Artificial Ant Colonies and E-Learning: An Optimisation of Pedagogical Paths

Y. SEMET[1]*, Y. JAMONT, R. BIOJOUT, E. LUTTON, P. COLLET*

Projet Fractales - INRIA Rocquencourt, B.P. 105
FR-78153 Le Chesnay Cedex, France
Yann.Semet@tremplin-utc.net
[Evelyne.Lutton;Pierre.Collet]@inria.fr
[Yannick.Jamont;Raphael.Biojout]@paraschool.com

Abstract

This paper describes current research on the optimisation of the pedagogical path of a student in an existing e-learning software. This optimisation is performed following the models given by a fairly recent field of Artificial Intelligence: Ant Colony Optimisation (ACO) [1,2,4]. The underlying structure of the E-learning material is represented by a graph with valued arcs whose weights are optimised by virtual ants that release virtual pheromones along their paths. This gradual modification of the graph's structure improves its pedagogic pertinence in order to increase pedagogic success. The system is developed for Paraschool, the leading French E-learning company. Tests will be conducted on a pool of more than 10,000 users.

Keywords: E-Learning, Ant Colony Optimisation (ACO), Evolutionary Computation.

1 Introduction

The e-learning software of the French Paraschool company offers a complement to high-school teaching. The software is used in schools, over a LAN, with a supervising teacher, or from home over the Internet. It contains small tutorials, exercises and multiple-choice questions that allow students to practice on their own. The original software provided deterministic HTML links and Paraschool was looking for a system that would enhance the navigation by making it adaptive and user-specific, so that both individual profiles and collective characteristics could be taken into account in an automatic and dynamic fashion. For instance, some successions of lessons may prove particularly successful in helping students understand a particular notion and those successions, leading to high success rates in subsequent exercises, should be automatically detected and highlighted.

1.1 Evolutionary Computation and Ant Colony Optimisation

Numerous difficulties pertaining to this problem (multiple contradictory objectives, fuzziness, complexity) immediately ring the bell of evolutionary techniques (a set of AI engineering tools of bio-mimetic inspiration), among which Ant Colony Optimisation (ACO) seems particularly well suited. This subfield of Evolutionary Computation comes from the observation of actual ant colonies and social insects in general such as bees or termites, and of their extraordinary abilities

[1] Université de Technologie de Compiègne, BP 60319, 60203 Compiègne Cedex, France.

to co-operate at the individual level to trigger complex and intelligent behaviour at the global level, an emerging phenomenon also known as “Swarm Intelligence” [1,4]. The ants' ability to come up with optimal paths to fetch food, for example, through the release of chemicals along their way is very remarkable and modelling this simple idea yielded exciting results in the field of combinatorial optimisation [5] (efficient heuristic solving of the Travelling Salesman Problem (TSP), routing problems, etc.).

Applying ACO techniques in Paraschool’s context seems relatively straightforward when one sees the E-learning software as a graph with valued arcs through which students navigate, following suggestions made by the system and where:

- nodes are pedagogical items (exercises, lessons, quizzes, etc.);
- arcs are hypertext links between those items;
- weights on the arcs reflect the probabilities to suggest subsequent nodes to students;
- original weights are determined by the pedagogical team of Paraschool.

The task of the ACO is to optimise the weights on the arcs in order to maximise student success.

Besides their efficiency to quickly reach near-optimal solutions, ACO algorithms are also especially appreciated for their robustness and adaptability: just as natural ant colonies quickly find a new source of food when one disappears, ACO algorithms quickly find new optimal paths when the underlying graph suddenly changes. In Paraschool’s case, optimising a graph with respect to some dynamic cognitive behaviour at both an individual (with multiple instances) and a collective level therefore seems like a perfect job for ACO algorithms. The transposition, in particular, from the work successfully carried out with ants on TSPs is again straightforward: each student going through the graph is represented by a virtual ant that releases virtual pheromones (concretely by incrementing floating point values carried by the arc) proportionally to its amount of successes and failures. Out of this information, stored in the “environment” and called “stigmergic” information [1], emerges a representation of the interaction between the students and the pedagogic material. This representation is used to derive probabilities that dictate the forthcoming behaviour of the software. The key advantage of this system is that this representation is both reactive and robust. And this is so, firstly because pheromones evaporate with time -which prevents the system from freezing or converging towards a particular state- and secondly because students, by browsing the graph, continually update the representation, thereby reflecting the dynamics of their needs.

2 Features and specifications

2.1 Pedagogic weights

The pedagogical team gives a weight *W* to each arc, reflecting its importance with respect to other arcs coming out of the same node. This describes the pedagogic structure of the site: after a given lesson, the user can follow several possible arcs; the relevance of which is indicated by *W*. The higher *W*, the more adequate it is for students to follow the corresponding arc.

2.2 Pheromone release and evaporation

Following the validation of a node, an ant (i.e. a student) releases pheromones along the way that led it to that node. There are two kinds of pheromones: one for successes (*S*), one for failures (*F*). Pheromones are released backwards in time along the ant’s path starting from the last validated node with decreasing amplitude. This is meant to reflect the fact that all the nodes a student went through previously have an influence on its ability to succeed in validating its current node. Of course, this influence should decrease with time: the more ancient a visit to a node, the less influence it has. This “back propagation” of pheromone release is limited in scope for obvious

reasons (from both the algorithmic and pedagogic standpoints) and a number of nodes is thus set by the pedagogical team after which the back propagation stops. A typical value of 4 nodes has been agreed upon. In addition, pheromones released as stated above evaporate with time: their values tend to go back to 0 if the corresponding arc is unused for a long time. This is meant to make the system adaptive and to prevent it from being trapped in a particular state.

2.3 *H-node*s: historic weight computation and evaporation

In order to adapt the system to each student, track is kept -i.e. stored in a database- of each node visited by a student, not only in the present sessions but in all of his/her previous sessions as well. For each node and for each student a historic weight *H* is stored in a *H-node* with a default value of 1.0, meaning that the node has not been visited yet. When the node is visited, the value is multiplied either by *h1* (if it is a success) or by *h2* (if it is a failure). Values for *h1* and *h2* can be tuned, but typically, *h1=0.5* and *h2=0.75*. This *H* value is going to be used (see below for details on how) to discourage a student from visiting a node he/she has already seen, this discouraging being weaker when the node was failed. To reflect the fact that a student has limited memory, this *H* value tends to go back to 1.0 with time, along the following equation where *x* is the time elapsed since the last consultation:

$$(1) H_t = H_{t-1}\left(1+\frac{1-H_{t-1}}{H_{t-1}}\cdot\frac{1-e^{-\tau x}}{1+e^{-\tau x}}\right)$$

In the next equation, τ is a time constant that sets the speed of the phenomenon. It should be calibrated to correspond to the volatility of the students' memory:

$$(1) \Leftrightarrow \tau = \frac{1}{x}\cdot\ln\left(\frac{1+\alpha}{1-\alpha}\right) \text{with } \alpha = \frac{H_t - H_{t-1}}{1-H_{t-1}}$$

This latter equation can be used by the pedagogical team to tune the value of τ in a convenient way: provided one defines what "forgetting an exercise" means, for instance if its weight, starting from $H_{t-1} = 0.5$ (1 visit with success), grows back to $H_t = 0.9$, this gives $\alpha \approx 2.2$ and the pedagogic team then only has to estimate the time it takes to "forget an exercise": 1 week for example (x=604800 sec.) gives $\tau \approx 3.6E-6$.

2.4 Fitness calculation

Using all the information described above, each arc *a* is given a *fitness value*:

$$f(a) = H\cdot\left(\omega_1 W + \omega_2 S - \omega_3 F\right)$$

This value unifies in a weighted average all the factors that make an arc "desirable" or not: *f* is high when:

- The arc's ending node was last visited a long time ago (*H* is close to 1)
- The arc is encouraged by professors (high *W*)
- People have succeeded a lot around that node (high *S*)
- People have failed a little around that node (low *F*)

2.5 Arc selection and subsequent suggestion

After a node has been validated, the outgoing arcs are sorted according to this computed fitness value. One arc is randomly selected among the whole list, with a probability that is proportional to its fitness. It is suggested as an adequate follow-up to the student who pressed the ACO-powered NEXT button. A variety of selection procedures has been implemented, taken from

the field of genetic and evolutionary computation, among which: roulette-wheel selection, ranking based methods and stochastic tournament selection. Choosing one method or another gives more or less control on the phenomenon by allowing to tune more or less precisely the amount of randomness this selection procedure is going to have. (cf. [3] for details).

The three approaches have been implemented, but tests have not been thorough enough to determine which method was the best.

3 First Results

Numerous tests have been conducted. First, a simulation procedure has been defined to allow for stabilization and calibration of the various parameters. The algorithm was then applied to the actual Paraschool software.

3.1 Simulations

3.1.1 Modelling the population

A model of user population has been derived to conduct automatic simulation tests:

Each virtual student (i.e. ant), is given a certain *level* represented by a floating point value between 0.0 and 1.0. This value is normally distributed over the population of students with mean 0.5 and standard deviation 1/3. Each exercise is assigned a *difficulty* value, also between 0.0 and 1.0. When an ant arrives at a given node, if its level allows it to validate the node (*level>difficulty*), it succeeds, otherwise, it fails. Pheromones are released accordingly. General calibration of the algorithm was performed on a "real" graph, i.e. corresponding to an actual part of the Paraschool website (the "Vectors" chapter of a mathematics course for high school students around age 14). Arcs between nodes and corresponding weights have been assigned by the Paraschool pedagogical team. The sample case is therefore realistic (20 nodes, 47 arcs) and constitutes a meaningful structure with real size.

3.1.2 A particular test case

Several features are expected from the ant colony. In particular, it should be able to correct inappropriate arc weight values. To investigate this properly, after a rough calibration and observation process conducted on the real sized graph mentioned above, experiments are conducted on a reduced graph that exhibits such a situation:

After solving exercise 1, the student can either go to exercise 2 or to exercise 3 before he ends, in both cases, with exercise 4. Exercise 3 is encouraged by the pedagogical team as the arc leading to it is assigned a weight of 5 versus 1 for the arc leading to exercise 2. The problem is that the success rate of exercise 4 is much higher when the student comes from exercise 2 than when he/she comes from exercise 3. What is expected from the system in such a case is to detect the situation and to reverse the two probabilities so that students are encouraged to follow the right path. This should be achieved naturally, i.e. without any human intervention, thanks to the release of virtual pheromones along the arcs. The arc leading to exercise 2 will hold a large amount of success pheromones and a low amount of failure pheromones. The arc leading to exercise 3, on the contrary is going to be in the opposite situation and this double discrepancy is going to be reflected in the arcs' fitness, thereby modifying their probabilities to be followed. Progressively, the arc leading to exercise 2 takes over the arc leading to exercise 3 and experiments show that a reasonable situation is promptly re-established

In the real-world version, such a discrepancy between weights given by the pedagogic team and evolved weights will issue a warning so that measures can be taken to solve the problem.

3.2 Real world application

The application to the real Paraschool system is only in its early stage. The ant colony algorithm has been integrated to the entire website in a downgraded mode where pheromones are only used to gather information and do not yet influence arc probabilities. Ten days after the integration, 566 “ants” have browsed the site, 2419 arcs have been visited and 3021 *H-nodes* have been created. First observations tend to show that singular nodes (i.e. too easy or too difficult exercises) see corresponding amounts of pheromones cumulate around them (e.g. high *S* and low *F*), giving its first credits to the pheromone representation of the pedagogic structure. From an algorithmic point of view, these initial observations also show that the system is stable and able to handle all the additional computations without any noticeable overhead.

4 Conclusions and outline for future work

Time has now come to analyse the algorithm behaviour while in passive mode and tune the different parameters. When results show that the system is really stable, it will be switched to active mode wherefrom it is hoped that the ACO heuristic will provide:

- a seemingly intelligent system that improves the behaviour of the web site from the student's viewpoint,
- a refined auditing tool to help the pedagogical team identify the strengths and weaknesses of their software and pedagogic material.

From a theoretical perspective, this work brings encouraging elements of answers as to whether the emerging properties of social insect-based models can:

- make proper tools to enhance e-learning systems,
- adequately describe the cognitive behaviour of a social system (students and teachers),
- scale well from small experimental environments to a real-world application involving several thousands of individuals.

As this technique is original in the field of E-Learning and as the observation phase is only beginning, the present study should be seen as pointing out a potentially interesting research direction while great expectations are put in the observation of the forthcoming behaviour of the system.

References

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press 1999. ISBN 0-19-513159-2.

[2] E. Bonabeau, M. Dorigo, and G. Theraulaz . Inspiration for optimization from social insect behaviour. Nature, vol. 406, 6 July 2000.

[3] D. E. Goldberg, K. Deb, A comparative analysis of selection schemes used in genetic algorithms (1991),in FOGA91, vol. 1, pp 69-93.

[4] M. Resnick, Turtles, Termites and Traffic Jams. Explorations in Massively Parallel Microworlds. Complex Adaptive Systems series MIT Press, 1994 .

[5] T. Stützle and M. Dorigo. ACO Algorithms for the Travelling Salesman Problem. In Proceedings of the EUROGEN conference, M Makela, K Miettinen, P Neittaanmaki, J Periaux (Eds), John Wiley & Sons, 1999, ISBN: 0471999024.