In the II Symposium on Artificial Intelligence and Adaptive Systems, ISAS'99. H. Möhlenbein and A. Odnoa (ed.), La Habana. 236-241

An Evolutionary Cellular Program on the Solution of the Travelling Salesman Problem *

José A. Moreno, Adriana G. Egea [b]

Laboratorio de Computación Emergente,
Facultad de Ciencias,
Universidad Central de Venezuela.
Caracas - Venezuela. {jose,aegea}@neurona.ciens.ucv.ve
[b] *Current address: Wessex Institute of Technology, Ashurst, Southampton SO407AA, UK. e-mail: aegea@wessex.ac.uk*
http : //www.wessex.ac.uk/~aegea

An evolutionary algorithm is described for solving instances of the Travelling Salesman Problem (TSP). The key point of the algorithm is the distribution of the population over a grid, being in that sense a sort of cellular automata, but having as rules of change an iteration of a genetic algorithm each time. This genetic algorithm operating in miniature just takes on account a subset of the whole population, using for that a given neighborhood relation, among several defined.

This work compares the behaviour of the algorithm against a genetic algorithm, for different program's parameters. The set of benchmark instances of TSP was taken from the worldwide known collection TSPLIB.

1. Introduction

Resolution of problems can be seen as searching into a solutions' space, where it is needed to find the best one. When the problem is elementary, this can be done by using thorough analysis.

Unfortunately, many real-world problems are so large and difficult that these methods cannot be applied due their large storage or computational-time requirements.

Thus, it is a good approach designing algorithms which instead of searching an absolute optimum, will return, in a reasonable computing time, solutions that are quite close to the optimum. Such algorithms are called heuristics, and the program presented here certainly belong to this class of algorithms.

We discuss briefly about combinatorial optimisation on section 2 and in section 3 about the TSP, as a particular problem of this kind. In section 4 we consider evolutionary algorithms as heuristics to deal with this kind of problem, paying especial attention just to two of them: genetic algorithms and cellular automata. Section 5 is about the program we present here. Finally the experimental design and conclusions are presented in section 6.

* This work was partially supported by Consejo Nacional de Investigaciones Científicas y Tecnológicas (CONICIT).

2. Combinatorial Optimisation

An instance of a optimisation problem is a pair (F, c), where F is any set, the domain of feasible points; and c is the function of costs, a mapping

$$c : F \rightarrow \Re^1 \tag{1}$$

The problem is to find an $f \in F$ for which

$$c(f) \geq c(y) \; \forall y \in F \tag{2}$$

such a point f is called a *globally optimal solution* to the given instance, or, when no confusion can arise, simply an *optimal solution* [?].

Optimisation problems can be classified in two categories: those with *continuous* variables, and those with *discrete* ones, which are called *combinatorial*.

In combinatorial problems we are looking for an object from a finite (or possibly countably infinite) set. This object is typically an integer, set, permutation or graph, instead of a real number as in the other kind of problems.

There are many types of optimisation problems, each one with characteristics that makes their resolution expensive in computing resources or not. According to this, there are problems more difficult to be solved than others. Actually, there is an important class of problems for which no efficient algorithm is known: the $NP-complete$ problem class. This class has the following properties:

1. No NP-complete problem can be solved by any known polynomial algorithm

2. If there is a polynomial algorithm for *any* NP-complete problem, then there are polynomial algorithms for *all* NP-complete problems [9], [11].

That is why it is so interesting trying to solve efficiently any specific problem of this class. And, as we show further, the Travelling Salesman Problem is one of them.

3. The Travelling Salesman Problem

The Travelling Saleman Problem (TSP) is one of the most known and representative of the class of NP-complete problems, which has been investigated profusely [2], [4], [7], [9], [11], [?], [22], [28], [32], [33], [34] and that is why it is elegible to test our evolutionary program. In TSP, the cost function c to optimize is the total distance of a travelling salesman's tour, whose restrictions are to go through a set of cities, visiting only once each one of them, and returning to the first city visited. In an instance of the TSP we are given an integer $n > 0$ and the distance between every pair of the n cities in the form of an $n \times n$ matrix $[d_{i,j}]$ (not necessarily simmetric). Being F the feasible domain composed by all cyclic permutations π on n objects, the cost c maps π to

$$c = \sum_{j=1}^{n} d_{j,\pi(j)} \tag{3}$$

where $\pi(i)$ must be interpreted as the city visited after city i according to the permutation π.

The computing time required to solve this problem grows exponentially because there are $\frac{(n-1)!}{2}$ feasibles solutions in any instance of size n [4] and there is no polynomial algorithm to find the desired optimum. It is clear then that the enumerative way becomes nonviable to solve huge problems, and is essential the use of heuristics in these cases, and the evolutionary programming becomes an interesting option.

4. Evolutionary Algorithms

One class of heuristics is constituted by evolutionary programs. These are variants of the Holland's genetic algorithm [10], which have been widely studied in the last years and have been used mainly as function optimizers [8], [16].

Researchers have performed several changes and additions to this algorithm, limiting their generality in exchange for improving their performance on the solution of the specific problem. Nevertheless, in spite of all of these modifications, always is preserved the primal Darwinian idea of evolution of the species. That is why they are called evolutionary programs, or evolutionary algorithms (EAs) [5], [18], [18], [32].

There is a huge diversity of evolutionary computational models that have been proposed and studied, some of them even before Holland's work, and many authors agree in classify them in this category. EAs share a conceptual basis of simulation the individual structures' evolution by means selection and reproduction processes, opposite to other optimisation methods, which work with just a single solution instead a population of them. The evolutionary process depends on the performance of the individual structures (fitness) into the environment where it belongs.

Specifically, EAs hold a structures' population that evolves in accordance with certain rules of selection [1], [3], [7], [13], [20], [30], recombination[3], [4], [7], [13], [18], [27] and mutation[1], [3], [6], [7], [13], [17],

[18] . Each individual is evaluated and rated according its fitness. Every individual of the population has a score, which is used in the selection process. This process focuses in well-adapted individuals, but also generally allows poorly adapted survive (with fewer probabilities). After that, recombination and mutation operate over the selected structures, generating new information, which could improve the solutions population.

We are concentrated in two specific families of algorithms belonging to this class. They are the genetic algorithms and the cellular automata.

4.1. Genetic Algorithms

On one hand, genetic algorithms (GA) have all of these evolutionary processes, and most of them include some modifications to the original Holland's algorithm. For example, with the inclusion of an elitist criterion, it is possible to preserve the best structure found in the current generation into the next one. The generation gap strategies preserve not only the best, but a big percentage of individuals, removing just the worse ones.

As it is easily inferable, not all the genetic algorithms' iterations will imply the same computational work. That is why many authors compare their results using the number of substitutions as measure instead of number of generations or computational time. Both measures do not depend on the computational work itself but on other factors, as the design of the algorithm or the hardware used.

In spite of being simple from a biological point of view, these algorithms are the sufficiently complex thing like providing robust and adaptive mechanisms search [18], [25]. Some of them (the genetic algorithms) have even been compared very favorably with respect to non-evolutionary methods, like statistical methods [21].

4.2. Cellular Automata

On the other hand, we have cellular automata (CA). These are discrete and dynamic systems whose behaviour is completely specified on terms of a local relationship. The cellular automata can be seen as a universe on miniature, where the space is represented as a uniform grid and each cell contain just an elementary data. Here, the time progresses on discrete steps [12], [26].

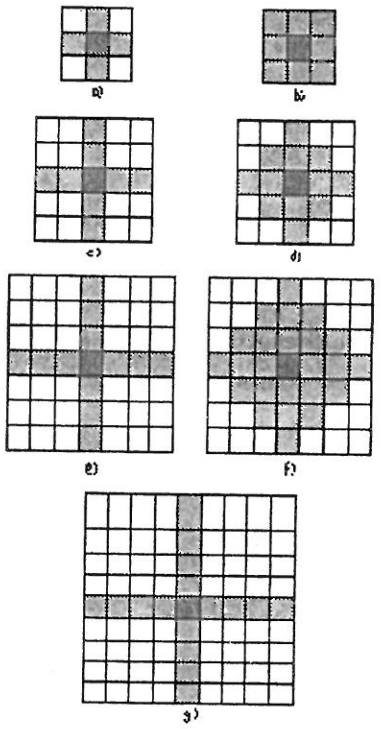

Figure 1. Neighbors (shaded cells) respect to the central (dark) according to the neighborhoods: a) VonNeumann, b) Moore, c) L2, d) R2, e) L3, f) R3 and g) L4

The laws of the universe are simple rules, local and uniform, which allows that each cell change of state according to the values of their neighbour cells. Thus, the relationship of neighbourhood plays a very important role in the dynamic system, because it allows the interaction and defines the information flow [24]. There are many shapes and sizes of neighborhood, and both characteristics affects the behavior of the automata, because they affect directly the information flow through the mesh. Some of these neighborhood are illustrated at Figure 1.

5. Cellular Genetic Algorithm

We present here a cellular genetic algorithm (CGA), called this way because it is a CA in which local laws are given by an iteration of a GA. Previ-

ously, other researchers have worked along this idea, [23], [31].

The population in each case is composed by the individuals belonging to the neighbourhood previously defined. The rule for the parents selection can be random or deterministic (but always depending on the fitness), because the size of the population is small enough to handle it. Nevertheless, this is still a stochastic algorithm, because the recombination and mutation are stochastic as well.

Because of the size of sub-populations, there is a quicker updating of the whole population. That is why other kinds of modifications to the GA perform better here, such as the use of elitism and elimination of repeated cells just in the moment of their creation.

It worth to be noted although, that those criteria are not absolutes anymore, because there are no more global interactions but just local. In that sense, it is impossible guarantee that the newly produced offspring does not exist anywhere into the grid but in the neighborhood knowing it is possible.

To evaluate appropriately the behaviour of CGA, it was subjected to several problems taken of the collection TSPLIB[1]. A comparison of these results against to those given by a typical genetic algorithm is performed, varying their input parameters.

The genetic algorithm chosen was the Whitley-Starkweather-Fuquay's [29] slightly modified. As a result, we took in account population size, maximum number of substitution, use of elitism, elimination of repeated cells in the neighbourhood and use of generation gap, haing as result an algorithm that we called TGA.

Additionally to those parameters, in our CGA we include in the study: type of neighbourhood used and synchronous or asynchronous updating of the grid.

The importance of changing values to these parameters lay in tuning the genetic search, because it affects both selective pressure and population diversity. As we comment in the following section, these two factors affect the performance of evolutionary algorithms.

[1] Available via ftp at the host `softlib.cs.rice.edu`

6. Experiment and Conclusions

The experimentation is being performed using the evolutionary programs TGA and CGA varying the input parameters shown in Table 1.

TGA < cities_file.tsp > < output_file > < population_size > < max.sustit >< elitism > < No_re_offspring > < gap >
CGA < cities_file.tsp > < output_file >< mesh_order > < max.sustit >< elitism > < No_re_offspring >< gap > < neighborhood >< updating >

Table 1:Parameters of the algorithms

The experimental values of test for the indicated input parameters vary as is specified in Table 2.

Parameters	*Values*
Cities file	From TSPLIB
Population size (TGA)	100 to 5625
Mesh Order	10 to 75
Max. Sustitut.	1000 to 50000
Elitism	0 /1
No repeated offspring	0 /1
Generational Gap	0 /0.1
Neighborhood (CGA)	Moore/Von Neumann /L2/L3/L4/L6/R2/R3
Updating	Synchronous / Asynchronous

Table 2:Values of the experimental parameters

The first computational results produced let us infer that the cellular genetic algorithm behaves better than the traditional genetic algorithm. The tendency to fall in a local optimal solution is lower. Althought the test have not been run for many problems, similar results are expected for most of the cases.

Most of the times, increasing the order or the grid does not imply improve the results obtained, and we strongly believe that the other parameters have a more critical influence on the convergence of the algorithm.

Regarding the neighborhoods, it is likely to be true the following hypothesis only for huge instances of

TSP: the greater is the radius, the faster is the convergence. But the fewer cells belonging to the neighborhood, the lower is the tendency to obtain degenerated solutions. The inclusion of generational gap doesn't seem as critical in the CGA as in the TGA, while the criterium of eliminating identical offsprings is more effective in the TGA rather in the CGA.

As an immediate future work we planned incorporate more instances of the TSP to our experimentation, while a more long term future work could be the implementation of this algorithm in a parallel system, and then study the influence of the neighborhoods in this changed environment.

References

[1] Bramlette, M.F., Initialization, Mutation and Selection Methods in Genetic Algorithms for Function Optimization" *4th. Int. Conference on Genetic Algorithms*, Morgan-Kauffmann, 1991, pp. 100-107.

[2] Carrasquero, N. and J.A. Moreno, A new genetic operator for the travelling salesman problem. Working Paper, Laboratorio de Computación Emergente, Fac. Ingeniería, UCV, 1998.

[3] Davis, L., Handbook of Genetic Programming. Van Nostrand Reinhold, 1991.

[4] De Jong, K. and W.M. Spears, Using Genetic Algorithms to Solve NP-Complete Problems, *3rd. Int. Conference on Genetic Algorithms*, Morgan-Kauffmann, 1989, pp. 124-132.

[5] De Jong, K. and W. M. Spears, On the State of Evolutionary Computation, *5th. Int. Conference on Genetic Algorithms*, Morgan-Kauffmann, 1993, pp. 618-623.

[6] Fogarty, T.C., Varying the Probability of Mutation in the Genetic Algorithm, *3rd. Int. Conference on Genetic Algorithms*, Morgan-Kauffmann, 1989, pp. 104-109.

[7] Goldberg, D.E., Genetic Algorithm in Search, Optimization and Machine Learning, Addison-Wesley, 1989.

[8] Gordon, V.S. and D. Whitley, Serial and Parallel Genetic Algorithms as Function Optimizers, *3rd. Int. Conference on Genetic Algorithms*, Morgan-Kauffmann, 1989, pp. 104-109.

[9] Grötschel, M., L. Lovász and A. Schrijver, Geometric Algorithms and Combinatorial Optimization, Springer-Verlag, 1988.

[10] Holland, J. Genetic Algorithms, *Scientific American*, July 1992, pp. 44-50.

[11] Lewis, H.R., Elements of the Theory of Computation, Prentice-Hall, 1998.

[12] Margolus, N. and T. Toffoli, Cellular Automata Machines. A New Environment for Modelling, MIT Press Series in Scientific Computation, 1987.

[13] Michalewicz, Z., Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 1994.

[14] Mühlenbein, H. et al., Evolution Algorithms in Combinatorial Optimization, *Parallel Computing* 7:65-88, 1988.

[15] Mühlenbein, H., Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization, *3rd. Int. Conference on Genetic Algorithms*, Morgan-Kauffmann, 1989, pp. 416-421.

[16] Mühlenbein, H. et al, The Parallel Genetic Algorithm as Function Optimizer, *Parallel Computing* 17:619-632, 1991.

[17] Mühlenbein, H., How Genetic Algorithm Really Work. I: Mutation and Hillclimbing, *Parallel Problem Solving from Nature (PPSN II)*, ed. Männer and Manderick, North-Holland, 1992, pp. 15-26.

[18] Mühlenbein, H., Optimal Interaction of Mutation and Crossover in the Breeder Genetic Algorithm, *Proc. of Fifth Int. Conf. on Genetic Algorithms (ICGA-93)*, Morgan Kaufmann, 1993, pp. 648.

[19] Mühlenbein, H., Evolutionary Algorithms: Theory and Applications, 1993. Technical Paper. Available via ftp at ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-93_03.ps

[20] Mühlenbein, H., On the Mean Convergence Time of Evolutionary Algorithms for Parameter Optimization, 1994.

[21] Reeves, C.R. and C.C.Wright, Genetic Algorithms and Statistical Methods: A Comparison, *Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, 12-14 September 1995, Conference Publication No. 414, pp. 137-140

[22] Rintala, T., Population Size in GAs for TSP, *2nd Nordic Workshop on Genetic Algorithms and their Applications*, 1996. Available via www at http://www.uwasa.fi/cs/publications/2NWGA.html.

[23] Rudolph, G. y J. Sprave, A Cellular Genetic Algorithm with Self-Adjusting Acceptance Threshold, *Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)*, 12-14 September 1995, Conference Publication No. 414, pp. 365-372.

[24] Sarma, J. y K. De Jong, An Analysis of the Effects of Neighborhood Size and Shape on Local Selection Algorithms. *Fourth International Conference on Parallel Problem Solving from Nature (PPSN96)*, Sept. 22-26, 1996.

[25] Spears, W. M., The Role of Mutation and Recombination in Evolutionary Algorithms, PhD. Thesis, George Mason University, 1998.

[26] Stauffer, D., Computer Simulations of Cellular Automata, J. Phys. A. Math. Gen. 24 (1991), pp.909-927.

[27] Syswerda, G., Uniform Crossover in Genetic Algorithms, *3rd. Int. Conference on Genetic Algorithms*, Morgan-Kauffmann, 1989, pp. 2-9.

[28] Telfar, G., Generally Applicable Heuristics for Global Optimisation: An Investigation of Algorithm Performance