EvolvingAnts

Enrique Areyan

Bio Inspired Computing, School of Informatics and Computing,
Indiana University, United States

eareyan@umail.iu.edu
http://www.enriqueareyan.com/?q=evolvingants

Abstract: this paper introduces EvolvingAnts, an algorithm that combines ant colony optimization (ACO) with a genetic algorithm (GA). In the algorithm, an ant is provided with genetic information that it uses together with *stigmergic* information to construct solutions to a path in a maze. The genetic information of a single ant is represented as a simple-grammar L-System (though other genetic encodings are possible), which is grown a fix number of iterations. Groups of ants are evolved at each step of the algorithm and only those that were able to construct longer paths are selected for reproduction. At each decision step an ant takes into consideration both its individual configuration as represented by its L-System, and the pheromone trail in the environment to decide where to go next. In addition, an overview of the software built for the algorithm will be presented, as well as results of experiments using Evolving Ants.

Keywords: ant colony optimization, ant system, genetic algorithm, genetic information, L-System genetic encoding, social behavior, emergence, evolving ants, heuristic optimization.

1 Introduction

Ant Colony Optimization (ACO) is a metaheuristic inspired by natural colonies of ants in search for food, and a useful technique to find approximate optimal solutions to the shortest-path problem in a directed graph, a well-known NP problem. In this paper I present Evolving-Ants (EVA), an algorithm in which I will combine a standard Genetic Algorithm (GA) with the ACO algorithm know as Ant System.

In the usual implementation of most ACO algorithms individual ants have no memory. Instead, one can think that the particular configuration of the pheromones in the environment constitute a kind of "shared memory" that changes over time. In EVA, a single ant still uses this *stigmergic* (indirect communication) information but in combination with a form of "individual memory" contained in its genetic information. In this context, an ant's genes are encoded using a simple L-System. The ant's L-System will be iterated a fixed number of times, and the structure of the resulting string will contribute to the ant's decision via a mapping function from the space of possible strings to a probability distribution over the string's fix alphabet.

The distribution expresses the probability of the ant taking a certain decision at each step in time. Note that EVA allows for different genetic encoding, not just one based on L-System. Whatever the encoding may be, this needs to be translated into a probability distribution that the ant will use to make a decision. For this particular implementation you can find an L-System and a Simple gene based on a fix-length string.

EvolvingAnts' genetic algorithm implements the usual genetic algorithm operations, i.e., crossover and mutation, specially engineered to work with an L-System (see section 3). At each run of the algorithm, only those ant's that were capable of constructing a "good" path according to some fitness function, usually the length of the path, will be selected for reproduction and thus, will pass their genes (L-System) vertically to the next generation. As part of the entire process of birth (random generation), combination and mutation of an ant, one can think of the resulting string of the L-System as the ant's phenotype as expressed by its genes through a developmental process. The genes encode information useful for the ant's search of the best path and, as generations pass, these genes change to meet the demand of a particular environment.

In the next sections I will present the structure of the algorithm in detail, including the genetic encoding and functions used by the ants, as well as specific purpose software created for EVA and experiments performed on the algorithm.

2 EvolvingAnts

The following is a pseudo code of the EVA algorithm.

```
Initialize Parameters (ACO,GA,EvolvingAnts)
Initialize Random Population of Ants
While stop-condition-not-reached do
  Node = start node
  For each Ant in Population do
    While node-is-not-goal do
      Node = selectNextNode(node,ant)
  Run Genetic Algorithm with solutions found

Function selectNextNode(node n,ant a)
  For each node reachable from n do
    Individual-Decision = a.develop()
    Collective-Decision = getPheromoneTrail(n)
    R = uniformRandomNumber(0,1)
    Choose node n' such that R > (Individual-Decision +
Collective-Decision) / normalizer factor
  Return n'
```

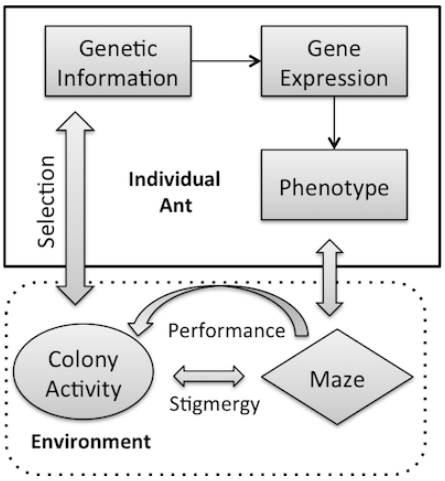
In the function selectNextNode one can see how an ant takes into account both its phenotype (as encoded by its genes) as well as the trail pheromone of the colony. This function is explored in more details in section 3.5. Also, notice how the genetic algo-

rithm is run each time after a run of the traditional ACO algorithm. In this way, the genetic algorithm can use the solutions of each ant as input for a fitness function.

3 Evolutionary Strategy

In this section I explain the details of the genetic encoding and genetic algorithm use for EVA's genetic algorithm. You can find two evolutionary strategies for this particular implementation of EVA: Simple and L-System. Each of these genetic design choices represents a possible way to encode information into the ant's genes.

EVA's genetic encoding and decoding should be thought of as a genotype/phenotype mapping in which each individual ant encode some information about itself and the environment, and later decodes it through a developmental or growth process (phenotype mapping) to work on the environment. For instance, the L-System phenotype mapping would be the iteration of the axiom with some rule until a string (phenotype) is obtained. The following is a schematic of this process.



EVA's phenotype/genotype mapping and developmental process

There can be many more choices than only these two: one just needs a function that maps from some form of information storage to a probability distribution. The study of these choices can be a thread of research on its own right. For instance, another possible strategy would be to build a cellular automaton into each ant, iterate it a number of time, and build a distribution based on the final state of the automata. This distribution can be a function of the number of states in some neighbors, or a majority rule, etc.

The software framework constructed for EVA allows the user to try new evolutionary strategies with no impact to the general structure of the algorithm (see appendix b). One would only need to extend a couple of classes to implement a new genetic encoding.

3.1 Simple Genetic Encoding

In this strategy, an ant's gene is a string of fix length from the four-letter alphabet $\Sigma = \{U, D, L, R\}$. The probability distribution is the proportion of each letter in the string, each of which map to an action, i.e.: U move up, D move down, L move left and R move right. This encoding is mainly used as a control case to test against more sophisticated strategies such as L-System.

3.2 L-System Genetic Encoding

In this strategy, an ant's genes $G = (\Sigma, \sigma, F_i)$, represent a L-System composed of a four-letter alphabet $\Sigma = \{U, D, L, R\}$, and axiom $\sigma \in \Sigma^*$, where Σ^* is the set of all possible strings generated with alphabet Σ, and four rules $F_i: x \rightarrow \Sigma^*, 0 < i < 3$, where $x \in \Sigma$. For computational purposes, Σ^* is allowed to take a fixed maximum number of symbols from Σ. The symbols of Σ are mapped to specific actions in the environment. For the version of EVA discussed in this paper, the symbols stand for actions in a maze mapped as follow: U move up, D move down, L move left and R move right.

The crossover operator takes two genes G_u, G_v, and a random number R uniformly distributed between 0 and 3, and replaces the rules (F_0, F_r) of gene G_u with rules (F_r, F_3) of gene G_v. A single gene can also be mutated, meaning that a rule F_j will be changed randomly.

The expression of a gene, which can be also though of as its phenotype, is equivalent the iteration of the L-System a fix number of times resulting in string S. Once the string is obtained, a function $h(G): S \rightarrow p(\Sigma)$ maps the space of possible strings to a probability distribution over Σ, which is used as weights that the ants will take into consideration when exploring the maze. In this version of EVA, h is implemented as the proportion of each letter in S.

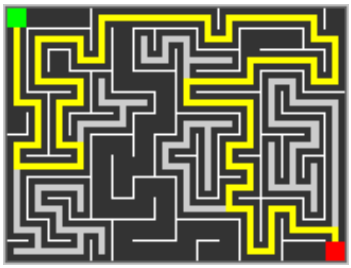
An L-System is a good choice for the purpose of EVA for three main reasons: (i) there is a clear genotype/phenotype mapping between the system's axiom and final string, through a developmental phase; (ii) they encode self-similar information which can be useful in a maze structure where a lot of the decision are similar; and (iii) they compact a lot of information effectively. In short, the L-System stands as a suitable metaphor for an ant's genetic information.

3.3 Genetic Algorithm

EVA employs a standard genetic algorithm to evolve fitter ants. The algorithm uses a roulette wheel selection mechanism whereby individuals are assigned portions of the roulette according to their fitness value. The fitness value is the length of the path constructed by the individual immediately before executing the genetic algorithm. Each pair of individuals, selected from the roulette wheel, is paired using the crossover operation as described before, and the descendants replace both parents. An elitist mechanism is employed whereby a fix number of elite individuals are allowed to be in the next generation without crossing them with other individuals. All individuals, including elite one, are subject to mutation.

3.4 Environment

The environment considered for this first implementation of EVA was that of a perfect maze, i.e., a maze that has one and only one path from any cell in the maze to any other cell. For the purpose of EVA, a cell is to be though of as a node in a graph. Thus, the task of the individual ants is to construct a path between the nest (start) node and the food (goal) node. The dimension of a maze is the number of cells, i.e., nodes in the graph. The structure of the maze is such that an ant only has a maximum of four reachable nodes at any node in the graph. This allows a nice mapping between the alphabet of an ant's genes and possible actions.



Example of a perfect maze used to test EVA

3.5 Ants' Decision Making Process

Ant k positioned at node i will make the decision to move to node j proportionally to the rule: $p(k)_{i,j} = \alpha \cdot G(k) + \beta \cdot T(i,j)$, where parameters α and β regulate the influence of genomic information G and pheromone trail T, individual and shared memory respectively. Both parameters α and β are real numbers between 0 and 1. The final computation is normalized to be a number between 0 and 1. Note that by setting $\alpha = 0$ and $\beta = 1$, EVA will only use pheromone information and thus become a standard ACO algorithm. If, $\alpha = 1$ and $\beta = 0$, EVA uses only genetic information. In this sense EVA generalizes both ACO and GA.

4 Experiments

4.1 Setup

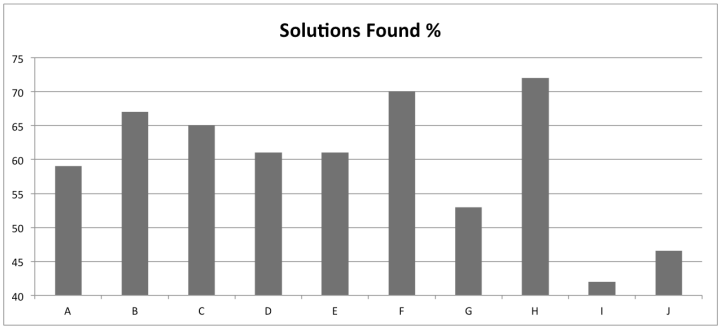
Mazes of different structure were created to test EvolvingAnts. Specifically, the structure of the mazes varied from 20 cells to 300 cells, in increments of 10 cells, i.e., 20, 30, 40, …, 290 and 300. In addition, for each maze of a certain dimension (number of cells), 10 random submazes with different number of columns (5, 10 and 20) were created. In the present implementation, a maze can have a certain number of columns according to its dimension. A maze of dimension D can only have number of cells C if $D \bmod C = 0$, so that the resulting maze is square. For instance, a maze of dimension 110 can only have submazes with 5 or 10 columns. For short, let us denote

a maze as $M(dimension, columns)$, e.g., $M(200,20)$ denotes a maze of dimension 200 and 20 rows. Note that the maze $M(220,15)$ is not a feasible maze. Also note that in the experiments the starting node (nest) is always node zero (located at the upper-left corner of the maze), and the end node (food) is the last node (located at the lower-right corner), e.g., in a maze $M(120,10)$, the food is located at node 119.

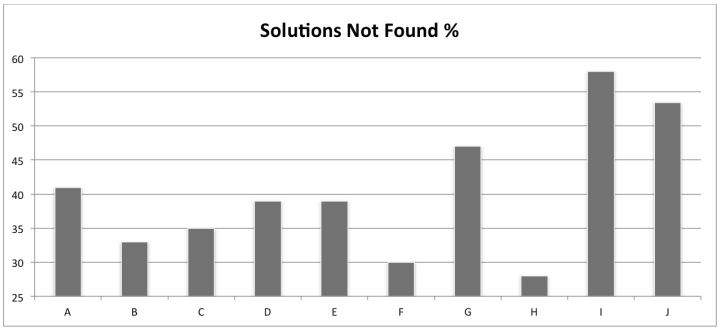
A trial of EVA is one run of the algorithm with all parameters fixed. In total, 820 possible mazes were created within the constraints previously described. Each maze was tested in 100 trials. The metrics used to test the performance of EVA were: (1) average number of iterations to find the solution and (2) whether or not a solution was found. These metrics make sense because each maze has only one possible solution. Moreover, these metrics are easily comparable with other algorithms. In this paper we compare EVA with the canonical ACO algorithm, i.e., AntSystem. The idea is to see the effect of the combination of individual and shared memory in the algorithm's ability to solve the problem at hand.

Note that the average number of iterations in metric (1) refers to the iterations of EVA in different trials of the algorithm, and that each iteration of EVA contains a run of the Genetic Algorithm with a fixed number of iterations. The parameters used in the experiments can be seen on appendix a.

4.2 Results



Percentage of EVA-solvable mazes (y-axis). Parameter Set (x-axis).

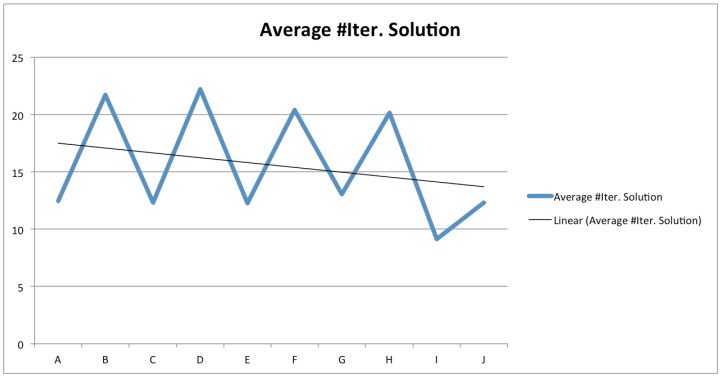


Percentage of EVA-unsolvable mazes (y-axis). Parameter Set (x-axis).

The graphs above shows the percentage of solutions found per set of parameters. The mazes for which results are incorporated in all graphs on this section are $M(d, c)$ where $20 \leq d \leq 300$ and $5 \leq c \leq 20$. For detailed results please see appendix c.

Overall, EVA (parameters sets A through H) is able to find solutions in more than half of the mazes, while ACO (parameters I and J) only finds solutions approximately in 45% of all mazes. Moreover, simple gene-EVA (A-D) achieves an average of 63% success, while L-System-EVA (E-H) an average of 64%. However, L-System-EVA parameter F and H outperforms every other configuration of the algorithm, and reaches the maximum of 71% average success. The best configuration is L-System-EVA parameter H, i.e., $\alpha = 0.75$ and $\beta = 0.25$, meaning that a larger weight for the genetic algorithm relative to the ACO algorithm results in better performance. The importance of the genetic algorithm becomes obvious once we realize that L-System-EVA set G, i.e., $\alpha = 0.25$ and $\beta = 0.75$, underperforms every other configuration (except from pure ACO). Therefore, the weight of the evolutionary strategy is very important for L-System, in which an ant's gene can grow and develop more useful structure to find solutions. However, the evolutionary strategy is not as important for the simple gene in which all results are fairly clustered around the mean success percentage, regardless of the values of α and β.

The following graph shows the average number of iteration per parameter set in case a solution was obtained. The graph also shows a linear trend line over all results.



Average number of iterations, and linear trend, when a solution is found

From the graph above one can see that there is a downward trend on average number of average iterations to find a solution. The trend goes in relation with the parameters set; meaning that in average set A took longer than set J. As expected, the maximum number of EVA-iterations dominates the number of iterations of a particular set in a given trial, however, the overall downward trend shows an interesting relationship on the number of iterations and the evolutionary strategy, i.e., if a solution is found, the L-System takes fewer iterations than the simple gene. For instance, set H employs less number of iterations than set D; even tough both sets use 100 EVA-iterations. In other words, the fact that the L-System encodes information more effec-

tively contributes to both number of solution found and speed in terms of number of iterations.

These results show that EVA outperforms ACO (both in terms of number of solutions found and average number of iterations), regardless of the evolutionary strategy employed (Simple or L-System's genes), but that different strategies may yield radically different results. A strategy based on a fixed characterization of the genetic information underperforms relative to another strategy that is allowed to grow and develop a better encoding of the environment. For these experiments, a strategy that employs an L-Systems and weights more the genetic information when making a decision (set H) prove to be the best strategy. However, the evolutionary strategy must be given enough number of iterations to produce such results, otherwise one can obtain a considerable poorer performance (set G).

These results point to the fact that the L-System makes a big difference in the encoding of useful information from the environment and the colony itself and, if given enough time, the resulting phenotype (L-System string) will guide the ants towards solutions faster than other configurations tested in these experiments.

5 Conclusion

Evolving Ants (EVA) is an innovative combination of a canonical genetic algorithm, i.e., using roulette wheel selection, random crossover point, fair probability of crossover and low probability of mutation; with a simple ant colony optimization algorithm, in which a population of ants is provided with genetic information that it uses in combination with *stigmergic* information to select a next node to visit in a path. In this context, the genetic information is to be thought of as "individual" memory whereas the *stigmergic* information, i.e., the pheromone trail, is to be thought of as the "shared" memory among individual agents in the colony. This paper covered the following topics: (1) introduction of EVA, (2) justification of EVA as a viable generalization of the ACO metaheuristic, (3) two different design alternatives for EVA's evolutionary strategy (Simple and L-System's genes), and (4) results to experiments performed in a maze in which the ant colony needed to find a path from their nest to a (fixed) food source.

From the results of the experiments performed using EVA, one can conclude that this algorithm outperformed a pure ACO algorithm when tested under the conditions described in section 3.4 and 4.1. Moreover, it was shown that the performance of EVA varies drastically with the choice of the genetic encoding of the evolutionary algorithm, as described in section 3. An encoding that maps directly from genes to phenotype, and is fixed throughout the execution of the system, exhibits a lower performance than a strategy that uses an intermediary, growth phase, in which genes are expressed into a phenotype which is later mapped to a decision in the maze. The evidence that supports this concludes follows from the fact in average the EVA-L-System outperformed the Simple gene encoding. As a continuation of this research, new evolutionary strategies can be tested, e.g., cellular automata; and the algorithm extended to solve other non-mazes environment easily modeled by graphs.

References

- Ant Colony based software framework to solve optimization problems" Enrique Areyan and Ignacio Calderon. Jornadas de Investigación y Extensión. Facultad de Ciencias 2010. Universidad Central de Venezuela.
- Nunes de Castro, Leandro [2006]. Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications. Chapman & Hall.
- http://www.mazeworks.com/mazegen/mazetut/index.htm
- I585 Bioinspired computing resources, Indiana University. Instructor: Dr. Luis Rocha. http://informatics.indiana.edu/rocha/i-bic/index.html
  Course's blog. http://life-inspired.blogspot.com/

Appendix A: Parameters

The following table contains ten sets of parameters, **A** trough **J**, used to test EVA (see section 4.1). Parameters **A** through **D** correspond to runs of the algorithm using the Simple gene. Parameters **E** through **H** correspond to runs of the algorithm using L-System encoding. Parameters **I** and **J** correspond to a run of traditional ACO. Other fixed parameters are: pheromone evaporation rate 0.65, mutation rate 0.05, and cross-over rate 0.6. **agImportance** and **acoImportance** correspond to α and β respectively (see section 3.5)

Parameter Set	A	Parameter Set	B
ACO		ACO	
numberOfAnts	15	numberOfAnts	20
GA		GA	
maxNumIterations	35	maxNumIterations	50
EVA		EVA	
maxNumIterations	50	maxNumIterations	100
agImportance	0.5	agImportance	0.5
acoImportance	0.5	acoImportance	0.5

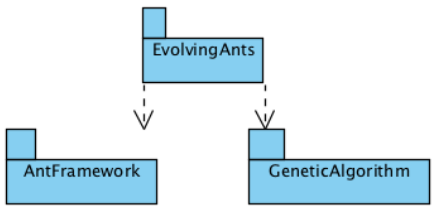
Parameter Set	C	Parameter Set	D
ACO		ACO	
numberOfAnts	15	numberOfAnts	20
GA		GA	
maxNumIterations	35	maxNumIterations	50
EVA		EVA	
maxNumIterations	50	maxNumIterations	100
agImportance	0.75	agImportance	0.25
acoImportance	0.25	acoImportance	0.75

Parameter Set	E	Parameter Set	F
ACO		ACO	
numberOfAnts	15	numberOfAnts	20
GA		GA	
maxNumIterations	35	maxNumIterations	50
EVA		EVA	
maxNumIterations	50	maxNumIterations	100
agImportance	0.5	agImportance	0.5
acoImportance	0.5	acoImportance	0.5

Parameter Set	G	Parameter Set	H
ACO		ACO	
numberOfAnts	15	numberOfAnts	20
GA		GA	
maxNumIterations	35	maxNumIterations	50
EVA		EVA	
maxNumIterations	50	maxNumIterations	100
agImportance	0.25	agImportance	0.75
acoImportance	0.75	acoImportance	0.25

Parameter Set	I	Parameter Set	J
ACO		ACO	
numberOfAnts	15	numberOfAnts	20
EVA		EVA	
maxNumIterations	35	maxNumIterations	50
agImportance	0	agImportance	0
acoImportance	1	acoImportance	1

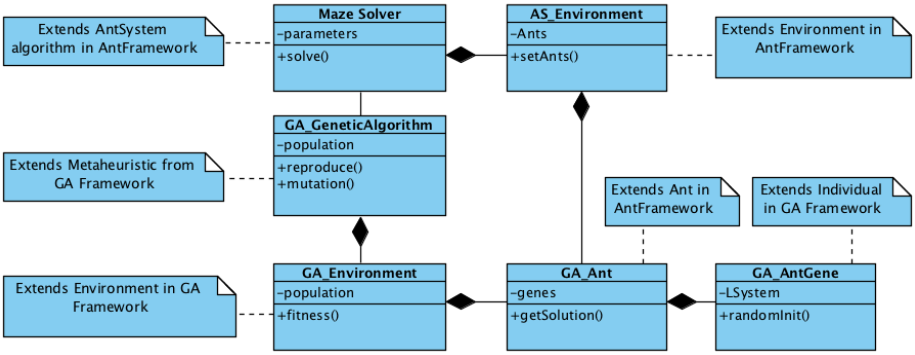
Appendix B: Software Framework



EVA's package diagram

To support EVA, two separated object-oriented frameworks were combined. The first, AntFramework, incorporates pre-prepared objects to work with traditional ACO algorithms. For the present implementation, only the ACO algorithm known as Ant System was used. In the future, EVA can be extended to incorporate other ACO algorithms. The second framework, specially built for this project, integrates abstract objects for the Genetic Algorithm's tasks, and it can be extended to suit other algorithms or different genetic encodings.

In addition, this implementation of EVA called for special software to build random mazes given arbitrary dimensions. The mazes were translated into graphs, which encoded the appropriate logic to represent the mazes' structure. Finally, the algorithm explores the graphs' adjacency matrix, which means that it can be use to solve virtually any problem that can be modeled as a graph. A schematic representation of EVA is show in the next diagram.



EVA's class diagram corresponding to the package EvolvingAnts

EVA combines the AntFramework and GA Framework by using a common set of Ants, depicted as GA_Ant in the above diagram. This ant object is an extension of the simple Ant in the AntFramework, but incorporates a gene object to encode the ant's L-System. Thus, as discussed earlier, in this architecture an Ant is provided with genetic information.

One can change the GA_AntGene class to encode a different kind of genetic information. Thus, one concrete implementation of GA_AntGene corresponds to the Simple gene and another to the L-System gene. The only requirements are that the encoding must be consistent with the process described in section 3.4 and the environment being used. However, one can always change both the genetic encoding and phenotype developmental process, as well as the actions valid in the environment with little impact to other objects.

Appendix C: Web Resources

For more EVA-information: **http://www.enriqueareyan.com/?q=evolvingants**