Busy Beaver – An Evolutionary Approach

Francisco B. Pereira[*], Penousal Machado[*], Ernesto Costa[**], Amílcar Cardoso[**]

[*] *Instituto Superior de Engenharia de Coimbra*
Quinta da Nora
3030 Coimbra, Portugal
E-mail: {xico, machado}@dei.uc.pt
[*,**] *Centro de Informática e Sistemas da Universidade de Coimbra*
Polo II da Universidade de Coimbra, Departamento de Engenharia Informática
3030 Coimbra, Portugal
{ernesto, amilcar}@dei.uc.pt

The Busy Beaver is an interesting theoretical problem proposed by Tibor Rado in 1962. Since then, it has attracted the attention of many researchers and several contests were organised trying to produce good solutions. In this paper, we propose an evolutionary approach to the problem. Our experimental results prove that this technique is very effective in attacking Busy Beaver, since we were able to find several Turing machines that outperform previous best-known solutions.

Keywords: Genetic Algorithms, Turing Machines, Non-computable Functions

1. Introduction

One of the most important results of theoretical computer science deals with the existence of non-computable functions. This fact can be easily established showing that there are functions, which are not Turing computable: there are more functions than Turing Machines to compute them.

In 1962 Tibor Rado proposed one such function based on what is known today as the "Busy Beaver Game or Problem" [11]. It can be described as follows: Suppose a Turing Machine (TM) with a two way infinite tape and a tape alphabet = {blank, **1**}. The question Rado asked was: What is the maximum number of **1**'s that can be written by an N-State (N does not include the final state) halting TM, when started 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 Busy Beaver (BB).

The problem with $\Sigma(N)$ is that it grows faster than any computable function, i.e., $\Sigma(N)$ is non-computable. Some values for $\Sigma(N)$, and the corresponding TMs are known today for small values of N. We have, for instance, $\Sigma(1) = 1$, $\Sigma(2) = 4$, $\Sigma(3) = 6$, $\Sigma(4) = 13$. As the number of states increases the problem becomes harder, and, for $N >= 5$, we have several candidates (or contenders) which set lower bounds on the value of $\Sigma(N)$. This is partially due to the fact that there is, neither a general, nor a particular theory about the structure of a BB. The only available technique for finding such machines is to perform an exhaustive search of all N-state TM. Current used techniques perform a partial search on the solution space, looking for TMs that produce the best lower bound for the value of $\Sigma(N)$. Some of the best contenders were obtained be Marxen [10] (E.g., he established that $\Sigma(5) >= 4098$). His approach involves enumeration and simulation of (nearly) all N-state TMs, using several techniques to reduce the number of inspected machines, accelerate simulation and determine non-termination.

In the original setting, the problem was defined for 5-tuple TMs. With this definition, machines, given a current state and the symbol being scanned in the tape, write a symbol over it, enter a new state and move the read/write head left or right. One of the main variants consists in considering 4-tuples TM. The main difference from the others is that, during the transition to a new state, a TM either writes a new symbol to the tape or moves the head (both actions are not simultaneously allowed).

In this paper we will address the problem of finding promising candidates for the 4-tuple seven state Busy Beaver, BB(7). Our approach uses Genetic Algorithms (GA) with learning and proved to be extremely effective. The previous best candidate [9] produced 37 ones. We found several machines with higher productivity, showing that $\sum(7)>= 102$. This machine was found in less than one day, using a 300MHz Pentium II computer. Only (8.5e-11)% of the search space (4(N+1))^(2N) was evaluated, and, during the process we also found TM with productivity 37, 39, 41 and 100.

Several researchers attempted to find good solutions to the BB problem using a broad variety of techniques. The previous 4-tuple BB(7) best candidate was obtained using an abstract representation of TMs [9]. Terry Jones used GA to attack 5-tuple BB [7], [8], and concluded that techniques inspired in hill-climbing achieved better results.

The paper has the following structure: Section 2 comprises a formal definition of five and 4-tuple TMs, and the specification of the rules of the Busy Beaver problem for each of these variants. In Section 3 we introduce our approach. Next, in Section 4, we present and analyse the experimental results. Finally, in Section 5, we state some overall conclusion and suggest some directions for future work.

2. Problem Definition

A deterministic TM can be specified by a sextuple $(Q,\Pi,\Gamma,\delta,s,f)$, where[12]:

- Q is a finite set of states
- Π is an alphabet of input symbols
- Γ is an alphabet of tape symbols
- δ is the transition function
- s in Q is the start state
- f in Q is the final state.

The original definition proposed by Rado [11], 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 Turing Machine (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 $\sum(N)$ is defined to be the maximum productivity that can be achieved by a N-state TM. This TM is called a Busy Beaver [11].

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 [2].

3. Our Approach

Genetic Algorithms are probabilistic search procedures, inspired by mechanisms that exist in biological systems, such as natural selection and genetics [3], [6]. They have been used to solve hard problems (those with a huge and multimodal search space), because, typically, they only need to explore a small portion of the space. A GA maintains an evolving population of individuals (points from the landscape), which are subject to selection forces in the presence of several genetic operators, like crossover and mutation. Hopefully, this process will lead to the discovery of new and fitter populations.

Representation

Each individual, defined by its chromosome, is a possible solution to the problem. In the BB(7), all TM with 7 states are potential solutions, since they belong to the fitness landscape. In our approach, the chromosome of each individual is codified as a binary string with the following format:

Each block, with 5 bits, has information about a particular transition of the TM: 3 bits representing the new state (7 states plus the halting state) and 2 bits representing the action performed (write a blank, write a one, move left, move right).

With this codification we can apply standard genetic operators for crossover and mutation and be sure that the descendants are always legal TMs.

Evaluation

The evaluation phase comprises interpretation of each chromosome and simulation of the resulting TM.

During the process of simulation, we convert each TM to its Tree Normal Form [10]. It is clear that there are several TMs with the same behaviour that can be considered equivalent. If we could construct sets of machines with equivalent behaviour we would only need to run one of the machines of the set. The most important equivalent class is known as the *Tree Normal Form* (TNF) [10]. Using a TNF representation ensures that machines differing only in the naming of the states or in transitions that never are used, are represented in the same way.

We believe that one of the most important factors in genetic optimization, and more specifically in the BB problem, is the assignment of fitness. One possible approach [8] is to consider as fitness the productivity of the TM. We think that this evaluation can be improved. Therefore, we contemplate the following factors in order of importance:

1º Halting before reaching a predefined number of steps.

2º Accordance to the rules [2].

3º Productivity.

4º Number of used transitions.

5º Number of steps made before halting.

The idea is to establish and explore the differences between "bad" individuals, e.g. a machine that never leaves state 1 is considered worse than one that goes trough all the states, even if they have the same productivity.

Learning

We give to several individuals of each generation the ability to learn. Individual learning, in this context, is implemented as an iterative local search algorithm. The fitness of the learning individual will be improved by searching for better points in its neighborhood. This will lead to a better understanding of the individual's quality. Fitness will be a measure, not only of the initial quality of the individual, but also, of its ability to evolve [1]. In our approach we use Lamarckian learning, implemented trough a hill-climbing algorithm [5].

4. Experimental Results

The experiments were performed with Genesis 5.0 [4]. We expanded the package by integrating the hill-climbing learning procedure. The parameters of the GA were the following: Population Size = {200, 1000}, Number of Evaluations = 25000000, two point crossover with rate = 70%, single point mutation with rate = {1%, 5%, 10%}, Learning rate =25% (25% of the individuals in each generation will learn) and learning length = 15. The learning length specifies the number of steps that the hill-climber makes, each of these steps counting as an evaluation.

We used probabilistic selection and both Elitist and non-Elitist strategies. The results presented are averaged over sets of ten runs, performed with the same initial conditions and with different random seeds.

A summary of the results is presented in Table 1.

Pop. Size	Mutation	Learning		Without Learning	
		Elitist	Non Elitist	Elitist	Non Elitist
200	1%	18	16	14	13
200	5%	23	18	18	17
200	10%	22	21	20	18
1000	1%	18	16	12	12
1000	5%	20	18	17	14
1000	10%	22	20	20	19

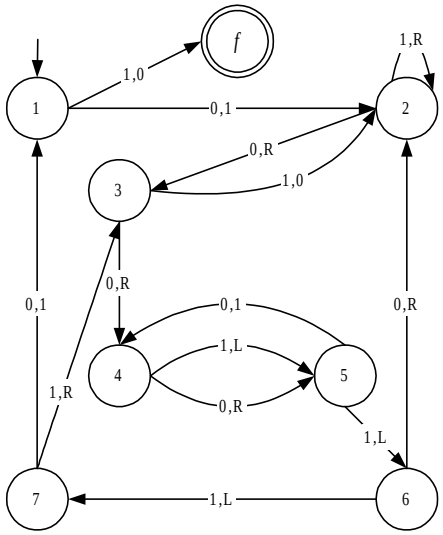
Table 1 – Best solution in the last generation averaged over 10 runs

From the results achieved it is clear that, independently of the learning and elitism strategies, experiments with high mutation rates attain better results. This can be explained by the importance of keeping genetic diversity while searching a highly irregular landscape. There is a strong pressure for the GA to converge to local optima and mutation may help to escape from such points. The GA showed to be relatively insensible to population size variations.

A GA with elitism outperforms a standard GA and the same effect is visible (and amplified) when we add a learning procedure. Combining both strategies further enhances the results. The learning strategy decreases the influence of the mutation rate. This can be explained by the fact that learning, by performing several conditional mutations, gives the opportunity to bad individuals to climb to a higher point.

The results presented here are not as conclusive as we desire and can be considered preliminary. Further testing,

with longer runs and with different configurations are necessary. In Figure 1 we present our best 4-tuple TM candidate and its transition table.



δ	By blank		By one	
State	New State	Action	New State	Action
1	2	1	*f*	L
2	3	R	2	R
3	4	R	2	0
4	5	R	5	L
5	4	1	6	L
6	2	R	7	L
7	1	1	3	R

Figure 1: A seven state 4-tuple TM and its corresponding transition table. The blank symbol is represented by 0. This machine is the best known 4-tuple BB(7) candidate.

5. Conclusions and Future Work

We applied GA with learning to the BB problem, a theoretical non-computable problem interesting by itself. This technique proved to be effective in attacking BB(7), since we regularly achieved machines with productivities higher than 37 (the previous known maximum). GA proved to be an efficient way for minimizing the number of inspected TMs.

We also have applied this approach to BB(6) and BB(8). In BB(6), we found machines equivalent to the current best known candidate (productivity 21). In BB(8), our current best candidate writes 384 ones. This is the best one known, but we are confident on the existence of better machines. We found that the increase in the number of GA evaluations needed to find good candidates, when passing from BB(6) to BB(7) was not significant. This indicates that the evolutionary approach handles the scalability issue efficiently.

As future work, we intend to test new selection methods (e.g., tournament selection) and new learning models (e.g., Baldwin Effect). We are also considering a distributed GA implementation, with several populations and migration mechanisms.

To attack BB(N), N >= 9, we need to make several improvements to the simulation of the TM in order to decrease the simulation time. Some of the possibilities are: using macro-transitions, avoid the evaluation of equivalent machines and early detection of non-halting machines [10].

6. Acknowledgments

This work was partially funded by the Portuguese Ministery of Science and Technology, under Program PRAXIS XXI.

References

[1] Belew R. and Mitchell, M. (1996). Introduction. In Belew, R. and Mitchell, M. (Eds.), *Adaptive Individuals in Evolving Populations: Models and Algorithms*, Proceedings Volume XXVI Santa Fe Institute, Studies in the Sciences of Complexity, Addison-Wesley, pp. 1-22.

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

[3] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley.

[4] Grefenstette, J. (1990). A User's Guide to Genesis 5.0.

[5] Hart, W. and Belew, R. (1996). Optimization with Genetic Algorithm Hybrids that Use Local Search. In Belew, R. and Mitchell, M. (Eds.), *Adaptive Individuals in Evolving Populations: Models and Algorithms*, Proceedings Volume XXVI Santa Fe Institute, Studies in the Sciences of Complexity, Addison-Wesley, pp. 483-496.

[6] Holland, J. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press.

[7] Jones, T. (1995). Crossover, Macromutation, and Population-based Search, *Santa Fe Institute WP 95-02-24*.

[8] 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 Kaufmann, pp 70-75.

[9] Lally, A., Reineke, J., and Weader, J. (1997). An Abstract Representation of Busy Beaver Candidate Turing Machines.

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

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

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