybernetics 26(1), pp29-41. 1996.

[3] Comellas, F. Ozón, J. *An ant algorithm for the graph colouring problem.* ANTS'98 - From Ant Colonies to Artificial Ants: First international workshop on ant colony optimization, Brussels. 1998.

[4] Roux, O. Fonlupt, C. Talbi, E. Robillard, D. *Co-operative improvement for a combinatorial optimization algorithm.* Evolution Artificielle99, Dunkirk, France. 1999.

[5] Bullnheimer, B. Hartl, R. F. Strauss, C. *An improved ant system algorithm for the vehicle routing problem.* Sixth Viennese workshop on optimal control, dynamic games, nonlinear dynamics and adaptive systems, Vienna, Austria. 1997.

[6] Colorni, A. Dorigo, M. Maffioli, F, Maniezzo, V. Righini, G. Trubian, M. *Heuristics from nature for hard combinatorial problems.* International Transactions in Operational Research, 3(1), pp1-21. 1996.

[7] Winston, P. H. *Artificial Intelligence. 3rd Edition.* Addison-Wesley. 1992.