Comparison of an Evolutionary Algorithm and Ant Colony Optimization as applied to a Path-finding Problem

Denton Cockburn
Victor Parmar
School of Computer Science, University of Windsor
cockbu3@uwindsor.ca, vic@socr.uwindsor.ca

Abstract

There are currently several Nature-Inspired Computing (NIC) methods. These include Genetic Algorithms (Holland, 1975), Particle Swarm Algorithms (Kennedy and Eberhart, 1995) and Ant Colony Optimization (Dorigo et al, 1991). These methods are able to provide very good solutions when applied to some very difficult problems. In this paper, we are looking in the domain of Path-finding. These algorithms have varying degrees of success in this area. What they have in common is an a priori knowledge of the search space, as well as a heuristic to guide their search towards goal states. These algorithms work well in situations where these requirements can be met, but are not applicable when they cannot. We seek to provide an evolutionary algorithm (HMA) that can provide good results without having a priori knowledge of the search space, as well as not having an initial heuristic with which to drive the algorithm's search.

We hereby declare this report is completed by our team independently. All contents referred to other people's work are cited explicitly.

1. Seminar Report

1.1. Application Domain

The background of the proposed project application is an implementation of an evolutionary algorithm based upon human migratory behaviour. The goal is to analyze the behaviour, in terms of performance, and efficiency, of said algorithm with an already established algorithm, Ant Colony Optimization.

A clear motivation is the development of an algorithm that may be more efficient in using certain resources than already established algorithms being applied to similar problem spaces. Also, another goal is the exploration of the possibility of the proposed algorithm being more effective when applied to certain problems.

1.2. Application Scenario

The comparison of the two algorithms is to be analyzed based on their performance on solving path-finding problems. A grid will be created with random costs, with the objective being to discover as short a path as possible, in a reasonable time.

Ideally, the evolutionary algorithm will prove to have better performance in terms of memory usage and solution on the given problem specification and the solution to a given problem specification would be the cheapest path found to the goal state.

1.3. Expected Demonstration

For the project, both the evolutionary algorithm and the Ant Colony Optimization algorithm shall be implemented and compared.

The proposed evolutionary algorithm is currently modeled on human migratory group behavior with male and female agents that possess information concerning the search space and visited nodes. These agents also have a lifespan that determines the number of times they can mate, with their children possessing information from both parents, thereby implementing continuity. There is an inclination on the part of the agents to stay within sight of someone else in their group to encourage clan behaviour, unless they are going to mate or feel compelled to explore unknown terrain. Thus, the current agent exploratory priorities are:-

i. First priority is to move towards the goal if known,
ii. Second priority is to move toward an unvisited node within sight of others,
iii. Third priority is to move to some unvisited node
iv. Fourth priority is mating (males seek females and vice versa),
v. Fifth priority is to move towards the least populated node within sight of others,
vi. Sixth priority is to move towards a randomly selected node.

1.4. Seminar Conclusions

The Human Migratory Algorithm (HMA) attempts to mix a number of different approaches to multi-agent systems: the use of gender-based heterogeneous agents, as well as age-based systems. The major shortcoming we noticed in HMA is its data representation. The representation used in the algorithm required interacting agents to ascertain the amount of knowledge the other agent has to determine knowledge differences. The problem with this is that as the knowledge base of agents grow, it becomes infeasible to compare knowledge efficiently. The algorithm also requires a lot of input parameters, even though this seems to be a general problem with these forms of algorithms. In HMA, max group sizes, age of adulthood, knowledge exchange ratio, among others. HMA, ACO and other evolutionary algorithms in general, do not seem to be very appropriate for path finding applications with many states and operators.

ACO, while promising in some areas such as network routing, does not seem to be very good in path findings applications such as game navigation. Another shortcoming is also in the fact that while ACO is dynamic, it has no incremental implementation, which would be quite useful in the highly dynamic game space. The fact that our ants would oftentimes find themselves trapped also highlight another problem with ACO, that being it's random exploration nature. This in itself is not an issue, except that ants are not allowed to navigate nodes they have previously visited, thus when an ant moves into a corner, say via a path with only one way in, then it is stuck, and dies. In such an example, if pheromones were deposited in an on-line fashion as nodes were traversed, it would also increase the likelihood of another ant also taking this bad path. In turn, this displays the path reinforcement behaviour of ACO, which, while good when a good path is found, can lead to agents converging upon weak paths. It is also hard to ascertain how many agents are appropriate in the case of random maps. Depending on the size of the map, a lot or only a few agents and generations may be needed. One thing we discovered in our implementation, is that ACO can have a drastically significant improvement in the path found with the addition of a few generations, so that confirms the difficulty in choosing an appropriate amount of generations.

A question that would be interesting to explore is how would HMA and ACO behave in a dynamic environment? When nodes are continuously changing, ACO would not have enough time to enforce a good path, while HMA agents knowledge would be quickly obsolete, even though its broadcasting features would be able to alleviate this somewhat. Let's say an agent senses a state has changed compared to its knowledge base, it can update itself and broadcast this new information to others within its group. The dynamic nature would mean that queries would be proposed to agents whose knowledge is most likely obsolete.

2. Project Report

2.1. Background

There are currently several Nature-Inspired Computing (NIC) methods. These include Genetic Algorithms (Holland, 1975), Particle Swarm Algorithms (Kennedy and Eberhart, 1995) and Ant Colony Optimization (Dorigo et al, 1991). These methods are able to provide very good solutions when applied to some very difficult problems. In this paper, we are looking in the domain of Path-finding. These algorithms have varying degrees of success in this area. What they have in common is an a priori knowledge of the search space, as well as a heuristic to guide their search towards goal states. These algorithms work well in situations where these requirements can be met, but are not applicable when they cannot. We seek to provide an evolutionary algorithm (HMA) that can provide good results without having a priori knowledge of the search space, as well as not having an initial heuristic with which to drive the algorithm's search.

2.2. ACO Algorithm

Ant Colony Optimization (ACO) algorithms were first proposed by Dorigo et al. in 1991. It was intended as a multi-agent approach to solving combinatorial optimization problems such as Traveling Salesman and the Quadratic Assignment Problem. ACO is based on the behaviour of real ant colonies, with modifications as regards to the problem domain being applied to. ACO works by assigning ants to find a goal, using only their knowledge of past locations they have visited, and information regarding the current and adjacent locations.

While walking from the starting location to the food source and back, ants deposit a substance called pheromone. Other ants are able to detect this substance in the environment, and thus this can be used to drive such ants towards the food source as well. The presence of pheromone increases the likelihood that another ant will choose the same path. When an ant finds the goal, the amount of pheromone on the trail increases; again increasing the likelihood of the path being followed by other ants. Collectively, these ants are very adept at finding an efficient path. Pheromone evaporates over time, so if a path is not reinforced by other ants, it fades away in the system. This allows ACO to be very adept at handling dynamic environments.

ACO has been applied to many problems such as the Traveling Salesman, Quadratic Assignment, and Shortest common supersequence, with good results. In this paper, ACO is applied in a path-finding context to locate the shortest path between a random starting location and a random goal location on a grid. The environmental information available to the ants is in the form of the cost of moving from one location to an adjacent location. The pheromone deposition approach used is that from Ant System (Dorigo, 1991). ACO requires that ants do not go to locations to which they have already visited, a requirement which presented a problem in our application. The ants would at times navigate themselves to locations to which no unvisited locations existed, such as in a corner of the grid. To help drive the ants towards the goal, a global heuristic was also given. Below is the ACO algorithm as used here.

2.3. HMA Algorithm

The premise of the algorithm begins with the existence of a group of wandering humans. These humans are unaware of where they are trying to reach, yet have the understanding to know when they have arrived. They have a natural inclination towards exploration, and a rudimentary communication system in the form of shouting to each other. They also have social tendencies, in that they will prefer to stay within groups, but are willing to break off from those groups to explore new territory.

In the search for the goal, these agents can mate, but not with ancestors or other individuals that have mated with an ancestor. Agents also do not mate with the same individual twice, because they have limited reproductive capacity and their goal is to create strong offspring. Male agents can get aggressive, in which state they may fight other male agents for dominance. New children need to reach an acceptable age before they can mate or fight. When an agent discovers new territory, it broadcasts this information to others within its group, with a group being defined as a set of agents within earshot of some other individual in the set.

On each time step, an individual has to decide where to go to next. The tendency is to find a location that it does not know about that is also within safe distance of another in the group. Failing this, the individual will still go to this unknown location. If no such location exists, the agent will look to adjacent locations for a possible mate if they are of mating age, if that also fails, it will simply move to a location with the group or if it is alone, then just a random location.

When two agents meet at a location they interact. If two female agents meet, they will simply exchange some knowledge. The interaction of two adult males depends upon relationship and aggression level. If there is no relation between the two males, and they are both in an aggressive state, they will fight. The result of the fight is dependent upon the knowledge of the agents, as well as an element of chance. The algorithmic result of the fight is that one agent's lifespan is shortened, and its knowledge is dominated upon by the winning agent. The blows in the battle are the removal of nodes which are not currently in the best path from the start to the agent's current location. When two adults of the opposite genders interact, if they are valid mates, they reproduce a child with the knowledge of both the parents. The gender of the child is random. After a random amount of allowed mating counts, an agent dies.

When an agent discovers the goal, it broadcasts that information to others in its group, and then “ascends”. Upon ascension, these individuals are no longer within the individuals in the populace. The other individuals, who now know the location of the goal, will move towards such goal. When enough individuals have ascended, these ascended individuals are asked the question posed to the algorithm, which in this case is the least expensive path to the goal. The minimal answer is the one the algorithm returns.

2.4. Algorithm Implementation

It was decided that Lisp would be the language in which we did our implementation and testing. Lisp made a prime candidate due to its rapid developing features inherent in dynamic languages. To develop our system, Linux was the main development platform, with Windows being secondary. Emacs with the addition of SLIME with SBCL and Lispworks were the development environments on Linux and Windows respectively.

The system is divided into two Lisp packages, HMA, and ANTS. Within the HMA package is all the environment setup code, such as the specification of the Graph object.

All the classes within the system and their brief descriptions are:

HMA::Graph
The grid containing the nodes to be traversed; the heuristics for each node are setup here and determined by the starting and ending locations specified by the testing environment. Each node is a neighbour of any adjacent node in the grid, not including diagonals.

HMA::Astar
Implementation of the A* algorithm.

HMA::Node
Representation of a node to be traversed and features terrain cost and heuristic. The terrain cost of node v is equal to the edge cost of (u, v) and vice versa for all u

that neighbour v.

HMA::Village
A specialized Node object that contains information about initial villagers, such as amount and who they are, also responsible for the creation of said Villager objects.

HMA::Villager
An HMA individual

HMA::Male
A male villager

HMA::Female
A female villager

ANTS::Ant
An ACO ant agent

Figure 1. The ACO algorithm in pseudo-code.

```
1. Initialize the world (grid)

2. procedure dispatch-ants()
    1. loop GENERATION times
    2. create N ants
        1. set the start location for each ant
    3. move-ants()
    4. output the cost of the best path

2. procedure move-ants()
    1. loop for each ant until goal reached or no more moves possible
        1. if goal reached
            1. deposit pheromone on path
            2. update best path cost if necessary
        2. else move-ant()

3. procedure move-ant()
    1. pick out the unvisited neighbours
    2. if there is an unvisited neighbour
        1. calculate the probability of going to each neighbour
        2. generate random number from 0 to 1.0
        3. move to new location
    3. else do nothing
```

Figure 2. The HMA algorithm in pseudo-code.

```
1. create villagers

2. procedure HMA_search()
    1. loop until desired amount reaches goal
        1. loop for each villager
            1. move-villager()
            2. if more than one villager at location, interact(villager other-random-villager)
            3. broadcast location to villagers in group

3. procedure move-villager()
    1. n = pick-best-destination(villager)
    2. set villager location to n
    3. add n to villager history
    4. increase villager age
    5. if villager location is goal
        1. remove villager from list of villagers
        2. add villager to ascended villagers

4. procedure pick-best-destination (villager)
    1. N = neighbours of location of villager
    2. if goal is known
        1. return n from N where distance from n to goal is minimal
    3. if unvisited spots US from N within range of group
        1. pick Q from US where villagers at Q is minimal
        2. return n from Q where edge-cost is minimal
    4. else unvisited spots U from N not in range of group
        1. pick Q from U where villagers at Q is minimal
        2. return n from Q where edge-cost is minimal
    5. else spots M from N where there is a potential mate for villager
        1. pick Q from M where villagers at Q is minimal
        2. return n from Q where edge-cost is minimal
    6. else spots S from N within range of group
        1. return random element in S
    7. else return random location in N
```

2.5. Algorithm testing and comparison

A Graph object is created. A random start and goal location is chosen. A Village object is then created with its location being the start location. The Village object creates 30 Villager objects, which are randomly male or female. These agents then operate and perform a maximum of 500 movements/decisions. The best path from any agent that has reached the goal is returned as the solution of the HMA algorithm. On the same Graph object, ACO is then applied. ACO loops for 10 generations, creating 100 Ant objects on each. These Ant objects are dispatched through the Graph, with the best path being its solution.

We used the A* algorithm as the benchmark for comparing the results of our experiments. A* is optimal and complete, so we would be guaranteed that the path found is best. There was no user interface created for our project, simply because we were interested in comparing two numeric algorithms. Even though these two algorithms could be displayed graphically, there would be nothing gained for the amount of work such an undertaking would have required.

For our testing methods, we decided to first aid the ACO algorithm by allowing it access to a global heuristic. All agents were run on a 100x100 grid, thus having 10,000 nodes. Random starting and ending locations were selected, for which both algorithms had to adhere to. We also decided upon running the ACO with 100 agents per generation, this was decided after running samples with other numbers, with varying results. 10 generations seemed appropriate for ACO as we reasoned that ACO would be converging upon a final solution at that point. Using the same principles, 30 agents was found to be good for HMA with a maximum group size of 10, as using more of either parameter would seem to lead to the population growing too fast, and actually resulting in the algorithm being slower.

We had to balance the gender of the HMA agents with a random probability of 50%. If we allowed too many males, then they would kill each other, decimating the population. On the other hand, too few men would result in a low reproductive rate, slowing knowledge acquisition.

We limited the amount of iterations the HMA agents would navigate for, this was done to limit the amount of time the algorithm could potentially spend if not enough agents have reached the goal, with enough being 3 as we decided in our testing. It was possible for us to accept when there was only 1 successful agent, which is what we did if the agents exceeded the 500 iterations limitations that we imposed. The tests were the comparison of 1000 cases where ACO and HMA both found solutions.

2.6. Results and conclusions

From the comparison we did between A*, HMA and ACO, it was found that the path found by ACO was 41% more expensive than the optimal path on average, which HMA was only 23% more expensive.

Additional tests which could have been performed if more time were available include comparing ACO and HMA on the basis of how many nodes are actually explored/expanded. Intuition would suggest that HMA would expand less, as agents don't expand nodes that someone else in their group has explored, unless there are sufficiently inviting conditions. ACO on the other hand, by its very nature, re-traverses nodes previously visited by others.

The algorithms could also be compared on how much processing time is used, however this would be highly implementation dependent and thus probably not very useful scientifically. Comparing computer memory usage would also suffer from the same limitation.

3. References

M. Dorigo, G. Di Caro, and L. M. Gambardella, “Ant algorithms for discrete optimization,” *Artificial. Life*, vol. 5, no. 2, pp. 137–172, 1999.

M. Dorigo and G. Di Caro, “The ant colony optimization meta-heuristic,” in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. New York: McGraw-Hill, 1999, pp. 11–32.