Binary Ant Algorithm

Carlos M. Fernandes
LaSEEB, Technical Univ. of Lisbon
Av. Rovisco Pais, 1, TN 6.21, 1049-001, Lisbon, PORTUGAL
cfernandes@laseeb.org

Agostinho C. Rosa
LaSEEB, Technical Univ. of Lisbon
Av. Rovisco Pais, 1, TN 6.21, 1049-001, Lisbon, PORTUGAL
acrosa@isr.ist.utl.pt

Vitorino Ramos
LaSEEB, Technical Univ. of Lisbon
Av. Rovisco Pais, 1, TN 6.21, 1049-001, Lisbon, PORTUGAL
vramos@laseeb.org

ABSTRACT
When facing dynamic optimization problems the goal is no longer to find the extrema, but to track their progression through the space as closely as possible. Over these kind of over changing, complex and ubiquitous real-world problems, the explorative-exploitive subtle counterbalance character of our current state-of-the-art search algorithms should be biased towards an increased explorative behavior. While counterproductive in classic problems, the main and obvious reason of using it in severe dynamic problems is simple: while we engage ourselves in exploiting the extrema, the extrema moves elsewhere. In order to tackle this subtle compromise, we propose a novel algorithm for optimization in dynamic binary landscapes, stressing the role of negative feedback mechanisms. The Binary Ant Algorithm (BAA) mimics some aspects of social insects' behavior. Like Ant Colony Optimization (ACO), BAA acts by building pheromone maps over a graph of possible trails representing pseudo-solutions of increasing quality to a specific optimization problem. Main differences rely on the way this search space is represented and provided to the colony in order to explore/exploit it, while and more important, we enrol in providing strong evaporation to the problem-habitat. By a process of pheromone reinforcement and evaporation the artificial insect's trails over the graph converge to regions near the ideal solution of the optimization problem. Over each generation, positive feedbacks made available by pheromone reinforcement consolidate the best solutions found so far, while enhanced negative feedbacks given by the evaporation mechanism provided the system with population diversity and fast self-adaptive characteristics, allowing BAA to be particularly suitable for severe complex dynamic optimization problems. Experiments made with some well known test functions frequently used in the Evolutionary Algorithms' research field illustrate the efficiency of the proposed method. BAA was also compared with other algorithms, proving to be more able to track fast moving extrema on several test problems.

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

General Terms: Algorithms, Experimentation.

Keywords: Ant algorithms, Swarm Intelligence, Stigmergy, Dynamic Optimization.

1. INTRODUCTION
Swarm Intelligence [8,12] refers to systems where unsophisticated distributed entities evolve by interacting locally with their environment or search landscape. The communication between entities via the environment causes the emergence of coherent global patterns and the formation of a social collective intelligence that easily lead to bio-inspired computational paradigms and Stigmergic Optimization [8, 12]. Ant Colony Optimization (ACO) [6], Particle Swarm Optimization (PSO) [9] and other algorithms inspired by the behavior of bees [22] or bacteria [16], for instance, are examples of self-adaptive multi-agent systems based on natural organisms.

ACO algorithms have proven to be suitable for several hard combinatorial optimization problems. 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. Together with a constant pheromone evaporation rate, these stigmergic mechanisms lead to the emergence of pheromone trails which are found to represent valuable solutions to combinatorial problems. ACO was first applied to the Traveling Salesman Problem and since then it has proved to be efficient in a large range of problems, like the quadratic assignment problem [14], vehicle routing problem [5] or scheduling [7] and timetabling problems [21]. Other Swarm Intelligent variations using discrete 3D grid representations instead of graphs as well as different local heuristics were also applied to Clustering, Data or Text Mining [17] and Image Processing and Pattern Recognition [10].

Following the eusocial insect foraging natural strategy of past works our proposal also mimics the ants' ability to create trails by depositing and following pheromone in the environment. Our objective is to build an algorithm suitable for the optimization of binary coded functions via stigmergy by pheromonal communication. Thus, the ants evolve in a binary landscape (graph in Fig. 1) composed of two interconnected sequences of 0s and 1s, moving in the environment along a chosen trail, creating a solution or path (binary string) to the problem constituted by the 0s and 1s that are found along the trail (nodes). The 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. Like so, pheromone laying in those trails that represent higher fitness solutions is, eventually attracting other ants in the following iterations. The necessary negative feedback is given by an evaporation process avoiding the system to become trapped in

local optima as well as allowing the system to be highly adaptive when dramatic changes occur. While the signal reinforcement works as a dynamic distributed memory, the evaporation allows for self-organized innovation and adaptation.

At each iteration, a certain number of ants go through this process of pheromone reinforcement and evaporation, and path-solution creation. The pheromone maps, created by the interaction between all the ants and the fitness landscape, evolve to a state where ants start to be attracted to regions near the optima of the problem. The self-adaptive and stigmergic nature of the model suggests that BAA may be an efficient strategy to deal with the problem of tracking extrema in dynamic landscapes.

In Dynamic Optimization Problems, the fitness function and the constraints of the problem are not constant [4, 1, 19]. When changes occur, the solutions already found may be no longer valuable and the process must engage in a new search effort (check [4] for enhanced analysis). Traditional Evolutionary Algorithms [2], for instance, may encounter some difficulties while solving dynamic problems, when the first convergence stage reduces population diversity, thus decreasing its capability to react to sudden changes. The crucial and delicate equilibrium needed between exploration and exploitation in static environments becomes even more important and complex when dealing with Dynamic Optimization Problems. Swarm Intelligence and Stigmergic Optimization major characteristics point toward promising research paths covering the field of optimization in dynamic environments. Previous results on different high-demanding areas like image processing between two altering images [10] or mobile wireless networking using SI [20] supports this general idea. Also, specific ACO algorithms were designed to tackle dynamic Traveling Salesman Problems [11] and dynamic real-world industrial problems [23], amongst other applications. This paper will show that the proposed swarm algorithm efficiently tracks the extrema in a set of dynamic binary test problems and outperforms the Standard Genetic Algorithm (SGA) when facing the same task. Also, results will show that BAA is more able than a co-evolutionary model of Genotype Editing (ABMGE) [20] when evolving on the dynamic landscapes of the test set.

2. ANT ALGORITHMS

Ant algorithms are one of the most successful examples of Swarm Intelligence. They have been applied to a wide set of problems, ranging from the Traveling Salesman Problem [8] to clustering problems [17]. We will briefly describe the ACO meta-heuristic which defines a particular class of ant algorithms. 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). A state transition rule is essential to guide the ants through the environment until a complete solution is built. The process of solution construction and pheromone update continues until a termination criterion is reached. Since the Traveling Salesman Problem was the first problem to be attacked by these methods we will take a closer look at the heuristics of that particular ACO algorithm.

Starting from a city (node), an ant moves from one another until all cities have been visited. When being at a node, the ant decides to go to an unvisited node with a certain probability that depends on two factors: the pheromone level of the connection and local heuristic information (the distance between the two cities). After all the ants have completed their tour, each one deposits an amount of pheromone on each connection that is used in its trail. The amount of pheromone is a function of the ant's performance since the shorter the tour, the greater the amount of pheromone deposited. After updating the pheromone levels, evaporation takes place by reducing the amount of pheromone in each connection. This simple method of indirect stigmergic communication based on the behavior of natural ant colonies proved to be effective not only in the Traveling Salesman Problem but also in a wide range of applications. Since its first use on the Traveling Salesman Problem, ACO has experienced numerous modifications in order to improve its performance or adapt itself to other types of problems - see [8] for a survey.

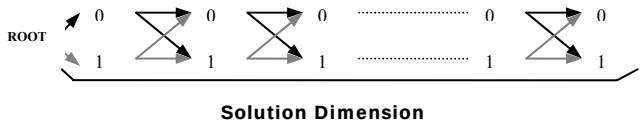


Figure 1. The *Binary Ant Algorithm* (BAA) environment and search space.

3. THE BINARY ANT ALGORITHM: BAA

In this section we describe BAA's heuristics and analyze its components and expected global behavior. This way, we look forward to identify the strengths and weaknesses of the model, an effort that not only guides further research in order to improve some of its limitations, but also allow us to determine possible areas of application. The model starts by building the environment where our ant-like agents will evolve. This environment 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.

```
initialize pheromone field τ_i,j = γ
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 of the *Binary Ant Algorithm* (BAA).

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 solution to the problem at hands at a given time should be represented by a finite binary string visiting those binary values at this precise left-to-right order, as the way a genotype is coded on traditional Genetic Algorithms. Any ant enters the field at the left side and stops on the other edge, creating a binary string as it passes trough

the nodes. Notice that the field length is equal to the solution dimension, so the binary string generated by each ant when crossing the field represents a possible solution to the problem.

With a proper environment to evolve, the ants can now start the search process, which is described in fig. 2. The pheromone is deposited at the connections (edges) between the nodes. Its level is 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 [6].

$$a_{0,1}(t) = \frac{\left[\tau_{0,1}(t)\right]^{\alpha}}{\sum_{l \in N_i^k} \left[\tau_{0,1}(t)\right]^{\alpha}} \quad (1)$$

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

Then, according to the probabilities, the ant decides the next step, repeating the process until it reaches the last bit, when finally the string 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.

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

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

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, the algorithm 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 $\tau^{k}_{0,1}(t)$ refers to a connection belonging to the trail traveled by ant k, that generated the bit string x_k that corresponds to a solution with fitness below or equal or bellow average fitness. 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.

Previous experiments made with some well known static test functions showed that BAA is able to converge to the optimal solution. The pheromone levels in the neighborhood of the global optimum grow during the search and the algorithm ends up finding the best solution. However, the process is not very fast (even so, BAA convergence speed is similar to a generational genetic algorithm in most of the functions we tested). This is not surprising since we deal with a basic version of the algorithm. Global and local search mechanisms are fundamental in any search process. In Evolutionary Algorithms, for instance, mutation guarantees global search (exploration), while crossover is usually considered to be the main responsible for local search ability of the algorithms (exploitation). Meanwhile, elitist strategies are possible and frequently used since they assure not only the maintenance of best solutions found but also the spread of their higher fitted genes across the entire population. BAA also holds mechanisms that may guarantee, even if indirectly, local and global search abilities. While evaporation acts as a kind of mutation enabler (by attenuating the differences in the pheromone levels of the two connections leaving each node), reinforcement drives the ants into the trails that already created good solutions, forcing them to exploit that regions of the search space. However, the BAA heuristic eliminates all solutions when starting a new iteration. What rests is a kind of distributed memory, that is, the remains of the collective action of the colony on the environment, which indirectly reflect the quality of those previous solutions, as well as somehow the features of the environment in itself. This anti-elitist and implicit essence of BAA slows down the convergence rate when optimizing static problems, but can be advantageous in dynamic environments. More than high speed of convergence (which may be even harmful to global performance), Dynamic Optimization Problems require population diversity and self-adaptive capabilities. When the objective function changes, Evolutionary Algorithms mutation's effort of moving away from the old optimum has weak directional impact, while the evaporation process of BAA acts upon the whole environment clearing the way for the emergence of different paths that reflect the new optimal bit string. A quick evaporation of the pheromone is essential to ant algorithms on Dynamic Optimization Problems. However, a fast reinforcement scheme is also necessary in order to reacquire the position of the moving extrema in a small number of generations. Remember that only above average solutions contribute to the reinforcement process, so it is appropriate to say that BAA emulates a kind of elitist selection. However, a strong evaporation rate and the elimination of all the solutions when starting a new iteration attenuates the selection pressure and provide the algorithm with a non-elitist general nature.

3.1 Ant Algorithms and PBIL

The Population-Based Incremental Learning algorithm (PBIL) was first proposed in [3]. PBIL combines Evolutionary Computation and competitive learning by evolving a probability vector from previously generated solutions of a binary coded problem. New solutions are then created based on those probability values. The vector is initialized with values 0.5, which means that the first population of solutions is randomly created since there is an equal probability of generating a 0 or a 1 for each locus. As the search advances, the probability values will tend to 0 or 1.0. If a highly fitted solution holds a 1 in a specific locus then the corresponding position of the vector is increased, approaching 1. Otherwise, the value is decreased and approaches 0. After certain a number of generations the vector values reflect

the genotypes of the best solutions found so far. The system eventually converges to a situation where optimum solutions have good chances of being created from the probability vectors.

In [13], the Binary Ant System was recently presented. Among several differences, our proposal diverges from Binary Ant System in the way the search space is codified into a graph. While in BAA an ant-like agent faces distinct transition probabilities if it stands on a 0 or a 1 (for the same position), Binary Ant System has only two edges connecting each bit of the solution, that is, the environment may be viewed has a transition probability vector [13] similar to the one found in PBIL. Also, Binary Ant System uses a great amount of parameters needed to be tuned, as well as repair functions and local search mechanisms. The pheromone update process is elitist and depends on the transition probabilities rather than the fitness of the solutions. In addition, the pheromone reinforcement process requires a set of five parameters. Due to its complex design and the fact that it aims at solving static problems we did not compare BAA and Binary Ant System directly. However, we did compare the basic models by implementing BAA transition and update mechanisms over the Binary Ant System environment. The results are presented in section 4.1 and show that BAA environment is more appropriate to solve the proposed dynamic problems.

There are obvious similarities between PBIL and the ant algorithms, namely with Binary Ant System since its graph is easily translated into a probability vector like the one found in PBIL. BAA, on the other hand, is based on a different concept which emphasizes the paths rather than the locus, since a decision made by an ant when standing on a bit will influence the rest of its trail, while in PBIL and Binary Ant System the solution construction is purely local and each bit is chosen without any influence from previous decisions. BAA is built in a way that ants easily engage in the exploration of new paths. Its environment allows the search mechanism to be more adaptive and reactive to sudden changes in the environment. For an exhaustive analysis of the similarities between ant algorithms and PBIL please refer to [24].

4. Test Set and Results

As already stated, the characteristics of BAA suggest that the algorithm may be effective in the task of tracking the extrema of dynamic problems. To explore this hypothesis, the algorithm was tested under two optimization experiments taken from [19].

4.1 Oscillatory Royal Road

The first test is based on a small version of the Royal Road R1 [15] where the fitness of the global optimum is 80, corresponding to a 40 bits string x with all 1's ($F(x) = \sum_{s \in S} c_s \sigma_s(x) = L_1$ where $S = \{s_1, \ldots, s_8\}$; $\sigma_s(x)$ is set as 1 if x is an instance of s and 0 otherwise, and $c_s = 10$ for all s. To build the oscillatory Royal Road [19] another function (L_0) is defined where each schema is composed of 0's resulting in an optimal bit string with all 0s (also with fitness 80). The objective function oscillates between these two strings.

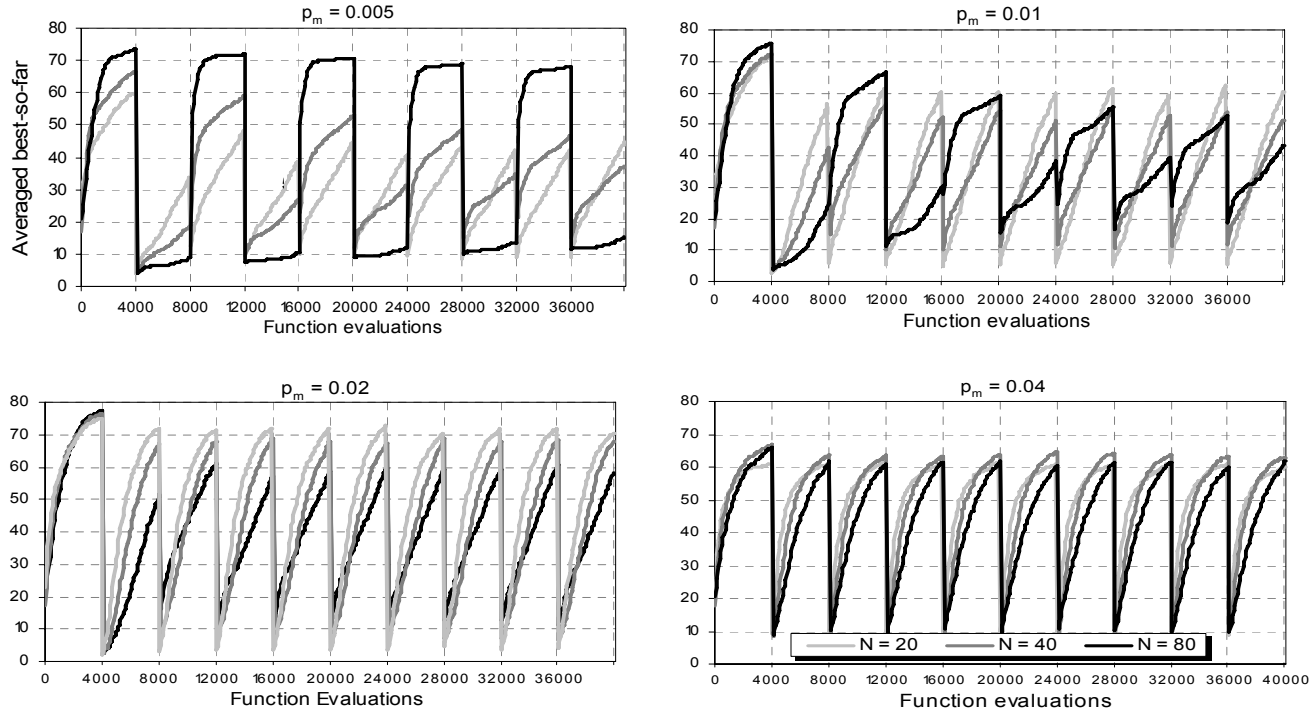


Figure 3, Averaged best-so-far performance of SGA.

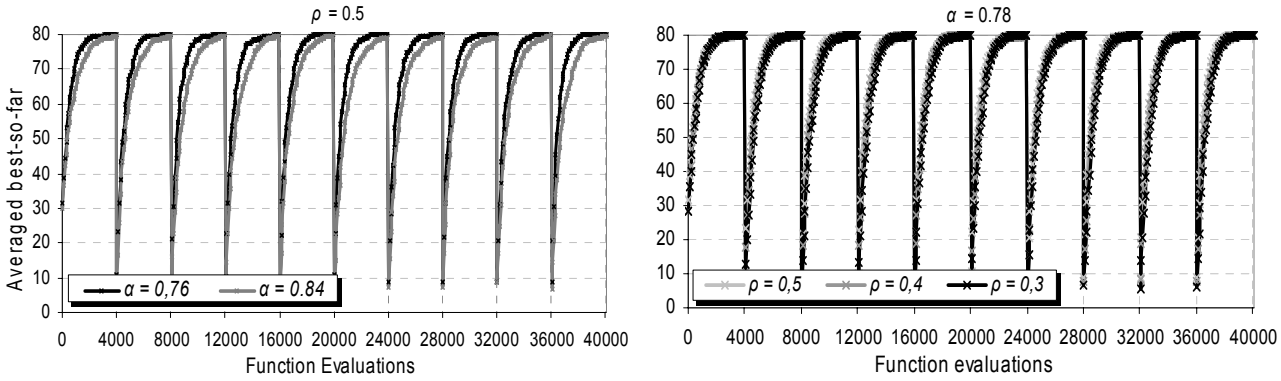


Figure 4. Averaged best-so-far performance of BAA. Parameters: = 0.07 and N = 10.

A Standard Genetic Algorithm (SGA) with generational replacement was tested with mutation rate (p_m) ranging through 0.005, 0.01, 0.02, 0.04 and 0.08, binary tournament, one-point crossover with crossover rate (p_c) equal to 0.7 and 0.9 and population size (N) ranging through 20, 40 and 80. Uniform crossover was also tested without significant improvement in the performance of the algorithm. By testing a wide range of parameter values we avoid comparing BAA with a sub-optimal configuration of a SGA. BAA was tested with populations of 5, 10, 20 and 40 ants. Evaporation rate (ρ) ranged through 0.3, 0.4 and 0.5 (meaning that 1- ρ in Eq. 3 is equal to 0.7, 0.6 and 0.5, respectively). Parameter α was set to diverse values in the range [0.5, 1.0] and parameter γ was set to 0.07 in all tests.

The graphics in figures 3 and 4 show the evolution of the best solution found so far by the algorithms. (Notice that when the objective function changes, the best-so-far solution must be reevaluated according to the new landscape.) For SGA, the results over the complete range of p_m are presented in fig. 3 (p_c = 0.9). A proper following of the extrema should generate a best-so-far curve that not only reaches the global optimum of the first environment, but also on later stages, producing a plot with a periodic appearance, as in fig. 3. SGA with p_m = 0.005 and N = 80, for instance, clearly fails in both objectives, producing a curve that resembles a rectangular wave. When mutation increases, the curves tend to be more stable and similar to the first search stage (between 0 and 4000 evaluations) which means that the algorithm is maintaining an higher genetic diversity, an essential feature to properly deal with dynamic problems. However, the SGA runs are far from reaching the global optimum of L_0 and L_1. BAA, on the other hand, revealed to be capable of performing the task of tracking the extrema of oscillatory Royal Road with several combinations of parameter values but, in general, performed better with α ranging from 0.65 to 0.85 and revealed a higher sensitivity to this parameter. Fig. 4 illustrates the previous statements. Notice how the performance degrades when increasing α from 0.76 to 0.84 while maintaining the other parameters fixed – left graphic of fig. 4. Changing the evaporation rate has much less impact in BAA performance – graphic on the right. In general, when comparing the algorithms' performance, it is evident that BAA is much more efficient, since the correspondent curves are in all stages almost replicas of the best-so-far fitness growth when searching the optimum of the first environment. BAA's ability to track the solutions may be clarified by the graphic of fig. 5 where it is shown the evolution of the average value of 1→1 and 0→0 transition probabilities. The 1→1 connections are the optimal local connections to attain the global optimum of environment L_1 while 0→0 are the optimal paths of L_0. Notice how the 1→1 transition probabilities grow in the periods where algorithm's task is to track L_1 optimum. When the environment changes, the 1→1 average probability decrease dramatically, and the ants will tend to choose 0→0 connections that at same time increased its transitions probabilities considerably.

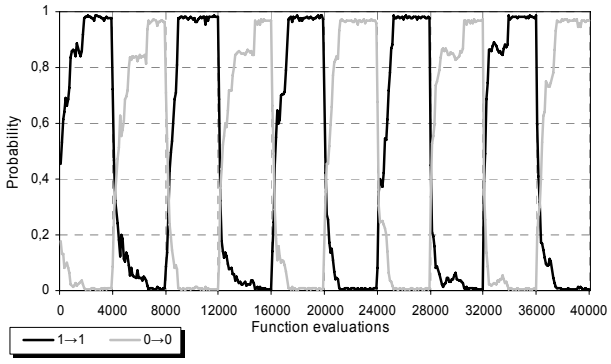
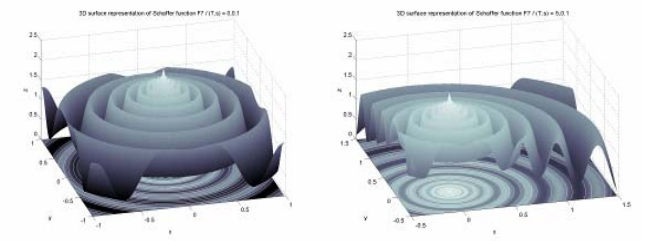


Figure 5. Evolution of the transition probabilities of 1 1 and 0 0 connections.

4.2 Dynamic Schaffer's Function

The second experiment was also taken from [19]. The test is a dynamic version of the *Schaffer's* function where the landscape is shifted every 10000 evaluations with linear dynamics' severity S - see [4, 1, 19, 18] for detailed descriptions, functions and experiments. Sketches of this multimodal function for different parametric values (different instances in time) are illustrated in fig. 6. Being s a parameter to set the severity and $X_i = x_i + \delta(t)$ (with $-1 \leq x_i \leq 1$ for i = 1, 2 and $\delta(t)$ described by Eq. 6), a possible test problem [19] can be described by Equation 5. In here, X and Y axis represent the index of the sample points in parameters x_1 and x_2 that are used to compute $f(x)$, which is then plotted on the z axis, being our aim to maximize it. The example uses linear dynamics with severity s. (This means that the extrema moves always in the same direction. Other modes of severity of



δ=0 (e.g. s=0.1,T=0 or s=1,T=0) δ=0.5 (e.g. s=0.1,T=5)

Figure 6. Sketches from the *Schaffer F7* complex multimodal function seen for different values of *T* related to the severity tests (here, *s*=0.1).

$$f\left(\vec{X}\right) = 2.5 - \left(X_1^2 + X_2^2\right)^{0.25}\left[\sin^2\left(50.\left(X_1^2 + X_2^2\right)^{0.1}\right) + 1\right] \quad (5)$$

$$\delta(0) = 0,$$
$$\delta(T) = \delta(T-1) + s \quad (6)$$

changes are possible, like circular or random. See Angeline [1] for a study and definition of dynamic test problems parameters.)

Each run on the algorithms evaluated 40000 individuals, which corresponds to four different environments. The two variables were encoded by 50 bits, so the solutions are binary strings of length 100. The results are averaged over 100 runs. These settings follow the experiences made in [19,18].

SGA parameter settings are the same as in the oscillatory Royal Road tests, except for N, that ranged through 20, 50 and 100. BAA was run with N equal to 10 and 50, γ = 0.07, ρ = 0.5 and several α values in the range [0.6, 0.8]. Fig. 7 shows the curves obtained by BAA and SGA with different mutation rate. This set of configurations is representative of SGA general behavior on dynamic Schaffer function. Notice that SGA only achieves a stable performance through the four optimization stages with high mutation rate (p_m = 0.08). However, the performance is poor. The population remains away from the optimum even when optimizing the first environment. On the other hand, with p_m = 0.02, the SGA does well on the first stage but degrades its performance significantly when the environment changes. BAA not only follows properly the optimum in the first environment, but also maintains the sameefficiency when tracking the extrema when the function changes. When comparing the graphics of fig. 6 it is clear BAA doesn't perform differently when tracking the extrema of Schaffer with s = 0.1 and s = 0.3. Table 1 shows the average value of the best-so-far fitness error (difference between the fitness of the best-so-far solution and the optimal solution). The results confirm that BAA performs better on both tests. Notice that SGA 2 improves its performance when facing s = 0.3, but the error value is still far from that attained by BAA.

Table 1. Best-so-far error values attained by the algorithms on the dynamic Schaffer's function. Results averaged over 100 runs.

	BAA	SGA 1	SGA 2	SGA 3
s = 0.1	0,023	0,389	0,288	0,091
s = 0.3	0,025	0,397	0,217	0,091

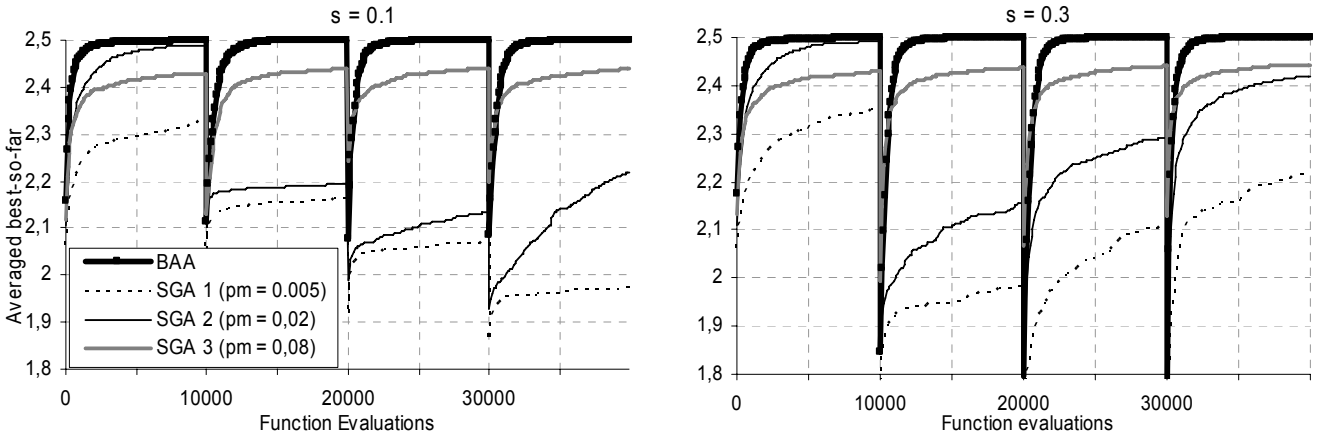


Figure 7. BAA and SGA on the dynamic *Schaffer* function. SGA: N = 20; p_c = 0.9. BAA: N = 20; □ = 0.68, □ = 0.5, □ = 0.07.

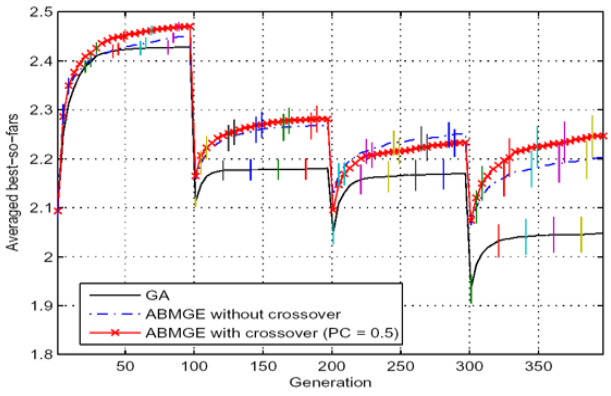


Figure 8. Results from [19] for Dynamic Schaffer with $S = 0.1$. ABMGE and GA with $N = 100$ and $p_m = 0.01$

4.1 Comparing BAA with ABMGE and the Binary Ant System

The graphic of fig. 8 was taken from [19] and show the performance of ABMGE [19] on dynamic Schaffer with severity 0.1. Although the algorithm appears to outperform the SGA, it is clear that BAA is more able to track the optima on both experiments - compare figures 7 and 8. Please notice the algorithms have populations of 100 individuals which means that the functions change at 10000 as in our experiments, since each generation evaluates 100 solutions.

Although we did not directly compare Binary Ant System and BAA (as already stated, Binary Ant System 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 two ways of translating the search space into a graph. For that purpose we built BAA's pheromone trail initialization, solution construction and pheromone update over a Binary Ant System-like graph (or probability vector), and run the resulting algorithm on the Dynamic Schaffer function. The performance of this *probability vector* BAA (v-BAA) is still clearly superior to SGA. However, when comparing BAA and v-BAA we conclude that the first model is more able to track the extrema of Dynamic Schaffer function. Also, v-BAA is more sensitive to parameters α than BAA. Some results are shown in fig. 9.

5. CONCLUSIONS

Our model was tested with success on the proposed Dynamic Optimization Problems: the Binary Ant Algorithm (BAA) efficiently tracks the extrema of oscillatory Royal Road and Dynamic Schaffer's function. Increasing the severity of changes in dynamic Schaffer posed no extra problems to BAA. In general, BAA clearly outperformed Standard Genetic Algorithm (SGA) in the experiments made for the current paper. The best-so-far curve shapes attained by the best configurations of BAA don't change throughout the different stages of the search (when the function changes). The plots resemble periodic functions and converge to the optimal neighborhood in all stages of the search.

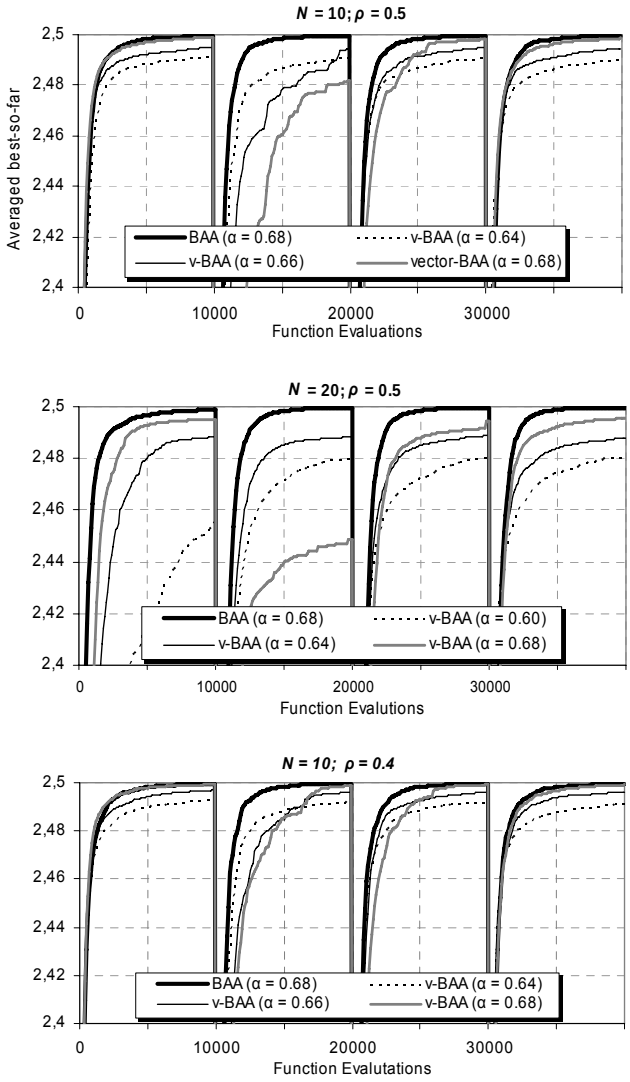


Figure 9. Comparing BAA with original graph and Binary Ant System-like graph (v-BAA) on the dynamic Schaffer's function with s = 0.3.

When comparing our results with those found in [19], we can conclude that BAA is more able to follow the optima of the dynamic test functions than ABMGE. Also, BAA's approach to the binary graph where ants evolve revealed to be more suitable to deal with the experiments than the Binary Ant System [13] environment. The good performance of our algorithm on dynamic environments may be explained by the fast variation of pheromone levels, which provides the algorithm with self-adaptive capabilities to follow the moving optimum. As the results obtained when comparing BAA with a probability vector BAA suggest, the graph may be also playing an important role in the algorithm's behavior.

The way parameters ρ, γ, α and N affect BAA's performance is still a matter for further research, but these preliminary experiments suggest that α may influence significantly the algorithms performance. Whether or not the optimal values of α

change drastically with different problems or the optimal performance is more dependent on the combination of all parameter values are also questions that need to be answered. Harder tests are also needed in order to perceive when BAA looses the ability to track the optimum in all stages as it does in the first one. Increasing severity of changes and changing update frequency may show the limits of the model. Comparing BAA to some variations of Evolutionary Algorithms usually applied to Dynamic Optimization Problems [4], like the Random Immigrants [4], is also needed and some preliminary results are encouraging. Finally, we also intend to investigate the possibility of optimizing static and noisy functions with BAA. As already stated, BAA is still very slow in some test functions. Some modifications in the algorithm were already suggested in section 3 that might improve BAA's capability of finding the optimum of static problems.

ACKNOWLEDGMENTS

The first author wishes to thank FCT, *Ministério da Ciência e Tecnologia*, his Research Fellowship SFRH/BD/18868/2004, also partially supported by Fundação para a Ciência e a Tecnologia (ISR/IST plurianual funding) through the POS_Conhecimento Program that includes FEDER funds..

References

[1] Angeline P., *Tracking Extrema in Dynamic Environments*, Proc. of the 6th Int. Conf. on Evolutionary Programming, LNCS, Springer, 1213: 335-345, 1997.

[2] Back, T., *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, 1996.

[3] Baluja, S., *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*, Technical Report CMU-CS-94-163, Carnegie Mellon 0University, USA, 1994.

[4] Branke, J*., Evolutionary Optimization in Dynamic Environm*ents. Kluwer Academic Publishers, 2002.

[5] Bullheimer B., *Ant Colony Optimization in Vehicle Routing*, PhD Thesis, University of Vienna, 1999.

[6] A. Colorni, M. Dorigo and V. Maniezzo, *Distributed Optimization by Ant Colonies*, In Proceedings of the 1st European Conference on Artificial Life, F.J. Varela and P. Bourgine (Eds.), MIT Press, Cambridge, MA, 134-142, 1992.

[7] Colorni, A., Dorigo, M., Maniezzo, V., Trubian, M., *Ant System for Job-shop Scheduling*, Belgian Journal of Operations Research, Statistics and Computer Science, 34(1): 39-53, 1994.

[8] Dorigo M., Blum C., *Ant Colony Optimization theory: A Survey*, Theoretical Computer Science, 344: 243-278, 2005.

[9] Eberhart R. C., Kennedy J., *A new optimizer using particle swarm theory*, In Proceedings of the 6th International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995.

[10] Fernandes C., Ramos V., Rosa A.C., *Self-Regulated Artificial Ant Colonies on Digital Image Habitats*, International Journal of Lateral Computing, 2(1): 1-8, 2005.

[11] Guntsch M., Middendorf M., *Applying Population Based ACO to Dynamic Optimization Problems*, Proceedings of the 3rd International Workshop ANTS2002, LNCS 2463: 111-122, 2002.

[12] Kennedy J. Eberhart R.C., Russel C. and Shi, Y., *Swarm Intelligence*, Academic Press, Morgan Kaufmann Publ., San Diego, London, 2001.

[13] Kong, M., Tian P. *Introducing a Binary Ant Colony Optimization*, In Proceedings of the 6th International Workshop on ACO and Swarm Intelligence, LNCS, 4150: 444-451, 2006.

[14] Maniezzo V., Colorni A., *The Ant System applied to the Quadratic Assignment problem*, IEEE Transactions on Knowledge and Data Engineering, 11(5): 769-778, 1999.

[15] Mitchell M., Holland J., Forrest S., *When will a GA outperform Hillclimbing?,* Advances in Neural Information Processing Systems, 6: 51-58, 1994.

[16] Passino, K.M., *Biomimicry of Bacterial Foraging for Distributed Optimization and Control*, IEEE Control Systems Magazine, 52-67, 2002.

[17] Ramos V., Merelo J.J., "Self-Organized Stigmergic Document Maps: Environment as a Mechanism for Context Learning", in E. Alba, F. Herrera, J.J. Merelo et al. (Eds.), AEB'2002,– 1st Spanish Conf. on Evolutionary and Bio-Inspired Algorithms, pp. 284-293, 2002.

[18] Ramos, V., Fernandes, C., Rosa, A.C., *On Self-Regulated Swarms, Societal Memory, Speed and Dynamics*, in L.M. Rocha, L.S. Yaeger, M.A. Bedau, D. Floreano, R.L. Goldstone and A. Vespignani (Eds.), Proc. of ALifeX, MIT Press, pp. 393-399, 2006.

[19] Rocha L., Maguitman A., Huang C., Kaur J., and Narayanan S., *An Evolutionary Model of Genotype Editing*, In Proceedings of ALifeX, MIT Press, pp. 105-111, 2006.

[20] Roth M., Wicker S., *Asymptotic Pheromone Behavior in Swarm Intelligent MANETs: An Analytical analysis of Routing Behavior*, Sixth IFIP IEEE International Conference on Mobile and Wireless Communications Network, 2004.

[21] Socha K., Sampels M., Manfrin M., *Ant Algorithms for the University Course Timetabling problem with regard to the state-of-the-art*, In Proceedings of the EvoWorkshops 2003, LNCS, Berlin Springer, 334, 345, 2003.

[22] Wedde H., Farooq M., Zhang Y., *BeeHive: An Efficient Fault Tolerance Routing Algorithm under High Loads Inspired by Honey Bee Behavior*, In Proceedings of the 4th International Workshop on Ant Colony and Swarm Intelligence (ANTS 2004), LNCS, 83-94, 2004.

[23] Xiao J., Li J., Xu Q., Huang W., Lou H., *ACS-based Dynamic Optimization for Curing of Polymeric Coating*, in AIChE Journal, 52(4): 1410-1422, 2005.

[24] Zlochin M., Birattari M., Meuleau N., Dorigo M., *Model-based search for combinatorial optimization: A critical survey*, in Annals of Operations Research, 131:373--395, 2000.