Environmental influence in bio-inspired game level solver algorithms

Antonio Gonzalez-Pardo and David Camacho

Abstract Bio-inspired algorithms have been widely used to solve problems in areas like heuristic search, classical optimization, or optimum configuration in complex systems. This paper studies how Genetic Algorithms (GA) and Ant Colony Optimization (ACO) algorithms can be applied to automatically solve levels in the well known Lemmings Game. The main goal of this work is to study the influence that the environment exerts over these algorithms, specially when the goal of the selected game is to save an individual (lemming) that should take into account their environment to improve their possibilities of survival. The experimental evaluations carried out reveals that the performance of the algorithm (i.e. number of paths found) is improve when the algorithm uses a small quantity of information about the environment.

1 Introduction

Bio-inspired algorithm research field has been widely used to solve problems or to search for the optimum configuration of complex systems. Due to these type of problems exhibit *NP-complete* or *NP-hard* complexity, the resolution process needs a huge amount of resources (such as computational effort or time). Some examples of such problems are *scheduling problems*, *constrained satisfaction problems*, or *routing problems*.

A good strategy to reduce the time needed to solve NP-complete problems is applying bio-inspired algorithms, such as evolutionary algorithm (EA) [Fogel, 1995, Eiben and Smith, 2009] or swarm intelligence [Engelbrecht, 2007].

Antonio Gonzalez-Pardo, David Camacho
Computer Science Department
Escuela Politécnica Superior
Universidad Autónoma de Madrid,
e-mail: {antonio.gonzalez,david.camacho}@uam.es

These types of algorithms work with a population of possible solutions that navigates through the solution space of the modelled problem.

In the case of EA, each individual is evaluated by a fitness function that allows their comparison. Then, individuals with better fitness value generates the next population by the use of the crossover and mutation [Forrest, 1993]. These operators allow the generation of new individuals taking into account their parents' characteristic.

Swarm intelligence algorithms focus on the collective behaviour of self-organizing systems [Farooq, 2008] where the iterations among individuals generate collective knowledge based on social colonies [Karaboga, 2005]. In this case, the initial population travels through the solution space in order to obtain the best solution to the problem.

In this paper a classical Ant Colony Optimization [Dorigo, 1999] and a Genetic Algorithm are applied to the well-known *Lemmings Game* to determine whether the different levels can be solved using the given skills.

The Lemmings Game is a popular proven NP-hard puzzle game [Cormode, 2004]. In spite of the popularity that this game obtained in the 1990s, few research has been applied to it. Computational intelligence has just been applied to video games such as Mastermind, the Art of Zen, Ms Pac-Man, Tetris or Mario Bros.

This paper tries to determine the influence that environment (in this case, the Lemmings level) exerts over the different algorithms. In order to do that the performance of a Genetic Algorithm (GA), an Ant Colony Optimization (ACO) and an heuristic for the ACO will be analysed.

2 The Lemmings Game

Lemmings are creatures that need to be saved. In each level, Lemmings start in a specific point of the stage and must be guided to the exit point by the player. They live in a two-dimensional space and are affected by gravity. They start walking in a specific direction until they find an obstacle. In this case the Lemming will change the direction and walk back. If the Lemming encounters a hole, it will fall down. The only two ways, considered in this paper, by which a Lemming can die is by falling beyond a certain distance or by falling from the bottom of the level.

In order to make Lemmings to reach the exit point, players have a set of ”skills” that must be given (not necessarily all of them) to the Lemmings. Using these skills, Lemmings can modify the environment creating tunnels, or bridges, and thus creating a new way to reach the exit. There are eight different skills that allow Lemming to have an umbrella, to dig or to climb, amongst others. Each skill can be used (i.e. assigned) a maximum number of times. It is not necessary to use all of the skills in the levels.

In the Lemmings' world, there are a huge number of materials, but all of them can be grouped in two different classes: the ones that can be modified (e.g. can be dug) and the ones that cannot be altered. In the case that a Lemming is digging

and finds a material that cannot be dug, the Lemming will stop digging and start walking.

3 Description of the studied algorithms

In this paper GA and ACO algorithms are studied. The aim of the experimental phase is to determine whether these algorithms can find the different paths that guide Lemmings to the exit point of the level.

3.1 Genetic Algorithm

The GA applied in this work, initializes individuals with a random phenotype length. The maximum length of the phenotype depends on the maximum time of the level or the maximum genotype length allowed. The phenotype is a list of genes where each gene ($< T, S >$) contains the skill (S) that is going to be executed in the step T. Both values (the step and the skill) are selected randomly depending on the maximum time given to solve the level, and the total number of remaining skills. The phenotype represents the different decisions that the player could make. This phenotype is then evaluated against the level. The lemming starts its execution applying the skills specified in the given steps.

The goal of the GA is to maximize the fitness function represented by Eq. 4 and it is composed by Eq. 1, Eq. 2 and Eq. 3. Eq. 1 takes into account the time spent by the Lemming to solve the level. Eq. 2 is used to favour those paths that use less actions, or less skills. Finally, another key concept is the number of lemmings saved in the level represented in Eq. 3.

$$T(Ind_i) = MaxTime - Time(Ind_i) \quad (1)$$

$$A(Ind_i) = TotalActGiv - ActionUsed(Ind_i) \quad (2)$$

$$S(Ind_i) = TotalLemm - BlockersUsed(Ind_i) - ExplodedLemmings(Ind_i) \quad (3)$$

$$F(Ind_i) = \frac{T(Ind_i) + A(Ind_i) + S(Ind_i)}{MaxTime + TotalActGiv + TotalLemm} \quad (4)$$

Although ACO will use the same function to evaluate the goodness of the paths, only GA can produce negative fitness. This negative fitness value is obtained if the individual produces an invalid path (i.e. in the evaluation, the lemming is not able to reach the exit point or the lemming dies trying it). In this case the fitness value is $-1 * F(Ind_i)$.

3.2 Ant Colony Optimization

In this work, the ants are the Lemmings of the game. So, they start in the entry point of the level and at each position, each ant will decide whether to continue executing the current action or to select another skill to execute. This decision depends on the number of remaining skills and the pheromones deposited in the current location. This means that skills with a higher pheromone value, and/or that can be applied several times, have more chances to be selected. Once the ant decides to change its behaviour, it deposits a pheromone in the current location and continues with its execution.

A pheromone is an object that represents a decision taken previously by any ant. It means that some ant had taken a specific skill at this point. For that reason each pheromone contains the name of the skill taken, the direction in which this skill had been executed and a value representing the goodness for this decision (this value is computed at the end of the ant execution).

In this work when any ant reaches the exit of the level, it will start again from the entry point updating all the pheromones with the fitness value of the path. This fitness is the same as the one used by the GA (Eq. 4). Finally, when all pheromones have been updated and the ant arrives to the destination (for the second time), it will forget the followed path and it will start again as a new ant, without any knowledge about the location of the exit.

3.3 The common-sense ant

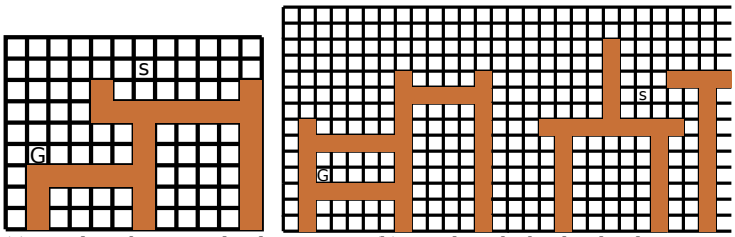
The problem with the described ACO algorithm is that randomness may generate strange situation like having an ant falling down and suddenly the ants start building a bridge in the air. For this reason, a common-sense heuristic is defined.

With this heuristic, the current skill applied and the ants' environment will influence in the decision of the next skill to apply. This heuristic will avoid the use of a skill when it cannot be executed, for example, it is not useful to apply the Climber skill if the ant is not in front of a wall.

4 Experimental Phase

The aim of the experiments is to analyse the influence of the environment in the behaviour of a GA and an ACO algorithms. Five different levels have been designed, the easiest one and the hardest one are shown in Fig. 1(a) and Fig. 1(b), respectively. This valuation is based on the size of the level, the different blocks contained into each level, the distance from the entry point to the exit point, the number of skills needed to solve the level, etc. It is important to take into account that all the terrains used in the levels are editable terrain, i. e. ants can dig, climb and bash into it.

For each level, GA, ACO and ACO with common-sense ants are executed. Each execution is repeated 50 times because all algorithms are stochastic. In order



(a) Level 1. The easiest level (b) Level 5. The hardest level

Fig. 1 Examples of the easiest and the hardest level modelled in this work.

to compare the results obtained among the different algorithms there are some parameters that are common to all of them. Each experiment uses the Eq. 4 to evaluate the different paths, the algorithms execute 100 ants (or individuals) during 200 iterations(or generations), and each experiment is repeated 50 times.

The GAs have, also, the following configuration: the maximum phenotype length is 20, the probability of having One-point crossover is 90%, while the mutation rate is fixed to 10%. There is not elitism and the goal is to maximize the results obtained from the fitness function.

The goal of the experiments is to compare the number of different solutions that each algorithm is able to build. This information is shown in Table 1. The values correspond to the number of different paths found in 50 executions of the algorithms.

Table 1 Number of different solutions found by the algorithms described.

Level	# Different Solutions (Genetic Algorithm)	# Different Solutions (ACO Random Ant)	# Different Solutions (ACO Common-Sense)
1	3219	3868	2516
2	12629	4463	4042
3	370	1130	2487
4	2	15	32
5	0	3	7

5 Conclusions

This paper analyses the possibility of applying Genetic Algorithm and Ant Colony Optimization to generate automatic game level solver tools. The application domain of this work is the well-known Lemmings game, where Lemmings have to apply different skills in order to reach the exit. Five different levels have been designed with different complexity depending on the size of the level, the number of available skills, or the distance between the start and the destination point, amongst others.

Experimental results reveal that both algorithms can successfully be applied to solve the levels. Nevertheless, as it can be seen in Table 1 the Genetic Algorithms provide less different paths when the levels are harder (i.e. levels 4 and 5). This is

produced because GA generates individuals without taking into account the level landscape (i.e. it is a blind generation of individuals). On the other hand, ACO uses the terrain information to apply different skills at specific steps and thus, it provides better results.

One of the problem observed in this work is that GA does not guarantee the goodness of the following generation of individuals. Parents are selected depending on its fitness value (better fitness value implies more chances for being selected), but crossover and mutation operations do no guarantee that the generated children have better fitness values. Ant Colony Optimization does not have this problem, because ants are guided by the pheromone trails and then good decision (represented in pheromones with high values) will have higher probabilities for being selected.

In order to determine whether the environment is important for generating automatic level solvers, an heuristic for ACO has been designed. Although this heuristic (called *common-sense heuristic*) provides less different paths in easiest levels (level 1 and 2), it provides better results when it tries to solved the hardest levels (levels 3, 4 and 5). This fact demonstrates that level information is very important to make automatic level solvers.

Nevertheless, the design of domain based heuristics makes the platform very dependent on the domain and thus the conclusions, and the approach, are not applicable to other different domains.

Acknowledgements This work has been supported by the Spanish Ministry of Science and Innovation under grant TIN2010-19872.

References

[Cormode, 2004] Cormode, G. (2004). The hardness of the lemmings game, or oh no, more npcompleteness proofs. In *In Proceedings of Third International Conference on Fun with Algorithms*, pages 65–76.

[Dorigo, 1999] Dorigo, M. (1999). Ant colony optimization: A new meta-heuristic. In *Proceedings of the Congress on Evolutionary Computation*, pages 1470–1477. IEEE Press.

[Eiben and Smith, 2009] Eiben, A. E. and Smith, J. E. (2009). *Introduction to Evolutionary Computing*. Springer-Verlag.

[Engelbrecht, 2007] Engelbrecht, A. (2007). *Computational Intelligence: An Introduction*. Wiley Publishing, 2nd edition.

[Farooq, 2008] Farooq, M. (2008). *Bee-Inspired Protocol Engineering: From Nature to Networks*. Springer Publishing Company, Incorporated.

[Fogel, 1995] Fogel, D. B. (1995). *Evolutionary computation: toward a new philosophy of machine intelligence*. IEEE Press.

[Forrest, 1993] Forrest, S. (1993). Genetic algorithms: principles of natural selection applied to computation. *Science*, 261, No. 5123:872–878.

[Karaboga, 2005] Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Techn Rep TR06 Erciyes Univ Press Erciyes*, 129(2):2865.