# Probabilistic Evolution and the Busy Beaver Problem

**Roberto Santana**
**Alberto Ochoa-Rodriguez**
**Marta Soto**
Center of Mathematics
and Theoretical Physics
ICIMAF. CP 10400. Habana. Cuba
{rsantana,ochoa,mrosa}@cidet.icmf.inf.cu

**Francisco B. Pereira**
**Penousal Machado**
CISUC
{xico,machado}@dei.uc.pt

**Ernesto Costa**
**Amilcar Cardoso**
Centre for Informatics and Systems
University of Coimbra. CISUC.
Polo II - Pinhal de Marrocos
3030 Coimbra, Portugal
{ernesto,amilcar}@dei.uc.pt

## Abstract

We discuss the use of probabilistic evolution in an important class of problems based on Turing Machines, namely the famous Busy Beaver. Despite the bad properties of this problem for a probabilistic solution: non-binary representation and variable associations with a strongly connected graph-like structure, our algorithm seems to outperform previous evolutionary computation approaches.

## 1 INTRODUCTION

In this paper we use Probabilistic Evolution, which means Evolutionary Computation plus the use of probability distributions, to approach the Busy Beaver problem [1]. Our contribution is a preliminary step toward a better understanding of the adequacy of Estimation Distribution Algorithms (EDA) [2], [3], [4] for problems with a graph like structure and non-binary representation. EDA proved to be very successful for many binary functions with tree-like structures, where Genetic Algorithms faced a lot of troubles. We believe that it is very important to investigate non-binary problems with graph-like structures within this probabilistic framework.

We present an evolutionary algorithm that uses probability distributions for the solution of a problem with a non-binary representation. We analyze some difficulties that arise in the design of EDA for this application domain: the Busy Beaver (BB) problem [1]. The problem can be defined as follows, suppose a Turing Machine (TM) with a two way infinite tape and a tape alphabet = {blank, **1**}, the goal is to find the N-state halting TM that writes the maximum number of **1**s when starting on a blank tape. This number, which is function of the number of states, is denoted by $\Sigma(N)$. A machine that produces $\Sigma(N)$ non-blank cells is called a BB.

Although simple in their representation, Turing machines can exhibit a very complex behavior. These characteristics make them a suitable application domain for the study of the performance of EA's in the optimization of difficult fitness landscapes. BB is an interesting problem for the identification of those features that can be considered as dominion of difficulty for EAs, and for the conception of more efficient alternatives to cope with these challenges.

Some of the features that make the BB problem challenging for evolutionary optimization are:

1. graph-like structure of the associations among the variables that encode the TMs.
2. non-binary representation.
3. the existence of possible interactions among any subset of variables during the simulation.
4. expensive simulation of the TM.

In this contribution we pay attention to the first 3 features, and leave the analysis of the cost of simulation to a forthcoming paper.

The paper is organized as follows: In section 2 we shortly introduce the class of evolutionary algorithms that use distributions. In section 3, the BB problem is formally presented. Previous evolutionary approaches to the BB are reviewed in section 4. Section 5 presents our current probabilistic approach to the problem. An evolutionary algorithm designed following the developed ideas is presented in section 6 and afterwards some seeding mechanisms that improve its performance are discussed. Finally section 7 presents our experimental results.

Table 1: EDA.

| | |
|---|---|
| **step 0:** | $t \leftarrow 1$. Generate N points randomly. |
| **step 1:** | Get a selected set S with M points. $(M < N)$. Estimate $p^s(x,t)$. |
| **step 2:** | Generate N points according to: $p(x,t+1) = p^s(x,t)$ |
| **step 3:** | $t \leftarrow t+1$. If the termination criteria are not met, goto to **step 1**. |

## 2 ESTIMATION DISTRIBUTION ALGORITHMS

Generally, in an EDA (see table 1) the estimation of the probability distribution of the best individuals is used to sample the points of the next generation, there are no mutation nor crossover operators. The term EDA could be also used to group evolutionary computation methods (like GP, GA, etc.) that have in common the use of distribution estimation to model promising solutions and guide the further search. EDA can be classified by considering the complexity of the models used to capture the interdependencies between the variables, simple models consider a lesser number of dependencies among the variables.

Other classification can be achieved by grouping EDA according to the way learning is done in the probability graphical model used [7]. A first class covers the algorithms that make a parametric learning of the probabilities, the second is composed by those algorithms where a structural learning of the model is done. Research on EDA has been mainly focused on binary string representation. Binary problems allow the application of simple and less costly statistical techniques. In [8] is presented a probabilistic evolutionary algorithm not based on binary representation.

The computational cost of an EDA implementation is determined by the memory needed to store, and the time spent to update and sample the probabilistic model. This time is often exponential in the maximum number of variables that interact in the problem, or which is the same, the size of the building blocks.

## 3 PROBLEM DEFINITION

A deterministic TM can be specified by a sextuple

$$(Q, \Pi, \Gamma, \delta, s, f)$$

where [9]: Q is a finite set of states, $\Pi$ and $\Gamma$ are alphabets of input and tape symbols, $\delta$ is the transition function, whereas $\{s, f\}$ denotes the start and final states.

The original definition of the BB problem by Rado [1], considered deterministic 5-tuple TMs with N+1 states (N states and an anonymous halting state). In each transition, the machine writes a symbol to the tape and moves its head either left or right, i.e., the transition function has the following format:

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

where L denotes move left and R move right. A common variation consists in considering 4-tuple TMs, where the transition function has the following format:

$$\delta : Q \times \Gamma \rightarrow Q \times \{\Gamma \cup \{L, R\}\}$$

i.e., a 4-tuple TM either writes a new symbol on the tape or moves its head before entering the new state.

The productivity of a deterministic 5-tuple TM can be defined as the number of ones present on the (initially blank) tape when the machine halts. Machines that do not halt have productivity zero. The function $\Sigma(N)$ is defined to be the maximum productivity that can be achieved by a N-state TM. This TM is what is called a Busy Beaver [1].

In the 4-tuple variant, productivity is usually defined as the length of the sequence of ones produced by a TM when started on a blank tape, and halting when scanning the leftmost one of the string, with the rest of the tape blank. Machines that do not halt or do not halt in this configuration have productivity zero [10].

We will denote a BB with N states as BB(N), 4-tuple and 5-tuple BB are respectively denoted as BB-4 and BB-5.

## 4 PREVIOUS RESEARCH

Early works on Evolutionary Computation [5] considered Finite State machines as an appropriate research domain for the design and validation of more sophisticated evolutionary techniques. A TM is essentially a finite-state sequential machine that has the ability to communicate with an external store of information [6].

Previous research on the application of Evolutionary techniques to the BB problem includes the work of Terry Jones regarding the use of GA for attacking the BB-5 problem. Although in [11] it is stated that their results do not necessary imply that every GA would be worse than a hillclimber on this problem, they support evidence that hillclimber is finding peaks ( for the BB-5, N=4) four and a half times as fast as the GA. Their results are achieved using a character string representation with two point crossover, tournament selection of size 2 and a mutation rate of 0.01.

Pereira et al. [12] propose a GA for the BB-4 problem. This GA uses a string based, two point crossover, simple point mutation, probabilistic selection and elitism. Additionally a hillclimbing procedure is incorporated. Experiments for BB(6), BB(7) and BB(8) were conducted. For BB(7) a new contender (productivity equal 102) was found (previous contender had a productivity of 37) GA proved to be an effective way for minimizing the number of TMs inspected.

Alternative ways of codifying and interpreting the TMs have been also tried [13]. There are several TMs that exhibit the same behavior, these machines can be considered equivalent and can be grouped in equivalence classes. The most important among the known equivalent classes for TMs is the Tree Normal Form (TNF) [14]. Experimental results show [13] that TNF representation provides important improvement over the standard genetic codification for the Genetic Algorithms used. Recent applications of GA to the BB-4 problems [15] have shown that $\Sigma(6) \geq 25$, and $\Sigma(7) \geq 164$.

These results were achieved using TNF representation, a graph based crossover operator, single point mutation and Tournament selection. A detailed description of the GA used can be found in [15].

## 5 OUR APPROACH

### 5.1 REPRESENTATION

In our representation for the BB(N) each potential solution is an integer vector with 4*N variables . Each state in the TM is expressed in 4 variables which represent:

1. The new state to be visited when a blank is read.
2. The action performed when a blank is read (write a blank, write a 1, move left, move right).
3. The new state to be visited when a 1 is read.
4. The action performed when a 1 is read.

Variables that represent transitions to the new state can have N+1 different values (N states plus the halting state). The values for the remaining variables range between 0 and 3, corresponding to the 4 actions that can be performed.

Additionally, we set a constraint in our representation to reduce the dimension of the space of solutions. We will allow each individual to have only one transition to the halting state, this transition can be allocated only in variables that represent transitions when a 1 is read. Our operators guarantee that all machines satisfy this constraint.

Figure 1 shows a population of 5 TMs, all have productivity equal or higher than 17. Column 2i-1 stores the action and transition done by the state i when a blank is read. Column 2i represents similar information but when a 1 is read in state i. The upper row in the machines stores variables representing transitions to the new states; the lower row, variables representing actions. Note that each individual has only one transition to the final state (F).

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 6 | 5 | 1 | 4 | 2 | 2 | 4 | 6 | F | 1 | 3 |
| R | R | 1 | R | 1 | R | 1 | L | 1 | 1 | R | R |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 5 | 6 | 2 | 2 | F | 5 | 6 | 3 | 4 | 4 | 1 |
| 1 | L | R | R | R | 1 | R | L | R | 0 | 1 | L |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | F | 5 | 4 | 1 | 6 | 1 | 3 | 4 | 5 | 3 | 2 |
| 1 | 1 | 1 | R | R | R | 1 | R | 1 | L | R | R |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | F | 1 | 6 | 6 | 4 | 5 | 2 | 2 | 5 | 1 | 3 |
| 1 | 1 | 1 | L | L | L | 1 | L | 1 | R | L | L |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | F | 4 | 5 | 4 | 2 | 6 | 4 | 6 | 6 | 1 | 3 |
| 1 | 1 | R | R | 1 | R | 1 | L | 1 | R | 1 | R |

Figure 1: Population of five possible solutions with a high fitness evaluation. The Blank symbol is represented by 0.

This representation faces what has been called permutation or competing convention problem [16]. It is caused by the many-to-one mapping from the representation to the actual TM, since two TMs that order their states differently in their chromosomes will still have equivalent functionally.

The dilemma existing between the computational cost and the expressiveness of the probability model is evident in the BB problem. In principle, during the simulation of the TM, there can arise interactions among any subset of the variables. This fact indicates that the building blocks for the BB problem are not short, implying that a probability model able to express its structure has to cope with higher order interactions among the variables.

Unfortunately, the computational cost of an EDA (section 2) is often exponential in the size of the building blocks. For example, the simplest probabilistic model for the BB(N), which is based on the assump-

Table 2: Simplified Bivariate model

| Cf. | Fq. | Cf. | Fq. | Cf. | Fq. | Cf. | Fq. |
|---|---|---|---|---|---|---|---|
| 00 | 0 | 10 | 0 | L0 | 0 | R0 | 1 |
| 01 | 0 | 11 | 0 | L1 | 0 | R1 | 0 |
| 0L | 0 | 1L | 7 | LL | 2 | RL | 1 |
| 0R | 0 | 1R | 8 | LR | 0 | RR | 6 |

tion that the states of the machine are independent, requires a table with $N \cdot [4 \cdot (N+1)]^2$ entries to be stored. Moreover, if we consider a model that stores all first order interactions we obtain an amount of $N \cdot \frac{(N-1)}{2}[4 \cdot (N+1)]^4$. The corresponding values of this two models for the BB(6) are 4704 and 9219840 respectively.

There is a strong connection between the representation and the choice of the probabilistic model. However, for a given representation several models can be chosen. In the next subsection we explore one of the possible models.

## 5.2 THE ACTION MODEL

It is clear that the 4*N variables that represent a TM can interact during the simulation. Nevertheless there are differences in the role played by the set of variables representing states and variables that represent actions. BBs with a high fitness are characterized by a highly interconnected structure (determined in our coding by variables that represent states). On the other hand BBs exhibit a non uniform distribution of the values for variables representing actions.

The approach introduced in this paper considers a model that uses probabilistic information of variables representing actions. Thus, just 2*N variables are considered in the probabilistic model. Other kind of dependencies that arise during the evolution, are respected in some way by our sampling algorithm that applies by partially modifying the existing solutions.

Our probabilistic model groups variables representing actions, trying to capture the dependencies existing between them in the selected set. Variables can be grouped in different ways, for example we can define a bivariate model by keeping the N tables of bivariate marginals corresponding to the pair of actions in each state. We find this choice useful in a representation for which the order of states is relevant, however in our case the permutation problem determines that equivalent solutions can have a different ordering of their states. We make a simplification of the previous model by having just one table of 16 entries that keeps the frequencies of all possible states' actions regardless their position in the chromosome.

Table 2 shows the simplified bivariate model corresponding to machines in figure 1. A table with 16 entries is enough to describe the model. In the table we present the frequencies (Fq.) for each possible configuration (Cf.) of the pairs of actions variables. Each pair is composed by actions that belong to the same state. Of the 12 variables in the lowest row of each machine, only 10 are considered for the probabilistic model (the state that includes transitions to the halting state is not computed).

It is evident that the way states are connected in the TM is not independent of the actions taken, and that the simplification implicit in the model can affect the quality of the search. One strong argument supporting the simplification is the fact that the probabilistic representation of the topology of the TM is hard, due to the permutation problem. Besides, these variables are precisely the ones with a highest number of configurations, by excluding them from the model we drastically reduce the memory requirements.

Evidence from a statistical analysis of good solutions supports the idea that the use of probabilistic information of the transition values can bias the search to promising regions of the solution space. By measuring the frequency of the 4 different transitions in good solutions it is clear that exists an unequal distribution of these values. This can be expected from the fact that BBs must have for example more states that write ones that those writing blanks. The bivariate model can store these differences in the frequencies of single transitions. Additionally, it can capture the dependencies between the actions the TMs do when they read a blank and the action done by the states when reading a 1.

Two extensions of the action model that consider respectively less and more interactions were tested, but best results were achieved with the bivariate model. In this paper we will present only results with the bivariate action model. The other two variants were:

Univariate Model - The state transitions are also considered but variables representing transitions where a blank is read are considered independently of variables representing transitions where a 1 is read. Instead of tables of 16 entries we keep the whole model in two tables, each of 4 entries.

Trivariate Model - More dependencies are represented. We measure the frequency of subsets of 3 variables. For each state, the two variables that represent actions when respectively reading a blank and a 1 are considered in separation. Each action in the state is linked to

Table 3: POSATM.

| | |
|---|---|
| **step 0:** | $t \leftarrow 1$. Generate T points randomly or using a seeding procedure. |
| **step 1:** | Get a selected set S with M points. $(M < T)$. Estimate the model $p^s_M(\widehat{x}, t)$. |
| **step 2:** | Generate T points making a partial modification of the selected points using $p^s_M$ to update $\mathrm{V}_a$ variables. |
| **step 3:** | mutate variables |
| **step 4:** | $t \leftarrow t + 1$. If the termination criteria are not met, goto to **step 1**. |

the actions of the new state their corresponding transition variable points out. Two tables of 64 entries each keep the statistics of the population.

## 6 THE POSATM

Having in mind the ideas presented above, we have developed a simple algorithm (see table 3). We have called it Probabilistic Oriented Search Algorithm for Turing Machines (POSATM). Besides the use of a probabilistic model that does not consider all the variables (note that we use $\widehat{x}$ instead of $x$ ), the POSATM has other characteristic that distinguishes it from classical EDA.

Perhaps the most important difference is the way new solutions are generated (step 2). This process should guarantee a proper mixing of building blocks. However, according to our experience, it is very difficult to form good solutions by just joining sub-solutions of shorter size. This situation seems to remain the same for problems defined on TMs, as far as they are strong connected. Our alternative to this situation is to modify the current candidate using the probabilistic distribution. For the bivariate model we select V variables that represent transitions used during the simulations and alter them. Modification is done using the table with the bivariate marginal distributions. Let $a_1$ be the value of variable $x_1$ to be modified, $a_1$ is changed to $a_2$ with probability $p\,(x_1 = a_2 \mid x_2 = c)$ where $x_2$ is the other variable that defines the states configuration and $c$ is its current value. This process is repeated for the V variables.

Figure 2 shows an example of how step 2 of POSATM is applied. Figure 2a) presents the actions variables of the first TM in figure 1. The last variable is selected to be modified, its current value is $a_1 = R$. Following the bivariate model represented in Table 2 only two changes are possible, these are shown in figure 2b) and 2c). In Table 2 it can be seen that these two changes have the same probability.

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a) | R | R | 1 | R | 1 | R | 1 | L | 1 | 1 | *R* | *R* |
| b) | R | R | 1 | R | 1 | R | 1 | L | 1 | 1 | *R* | 0 |
| c) | R | R | 1 | R | 1 | R | 1 | L | 1 | 1 | *R* | *L* |

Figure 2: An initial solution and two alternative outputs for the application of step 2 of POSATM.

Best results were achieved with the parameter V set to 1. In this way we get a new point that does not radically differ from the previous one, as is the case in Hillclimbing algorithms. Nevertheless, the change is determined by the probabilistic model of the selected solutions. The number of variables that are changed is determined by the parameter V.

As far as the variables representing the topology are not considered for the model, they do not change in this step of the algorithm.

In opposition to classical EDA a mutation step is introduced in POSATM that allows to change those variables representing the connections to new states. Also transitions can be modified. Although in the case of transitions variables the mutation effect can be achieved by perturbing the probabilistic model, we separate these two steps to make easier the tuning of the algorithm.

In terms of the building blocks theory the above modifications can be seen as alternatives to cope with problems where there are interactions between many variables. We avoid the disruption of higher order building blocks by partially modifying the solutions and achieve the supply of new building blocks by direct infusion of them through the mutation step.

Another important problem that has been addressed in POSATM is the initial supply of building blocks. Individuals of the initial population are not randomly generated, instead heuristic methods that produce better although suboptimal individuals are applied to create them. Seeding has shown to be useful in GA for problems defined on graphs [17]. More recently seeding has been used in the framework of EDA by generating the initial population using information about the problem represented in a probabilistic scheme.

Several seeding mechanisms are possible. We have explored two of them. In the first approach we filled transition values from an initial empirical distribution. The topology of the TMs is built in two ways: randomly, and trying to connect all the states through a cycle. In the second approach we seed subsolutions found from the analysis of the set of TMs that stop for BB(3) problem using a TNF representation. Although in all the experiments with POSATM reported in the next section we seed the initial population, we let the discussion on this issue to a forthcoming paper.

## 7 EXPERIMENTS

In our experiments we will use POSATM to seek for the 4-tuple 6 state BB. Due to the existence of the Halting Problem it is necessary to set a limit for the number of transitions. Machines that do not halt before this limit are considered non-halting TMs.

An initial set of experiments was oriented to compare the behavior of our approach with the results presented in [15] obtained with a genetic algorithm with two point and graph based crossover (GBC). The main idea of GBC is the exchange of subgraphs between the individuals. Behind its conception is the assumption that subgraphs or submachines are the building blocks of the problem. In the 6 state BB, best results obtained with GBC [15] have been achieved using a maximum graph crossover size (maximum number of exchanged subgraphs) equal to 3.

The first experiments concern how efficient is the search for the 4-tuple BB(6). We set the maximum number of allowed transitions to 250. This was the same limit imposed in [15] and was motivated by the fact that the previously best known candidate wrote 21 1s in 125 transitions.

Both algorithms have as a common set of parameters the following:

- Maximum number of Evaluations = 40 000 000;
- Population Size = {100, 500};
- Generation Gap = 1;
- Single Point mutation rate = {1%, 5%, 10%};
- Elitism strategy + Tournament Selection. Tournament size = {2, 5}.

For the GA, two point crossover was restricted to gene boundaries and the maximum graph crossover size was set to 3. The parameter V in the POSATM was set to 1, and in all cases POSATM was run using seeding of the initial generation.

Despite its simplicity, our bivariate model has been shown to be superior to both, one point crossover and graph based crossover GAs for the experiments conducted. The results are impressive: POSATM was able to find 42 times the maximum value, whereas the best GA only 6 times (see table 4). On the other hand, the average productivity of the best individual in the last generation was clearly superior for the probabilistic algorithm (see table 5).

Table 4: Results for the 4-tuple BB(6). Number of runs in which the maximum was reached. Blanks cells indicates that none of the 30 runs reached the maximum. N- popsize; T- tournament size; A0- GA with 2 point crossover; A1-Graph based crossover; A2-POSATM. Mutation is given in %.

| | N | T | 1% | 5% | 10% | Total |
|---|---|---|---|---|---|---|
| A0 | 100 | 2 | | | | |
| A0 | 100 | 5 | | | 1 | 1 |
| A0 | 500 | 2 | | | | |
| A0 | 500 | 5 | | | | |
| A1 | 100 | 2 | | 1 | | 1 |
| A1 | 100 | 5 | 1 | 1 | | 2 |
| A1 | 500 | 2 | 1 | | | 1 |
| A1 | 500 | 5 | | | 2 | 2 |
| A2 | 100 | 2 | 4 | 5 | 0 | 9 |
| A2 | 100 | 5 | 2 | 6 | 5 | 13 |
| A2 | 500 | 2 | 1 | 2 | 1 | 4 |
| A2 | 500 | 5 | 0 | 14 | 2 | 16 |

Results confirm that the bivariate model is able to capture relevant information about variables representing actions, and that POSAMT efficiently incorporates this information to the search for solutions. Here, it is worth to remember that the cost of evaluation of POSATM is not high. It simply computes one bivariate marginal, and for each member of the selected set, updates some variables (usually 1) using the computed probabilities.

Another experiment (table6) was oriented to explore the influence of probabilistic information in the behavior of the algorithm. We run a POSATM (EA1) where probabilistic information is not used. In the step 2 of the algorithm the new value is chosen randomly using a uniform distribution. EA2 is the same algorithm (it has the same parameters), but using the probabilistic information.

Finally, we investigate the role of parameter V in the behavior of the algorithm. EA3 shares all the parameters with EA2 except parameter V that has been set to 3, it means that in the step 2 of the algorithm 3 variables modify their values. Results deteriorate, and this fact can be explained by the magnitude of the disruptive effect.

Table 5: Results for the 4-tuple BB(6). Productivity of the best individual in the final population. Each experiment was repeated 30 times. The results are averaged. N- popsize; T- tournament size; A0- GA with 2 point crossover; A1-Graph based crossover; A2-POSATM. Mutation is given in %.

| | N | T | 1% | 5% | 10% | |
|---|---|---|---|---|---|---|
| A0 | 100 | 2 | 8.5 | 9.8 | 9.3 | 9.2 |
| A0 | 100 | 5 | 7.4 | 14.3 | 10.6 | 10.8 |
| A0 | 500 | 2 | 8.6 | 8.2 | 8.1 | 8.3 |
| A0 | 500 | 5 | 7.9 | 12.6 | 9.0 | 9.8 |
| A1 | 100 | 2 | 14.6 | 12.1 | 9.9 | 12.2 |
| A1 | 100 | 5 | 10.4 | 16.1 | 12.2 | 12.9 |
| A1 | 500 | 2 | 13.7 | 10.5 | 8.6 | 10.9 |
| A1 | 500 | 5 | 9.1 | 16.9 | 10.8 | 12.3 |
| A2 | 100 | 2 | 16.7 | 15.8 | 14.1 | 15.5 |
| A2 | 100 | 5 | 13.7 | 19.2 | 16.8 | 16.8 |
| A2 | 500 | 2 | 13.3 | 15.4 | 13.7 | 13.7 |
| A2 | 500 | 5 | 14.8 | 19.9 | 17.4 | 17.4 |

Table 6: Best productivity reached in a number of functions evaluations for different variants of POSATM. NE- number of evaluations multiplied by $10^{-7}$. Results averaged in 30 runs.

| NE | 0.25 | 0.5 | 1 | 1.5 | 2.5 | 3 |
|---|---|---|---|---|---|---|
| EA1 | 12.25 | 14.4 | 16.2 | 16.3 | 16.4 | 16.5 |
| EA2 | 17.2 | 18 | 19 | 19.4 | 19.8 | 20 |
| EA3 | 10.2 | 12 | 12.5 | 13.0 | 13.5 | 14 |

## 8 CONCLUSIONS

We recall the main goal of the paper: to study the effectiveness of EDA-like approaches to problems that are based in a non-binary representation, and have graph-like structures. We consider the obtained results encouraging. First of all, the experiments have shown that at least under certain conditions an EDA-like approach seems to outperform recombinative methods in the optimization of problems with graph-like structures. This result is highlighted by the fact that the cost of evaluation of POSATM is not high with respect to the previous GA approaches.

Besides, we have been able to show that it is possible to construct useful (from the optimization point of view) partial probabilistic models based only on the dependencies displayed for a subset of the variables. This is of course an important issue because many problems could benefit from a preliminary variable selection step.

We would like to point out that we consider the model studied here as preliminary and that we are confident that much better results can be achieved following this line of thinking. We plan to develop new and better EDA-like approaches to this class of problems.

Finally, it is worth noting, that although we have explored here a concrete problem, we believe that our results have a more general scope, they could be applied also to other Turing or Finite State Machine problems.

## References

[1] Rado, T. (1962) On non-computable functions, *The Bell System Technical Journal*, vol. 41, no. 3, pp.877- 884.

[2] Mühlenbein, H., Paass G. (1996). From recombination of genes to the estimation of distributions {I}. *Binary parameters.* pp 178-187.

[3] Mühlenbein, H., Mahnig, T., Ochoa A. (1999). Schemata, Distributions and Graphical Models in Evolutionary Optimization, *Journal of Heuristics* Vol 5, No. 2. pp. 215-247.

[4] Soto M., Ochoa A., Acid S., De Campos L. (1999). Introducing the Polytree Approximation of Distribution Algorithm. *Proceedings of the II Second Symposium on Artificial Intelligence CIMAF99.* pp. 360-367.

[5] Fogel, L. J., Owens J. A., Walsh M. J. *Artificial Intelligence through simulated evolution.* John Wiley & Sons Publishers, 1967.

[6] Booth, T. L. *Sequential machines and automata theory* (1967) John Wiley & Sons Publishers.

[7] Ochoa A., Soto M., Santana R., Madera J. C., Jorge N. (1999). The Factorized Distribution Algorithm and The Junction Tree: A Learning Perspective. *Proceedings of the II Second Symposium on Artificial Intelligence CIMAF99.* pp. 368-377.

[8] Salustowicz, R. , Schmidhuber J. (1997). Probabilistic Incremental Program Evolution. *Evolutionary Computation* 5(2): pp. 123-141.

[9] Wood, D. (1987). *Theory of Computation*, Harper and Row, Publishers.

[10] Boolos, G., and Jeffrey, R. (1995). *Computability and Logic*, Cambridge University Press.

[11] Jones, T., Rawlins, G. (1993) Reverse HillClimbing, Genetic Algorithms and the Busy Beaver Problem, In Forrest, S. (Ed.), *Genetic Algorithms: Proceedings of the Fifth International*

*Conference (ICGA-93)*. San Mateo, CA: Morgan Kaufman, pp 70-75.

[12] Pereira, F. B., Machado, P., Costa E. and Cardoso, A. (1999). Busy Beaver: An Evolutionary Approach. *Proceedings of the Second Symposium on Artificial Intelligence*, Havana, Cuba. pp. 212-216.

[13] Machado, P., Pereira, F. B., Cardoso, A. and Costa E. (1999). Busy Beaver: The Influence of Representation. *Proceedings of the 2nd European Workshop on Genetic Programming*, Goteborg, Sweden. pp.

[14] Marxen, H. Buntrock, J. (1990). Attacking Busy Beaver 5, *Bulletin of the European Association for Theoretical Computer Science*, Vol 40.

[15] Pereira, F. B., Machado, P., Costa E. and Cardoso, A. (1999). Graph Based Crossover A Case Study with the Busy Beaver Problem. Banzhaf W., Daida J., Eiben A. E., Garzon M. H., Honavar V., Jakiela M. and Smith R. E. (Eds.). *Proceedings of the Genetic and Evolutionary Computation Conference GECCO 99.* Volume II (pp. 1149-1155) Orlando Fl. Morgan Kauffman Publishers. San Francisco, California.

[16] Hancock P. J. B. (1992) Genetic Algorithms and permutation problems: a comparison of recombination operators for neural net structure specification," in P*roceedings of the International Workshop on Combination of Genetic Algorithms and Neural Networks (COGANN-92)* ( D. Whitley and J. D. Schaffer, eds.) pp. 108-122, IEEE Computer Society Press, Los Alamitos, CA.

[17] Ponce de Leon, E., Santana, R., Ochoa A. (1997). A genetic algorithm for a Hamiltonian path problem: mutation - crossover interaction. In *Proceedings of the 13th ISPE/IEE International Conference on CAD/CAM, Robotics and Factories of the Future.* pp. 788-793.