Path Optimization for Multiple Objectives in Directed Graphs using Genetic Algorithms

Juan Rada, Rubén Parma and Wilmer Pereira

***Abstract*— This paper presents a genetic algorithmic approach for finding efficient paths in directed graphs when optimizing multiple objectives. Its aim is to provide solutions for the game of Animat where an agent must evolve paths to achieve the greatest amount of bombs in the fewest moves as possible. The nature of this problem suggests agents with memory abilities to choose different edges from a vertex v such that each time v is reached, the agent can avoid cycles and be encouraged to keep searching for bombs all over the directed graph. This approach was tested on several random scenarios and also on specially designed ones with very encouraging results. The multi-objective genetic algorithm chosen to evolve paths was SPEA2 using one-point crossover and low mutation to allow genetic diversity of the population and an enhanced convergence rate. Results are compared with an implementation for the same game using Ant Colony Optimization.**

I. Introduction

ACCORDING to [1], the most critical task for developing a genetic algorithm is how to encode a path in a graph into a chromosome. Two approaches were taken to represent paths into nodes in [1]. The first method is a previous-node-based encoding which uses the allele (the value a gene takes) to store the previous node id and its locus (position of a gene in a chromosome) to indicate its node id. The second approach was based on priorities but it was focus on a graph itself and not a digraph.

Ideas were taken from the previous-node-based encoding but the whole structure showed flaws applying it to this problem. So a novel approach had to be taken and it is the one described in this article.

Ant Colony Optimization, on the other hand, is a highly effective algorithm for finding paths in graphs and digraphs. This approach is frequently used, achieving results that may position it as an ideal problem solver when dealing with this type of situations. It could only be natural to compare the approach of this article against the Ant Colony Optimization algorithm described in [2].

II. The Game of Animat

Animat is a game about an agent trying to find all bombs in the fewest moves as possible. It takes place in a grid of `MxN` squares containing bombs, obstacles or the agent itself. Each square determines the possible directions an agent may take. Figure 1 shows an example of how a grid of `7x7` looks like.

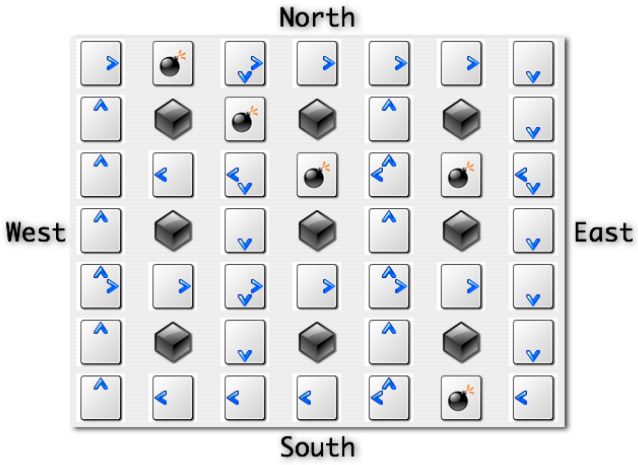

Fig. 1. Animat grid$(7, 7)$

The criteria for building animat grids is based on for each (row,column) in the grid:

```
if row mod 2 == 0 ∧ column mod 2 == 0
   then  grid(row, column).content = BLOCK
if row mod 4 == 0
   then  grid(row, column).direction = NORTH
   else  grid(row, column).direction = SOUTH
if column mod 4 == 0
   then  grid(row, column).direction = EAST
   else  grid(row, column).direction = WEST
```

III. Genotype and Phenotype

The genotype is the genetic constitution of an individual. It produces a set of observable characteristics (phenotype) when it interacts with the environment. In this case, the environment is the Animat grid and the phenotype is the resulting path from the interaction between the environment and the agent's genotype.

In the game of Animat, the genotype of an agent contains decisions to be taken at bidirectional squares only. Other squares are not relevant since they offer one or no decisions at all. Initially, bidirectional squares were numbered and indexed into the genotype but this led to phenotypes with numerous cycles (suggesting lack of memory), misuse of genotype useful information and evolution hardly ever

Juan Rada (jcrada@gmail.com) is with the Department of Electronic Engineering at Universidad Fermín Toro, Cabudare - Venezuela.

Rubén Parma (parmaia@gmail.com) is with the Department of Systems Engineering at Universidad Centroccidental Lisandro Alvarado, Barquisimeto - Venezuela.

Wilmer Pereira (wpereira@ucab.edu.ve) is with the School of Informatics Engineering at Universidad Católica Andrés Bello, Caracas - Venezuela.

reached convergence. This approach was based on the idea behind the previous-node-based encoding in [1], but due to its results in this problem a different approach had to be taken.

This time the genotype would contain sequential decisions. The first gene determines the direction to be taken at the initial square to move to the next one. No matter which square is next, the second gene determines the next direction to be taken. And so on.

Up to this point the phenotype would lead to a valid path through the digraph. Cycles would become less frequent since the agent may take different directions each time a previous visited vertex is reached. Memory issues are solved and evolution will take care of providing the agent with it.

However, the genotype's length is fixed and determined by the amount of bidirectional squares, so it probably might not be long enough to build a path where all bombs could be found. This issue was solved by adding equally-length layers to the genotype and a policy to switch from one layer to another. The number of layers added is given by the amount of bombs in the grid and the policy is to switch to the next layer whenever the agent finds a bomb from any bidirectional square s_i to square s_j or when the end of the layer is reached.

Figure 2 shows the genotype and phenotype for an agent in a grid of 7x7 where the initial square is enclosed in a circle and bidirectional squares are remarked in bold. Each gene contains binary values indicating to move vertically (1) or horizontally (0).

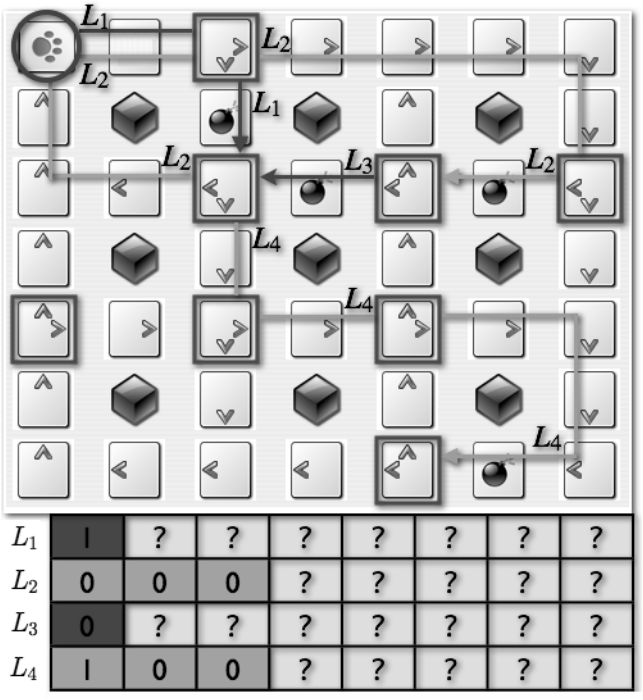

Fig. 2. Agent's genotype and phenotype

IV. SPEA2

The Strength Pareto Evolutionary Algorithm 2 (SPEA2) is a technique for finding or approximating the Pareto-optimal set for multiobjective optimization problems [3]. The aim in this application is to maximize $\mathcal{O}_1$ and minimize $\mathcal{O}_2$. These are the sum of bombs found at each layer (1) and the sum of moves multiplied by the cycles at each layer (2), respectively. The latter is an heuristic to punish paths with cycles.

Crossover is done by selecting a random locus and applying one-point crossover for all layers at the same locus.

$$\mathcal{O}_1 = \sum_{L_1}^{L_n} bombs_{L_i} \quad (1)$$

$$\mathcal{O}_2 = \sum_{L_1}^{L_n} moves_{L_i} \cdot (cycles_{L_i} + 1) \quad (2)$$

Mutation rate is calculated according to [4] where they suggest high mutation rates to obtain a best overall performance when strong elitism is used. According to their experiments, they suggest a normalized mutation rate of 5 when the probability ρ_e to choose a parent individual from the archive instead of the previous offspring population is higher than 0.7 (in SPEA2, $\rho_e = 1$). From Equation 3, mutation rate is calculated.

$$\sigma_{mut} = \sigma \cdot n^{-1} = 5 \cdot (bombs \cdot decisions)^{-1} \quad (3)$$

where n is the length of the genotype.

V. Results

An experiment consists in a 200 epochs evolution of a given population using SPEA2. Results are based on the population emerged from recombination and mutation of the mating pool. This mating pool does not necessarily contain non-dominated individuals as it may be filled by the best dominated individuals in order to satisfy the archive size. The criteria used to fill or truncate the mating pool is explained in [3].

Initially all experiments were made using a mutation rate calculated with Equation (3) but results were not satisfactory, so a conservative rate of 0.001 was used. Evolution was made using parameters in Table I on six specially designed environments in order to test different courses of evolution.

TABLE I
Evolution Parameters

Parameter	Value
Population Size	100
Archive Size	50
Mutation	0.001
Generations	200

Results for each environment are based on the experiment that led to the best population average fitness among 20 experiments performed. The following figures show the environment (left), the average bombs found (upper-right) and

the average moves done (lower-right), both objectives by the whole population during evolution in term of epochs.

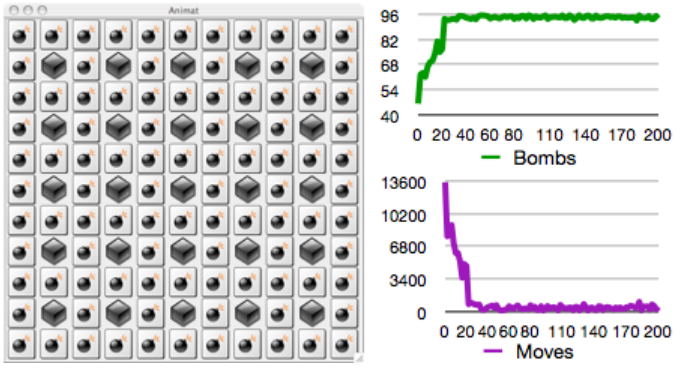

Fig. 3. Environment 1

`Environment 1` (Figure 3) was designed to find the shortest path such that all squares are visited at least once. The best population found an average of 95.93 bombs out of 96 in 218.94 moves.

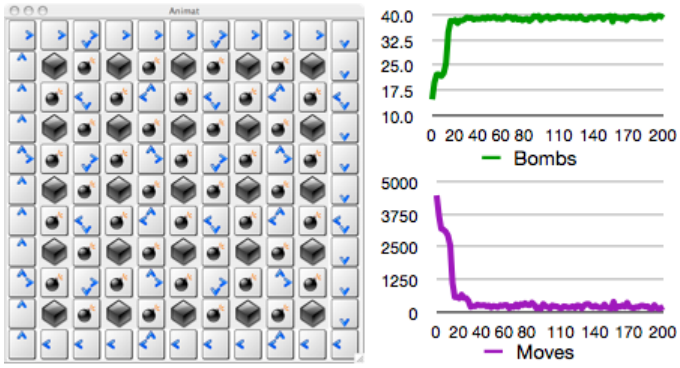

Fig. 4. Environment 2

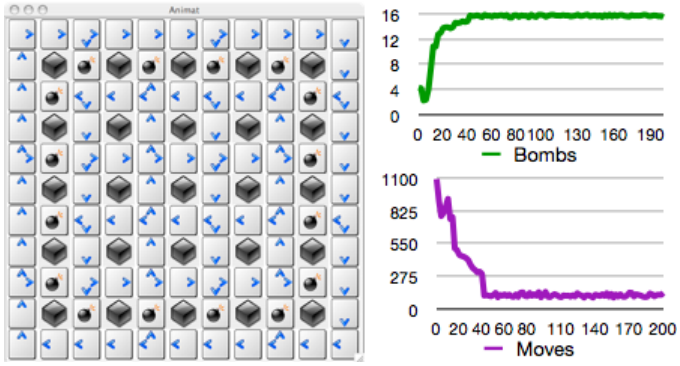

Fig. 5. Environment 3

`Environments 2` and `3` (Figures 4 and 5, respectively) presented some difficulties for finding an efficient path in many experiments due to the distribution of bombs. The bests populations found 39.96 bombs out of 40 in 158.91 moves and 16.00 bombs out of 16 in 103.00 moves, respectively.

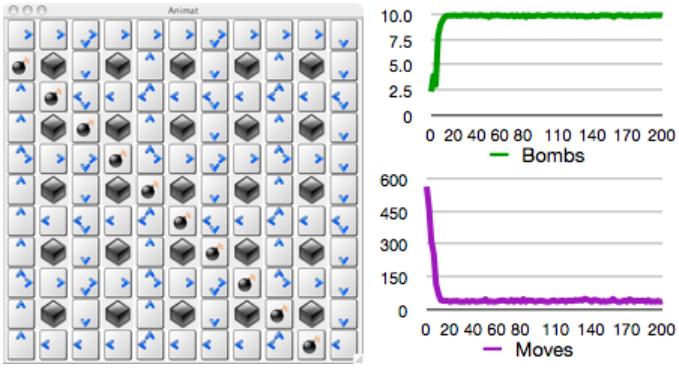

Fig. 6. Environment 4

`Environment 4` (Figure 6) is quite peculiar. In this case, the agent must perform a *ladder* path from top-left to bottom-right and then go back again. The agent performed smoothly and according to the plans. The best population was found at the 20th generation finding an average of 10.00 bombs out of 10 in 39 moves.

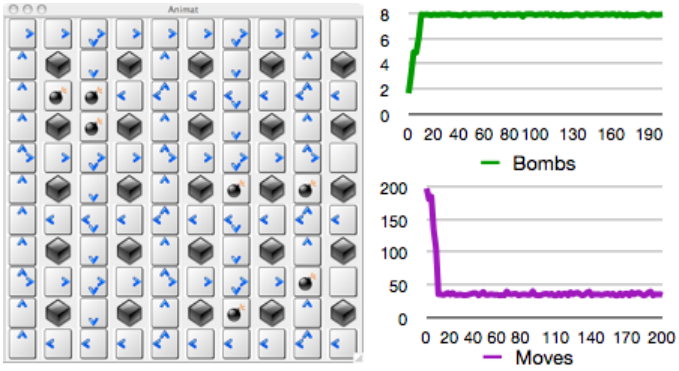

Fig. 7. Environment 5

`Environment 5` (Figure 7) introduced a new variant. Dead ends were added to the grid so whenever an agent reaches one, it gets stuck incrementing cycles and moves for each layer until all layers are executed. Consequently, its genotype has much less preferability than others in its population.

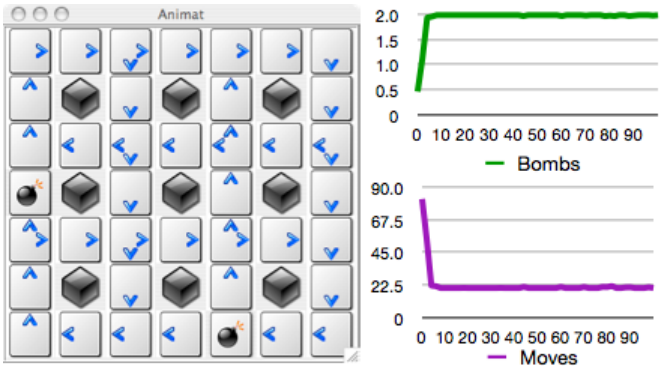

Fig. 8. Environment 6

Finally, `Environment 6` (Figure 8) was designed to prove that the shortest path is not necessarily based on a greedy algorithm finding the nearest bomb at each decision square.

All results are grouped in Table II, where it details the best population evolved in each environment.

TABLE II
BEST POPULATIONS

	Bombs	Max Bombs	Min Moves	Generation
E1	96	95.93	218.94	52
E2	40	39.96	158.91	196
E3	16	16.00	103.00	130
E4	10	10.00	39.00	20
E5	8	8.00	35.00	44
E6	2	2.00	21.00	8

The same experiments were performed using Ant Colony Optimization according to [2]. Table III shows the bests results of each approach (based on the fact that all bombs were found) and Figure 9 shows a graphical comparison.

TABLE III
SPEA2 $vs.$ ACO

		SPEA2	ACO
Environment	# Bombs	# Moves	# Moves
E1	96	218	155
E2	40	158	133
E3	16	103	93
E4	10	39	39
E5	8	35	37
E6	2	21	30

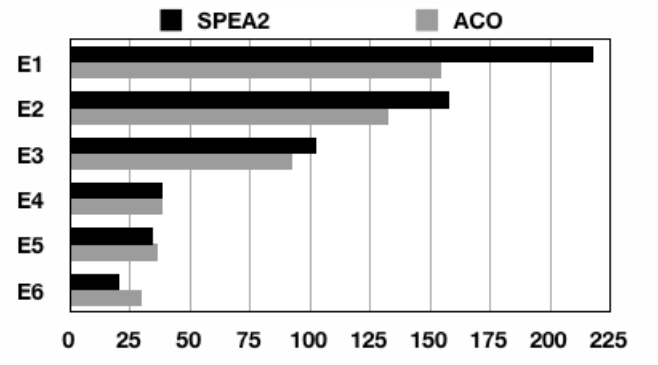

Fig. 9. SPEA2 $vs.$ ACO

VI. CONCLUSIONS

Results were very encouraging using the layered structure of the genotype in the vast majority of experiments performed. These results show the immense power of multi-objective genetic algorithms when using an appropriate structure for encoding a path in a graph into chromosomes.

The layered structure of the genotype is presented as a new approach for finding efficient paths in digraphs where multiple objectives are to be optimized.

The length of each layer L_i is given by the amount of bidirectional squares, this way an agent may visit all of these squares (if needed) to find one bomb. Increasing its length in n genes implies that an agent could visit all squares once and revisit n squares more before finding one bomb. Revisited squares form cycles and these are heavily punished by the heuristics used to evaluate objective $\mathcal{O}_2$. In consequence, increasing its length is not going to improve its fitness despite the bigger search space, instead more generations are needed to find similar solutions because noise is added to evolution rather than relevant genetic information.

The number of layers within the genotype is given by the amount of bombs in the grid. Each layer is intended to be used for finding one bomb and then switch to the next layer. Any number of layers greater than the amount of bombs would have no effect in evolution.

Heuristics used for avoiding cycles worked flawlessly, helping evolution to better classification of individuals in the population. Cycles were successfully avoid in each layer through the course of evolution, resulting in more efficient paths.

Mutation rate as described in [4] was not satisfactory because it was too high causing the average population to reach approximately $n - 3$ bombs in best cases. This is because genes are highly coupled and when one gene is mutated it alters dramatically the whole path.

The Ant Colony Optimization Algorithm described in [2] outperformed SPEA2 in environments where most of the bombs were grouped. However SPEA2 showed better results when the distribution of bombs was less clustered.

Future works rests on improving the structure in order to be able to choose from more than two options at any vertex. We are currently working on extending this genotype's structure for the Open Shortest Path First (OSPF) hop-by-hop routing problem in order to compare it directly against the approach taken in [1]. We are, as well, working on solving the game of Animat via multi-agents.

REFERENCES

[1] N. Selvanathan and W. J. Tee, "A genetic algorithm solution to solve the shortest path problem in ospf and mpls," in *Malaysian Journal of Computer Science*, 2003.

[2] R. Parma, J. Rada, and W. Pereira, "Ant colony optimization applied to an autonomous multiagent game," in *10th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games*, 2007.

[3] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," in *Evolutionary Methods for Design, Optimization and Control with Application to Industrial Problems (EUROGEN 2001)*, 2002.

[4] M. Laumanns, E. Zitzler, and L. Thiele, "On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization," in *EMO '01: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, 2001.