

Stigmergic Optimization in Dynamic Binary Landscapes

Carlos Fernandesⁱ

LaSEEB, Technical Univ. of Lisbon
Av. Rovisco Pais, 1, TN 6.21, 1049-001,
Lisbon, PORTUGAL

cfernandes@laseeb.org

Vitorino Ramos

LaSEEB, Technical Univ. of Lisbon
Av. Rovisco Pais, 1, TN 6.21, 1049-001,
Lisbon, PORTUGAL

vitorino.ramos@alfa.ist.utl.pt

Agostinho C. Rosa

LaSEEB, Technical Univ. of Lisbon
Av. Rovisco Pais, 1, TN 6.21, 1049-001,
Lisbon, PORTUGAL

acrosa@laseeb.org

ABSTRACT

Hereafter we introduce a novel algorithm for optimization in dynamic binary landscapes. The *Binary Ant Algorithm* (BAA) mimics some aspects of real social insects' behavior. Like *Ant Colony Optimization* (ACO), BAA acts by building pheromone maps over a grid of possible trails that represent solutions to an optimization problem. Main differences rely on the way this search space is represented and provided to the colony in order to explore/exploit it. Then, by a process of pheromone reinforcement and evaporation the artificial insect trails converge to regions near the problem solution or extrema. The negative feedback granted by the evaporation mechanism provides the self-organized system with population diversity and self-adaptive characteristics, allowing BAA to be particularly suitable for hard *Dynamic Optimization Problems* (DOP), where extrema continuously changes at severe speeds.

Categories and Subject Descriptors: I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods and Search – *Heuristic methods*.

General Terms: Algorithms, Experimentation.

Keywords: Ant algorithms, Stigmergy, Dynamic Optimization.

1. INTRODUCTION

Ant algorithms [2] are one of the most successful examples of Swarm Intelligence and Stigmergic Optimization [1]. They have been applied to a wide set of problems, ranging from TSP to clustering – see [2] for a survey. ACO meta-heuristic [2] defines a particular class of ant algorithms. Based on the ability of natural ants to find the shortest paths to food sources, ACO simulates the ants' process of pheromone deposition and their stochastic tendency to walk in the direction of sensed pheromone. There are several ACO algorithms, each one designed for a specific problem and differing in the transition, reinforcement and evaporation rules, amongst other properties. In general, an ACO algorithm comprises the following steps: pheromone trail initialization, solution construction using pheromone trails and pheromone update (evaporation and reinforcement). These stigmergic mechanisms lead to the emergence of pheromone trails which are found to represent valuable solutions to combinatorial problems. Following the eusocial insect foraging natural strategy of past works [3,1], and the implicit adaptive ability of *Self-Regulated Swarms* and *Bacterial Foraging Optimization Algorithms* [5] our proposal also mimics ants ability to create trails by depositing and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'07, March 11-15, 2007, Seoul, Korea.

Copyright 2007 ACM 1-59593-480-4/07/0003...\$5.00.

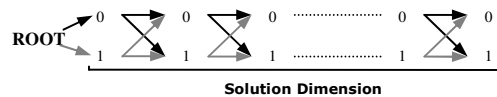


Figure 1. The BAA environment and search space.

following pheromone in the environment. Our objective is to build an algorithm suitable for the optimization of binary coded functions via stigmergy [2,1] by pheromonal communication. Having that aim in mind, our ant-like agents evolve in a binary landscape (graph in fig. 1) composed of two interconnected sequences of 0s and 1s, moving around the environment along a chosen trail, creating a solution or path (binary string) to the problem constituted by those bits found along the trail (nodes). Ants act upon the environment by depositing (*a posteriori*), on the visited connections, an amount of pheromone directly proportional to the quality of the solution represented by this binary string. Thus, pheromone laying at those higher fitness trails (best-so-far solutions) are reinforced, yielding the probability of attracting other agents in the following iterations to exploit them (*snow-ball* effect or positive feedback). The necessary counter-balanced negative feedback is given by evaporation, avoiding the system to become trapped in local optima as well as allowing it to be highly adaptive when dramatic changes occur.

2. THE BINARY ANT ALGORITHM - BAA

This environment where ants evolve is represented in fig. 1 and consists of two connected sequences of 0s and 1s. Starting from the root, the ant has two possible entries to the field. After that point, each 0 or 1 has two connections, each leading again to a 0 or a 1. These trails are unidirectional, since a pseudo-solution to the problem at hands at a given time should be represented by a finite binary string visiting those *Boolean* values at this precise left-to-right order. Any ant enters the field (left side) and stops on the other edge, creating a binary string along the way, as it passes through the nodes. The search process is described in fig. 2. The pheromone (deposited at the connections between the nodes) is

```

initialize pheromone field  $\tau_{ij} = \gamma$ 
do while stop criterion NOT TRUE
  for all N ants do
    for each bit do
      compute transition probabilities /*Eq. 1 and 2*/
      decide where to go and move to the next node
    end for
    evaluate the solution
  end for
  evaporate pheromone at all edges /*Eq. 3*/
  for all ants do
    if fitness is above average reinforce trail /*Eq. 4*/
  end for
end do

```

Figure 2. Pseudo-code for the *Binary Ant Algorithm* (BAA).

$$a_{0,1}(t) = \frac{[\tau_{0,1}(t)]^\alpha}{\sum_{i \in N_t^+} [\tau_{0,1}(t)]^\alpha} \quad (1)$$

$$p_{0,1}^k(t) = \frac{a_{0,1}(t)}{\sum_{i \in N_t^+} a_{0,1}(t)} \quad (2)$$

$$\tau_c(t) = (1 - \rho)\tau_c(t-1) \quad (3)$$

$$\tau_{0,1}^k(t) \leftarrow \tau_{0,1}^k(t) + \frac{\text{average_fitness}}{\text{fitness}(x_k)} \quad (4)$$

initialized with value γ in the beginning of the search while BAA guarantees that each node maintains at least this level of pheromone along the search process. Thus, a connection has always a chance of being chosen, maintaining a diversity for other solutions. When visiting a node (0 or 1), an agent decides which way to go by first computing the transition probabilities p – see Eqs. 1 and 2 based on ACO [2]. Then, according to the probabilities, the ant decides the next step, repeating the process until it reaches the last bit, when finally the string is found during the ant’s way through the field is evaluated as a solution to the problem. The procedure goes on until N ants ($N > 1$) have completed its journey across the field generating N solutions. The amount of pheromone at all edges or connections c is then evaporated according to Eq. 3, where ρ is the evaporation rate. Finally, the reinforcement process acts upon the environment reflecting the quality of the solutions on the pheromone levels. This is done by first choosing those ants that generated solutions with fitness equal or bellow population’s average fitness (minimization is considered here). Then, BAA revisits those trails that created the solutions and reinforces the pheromone at the connections with a value proportional to the quality of each solution – see Eq. 4. In Eq. 1, the parameter α controls the relative weight of pheromone trail in the probability computation and $\tau_{0,1}(t)$ refers to the pheromone in the connection for which the probability is being calculated, while in Eq. 4 the term $\tau_{0,1}^k(t)$ refers to a connection belonging to the trail traveled by ant k , that generated the bit string x_k . The described procedure guarantees that trails which generate better solutions receive larger amounts of pheromone, attracting insects in further iterations to swarm-around their neighborhood, increasing the probability of finding good solutions and, consequently, the global optimum. With the described mechanism working over the proposed binary environment we expect the resulting algorithm to properly follow fast moving extrema on Dynamic Optimization Problems (DOP).

3. RESULTS AND DISCUSSION

To test BAA on DOPs we choose two maximization experiments described in [6,5]. The first uses a *Royal Road* function to build an oscillatory *Royal Road*. The second test is based on a dynamic version of *Schaffer’s* function – see [6,5] for detailed descriptions, functions and experiments. We compared BAA with a *Standard Genetic Algorithm* (SGA) and a *Co-Evolutionary GA with Genotype Editing* (ABMGE) [6]. Our proposal outperformed both algorithms in the task of tracking the moving optimum for the two dynamic functions. The SGA was implemented with binary tournament and one-point crossover. Several mutation rates (p_m), crossover rates (p_c) and population size (N) values were tested. BAA was also run with several values for N , ρ and α . Parameter γ was set to 0.07 in all tests. Fig. 3 shows the BAA and SGA curves when tracking *Schaffer* function optimum (function shifts every 10000 evaluations). These captions are representative of BAA and SGA general behavior on both functions. BAA not only reaches

the optimum before the first change, but it is also able to track it down everytime the function shifts. SGAs are far from defeat BAA. Their populations consistently evolve away from the best fitness. Also, BAA shows to be more robust to severity increase (severity S measures the magnitude of these shifts [5]). BAA also outperformed the results achieved on both functions by ABMGE – see [5] for ABMGE results.

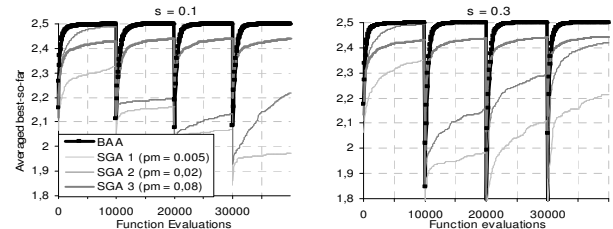


Figure 3. BAA and SGA on the dynamic *Schaffer* function. SGA: $N = 20$; $p_c = 0.9$. BAA: $N = 20$; $\alpha = 0.68$, $\rho = 0.5$, $\gamma = 0.07$.

Finally, in [4], a binary ACO (BACO) was recently presented. Among several differences, our proposal diverges from BACO in the way the search space is codified into a graph, as stressed in our introduction. While in our approach an ant-like agent faces distinct transition probabilities if it stands on a 0 or a 1 (for the same position), binary ACO has only two edges connecting each bit of the solution. Although we did not directly compare BACO and BAA (since BACO is designed to optimize stationary functions and uses a great amount of parameters needed to be tuned, as well as repair functions and local search) we did compare the basic models by implementing the mechanism shared by both algorithms and testing it over the distinct binary environments. The performance of the algorithm with a BACO-like graph was still superior to SGA. However, the convergence curves are still far from those attained by BAA. The ants’ binary environment proposed in this paper proved to be more suitable to deal with the *Royal Road* and *Schaffer* DOPs. A problem of increased complexity comparing it to stationary functions.

4. REFERENCES

- [1] Abraham, A., Grosan, C., Ramos, V. (Eds.) *Stigmergic Optimization*. Studies in Computational Intelligence, Vol. 31, Springer-Verlag, 2006.
- [2] Dorigo, M., Blum, C. *Ant Colony Optimization Theory: A Survey*. Theo. Comp. Science, 344, pp. 243-278, 2005.
- [3] Fernandes, C., Ramos, V., Rosa, A.C. *Varying the Population Size of Artificial Foraging Swarms on Time Varying Landscapes*. In Art. Neural Networks: Biological Inspirations, LNCS, Vol. 3696, pp. 311-316, 2005.
- [4] Kong, M., Tian P. *Introducing a Binary Ant Colony Optimization*. In Proc. of the 6th Int. Workshop on ACO and Swarm Intelligence, LNCS, Vol. 4150, pp. 444-451, 2006.
- [5] Ramos, V., Fernandes, C., Rosa, A.C. *On Self-Regulated Swarms, Societal Memory, Speed and Dynamics*, In Proc. ALife-X, MIT Press, pp. 393-399, 2006.
- [6] Rocha, L., Maguitman, A., Huang, C., Kaur, J., Narayanan, S., *An Evolutionary model of Genotype Editing*, In Proc. ALife-X, MIT Press, pp. 105-111, 2006.

ⁱ The first author is funded by FCT, *Ministério da Ciência e Tecnologia*, Research Fellowship SFRH/BD/18868/2004.