ARTIFICIAL EVOLUTION THOROUGH DYNAMIC POPULATION, COOPERATION AND COMPETITION

M. ANNUNZIATO, S. PIZZUTI

E.N.E.A. , Energy Department, Casaccia R. C.

Via Anguillarese, 301, 00060, Rome, Italy

Tel : +39-0630484411, Fax : +39-0630484811

E-Mail : stefano.pizzuti@casaccia.enea.it

Abstract

In this work we propose an innovative evolutionary strategy. It starts from the Holland's traditional simple genetic algorithm and moves towards artificial evolution by introducing new features. The first one concerns dynamic population. It means that, during the generations, the population size can grow up and shrink. Then the classic genetic operators, crossover and mutation, are considered as different reproduction strategies and real competition is introduced. In the paper a description and a discussion of the strategy are reported, moreover benchmarks on classical TSP problems are shown.

Keywords : Genetic Algorithms, Evolutionary Strategies, Computational Intelligence, Soft Computing, Combinatorial Optimization, Adaptive Behaviour

1 INTRODUCTION

Genetic and Evolutionary Algorithms are problem solving techniques with an astonishing property: they solve problems by evolving solutions as nature does [38].

J. Holland [9] in 1975 was the first who introduced genetic algorithms. These algorithms encode a potential solution to a specific problem on a simple chromosome-like data structure and apply recombination operators to these structures so as to preserve critical information. An implementation of such an algorithm begins with a population of (typically random) chromosomes. The fitness of each of these structures is evaluated according to the cost function of the target problem and then recombination operators, crossover and mutation, are applied to generate the new population. This algorithm is often referred as the *Simple or Canonical Genetic Algorithm* [13][28]. Starting from this point a lot of researchers developed modifications of the base algorithm and many applications, especially in optimization problems, were curried out.

In particular we wish to underline the *Genitor* Algorithm [11][12][13] and *CHC* [13][14]. Other researchers [10][17][18][19][20][28] tried to hybridize GAs with other optimization techniques, like Simulated Annealing and Hill Climbing, and others studied about the operators [15][16][19][28][32]. Efforts were also made in studying the parallel versions of GAs [2][18][20][21][28] and towards evolution strategies trying to model niche and speciation [28] and isolation [1][13][28].

Moreover GAs and evolution strategies are the inspiration towards new branches like Memetic Algorithms[33] and Artificial Life [34][35][36].

In applications GAs are most used to solve optimization problems. In academy these algorithms are tested on Complete-NP problems, as the travelling salesman problem [1][3][4][8][32], while in industry are often used to perform process optimization. One particularly interesting application is that of using such algorithms to tune fuzzy systems (genetic-fuzzy systems) [22][23] and to train neural networks (neuro-genetic systems) [22][24][25].

2 DYNAMIC POPULATION, REPRODUCTION AND COMPETITION : FROM GENETICS TOWARDS ARTIFICIAL EVOLUTION

2.1 Background

Life and evolution in nature are organized at least into four fundamental levels of structure [37] : the molecular level, the cellular level, the organism level and the population level. At present the tendency is to study the molecular level through wet-bench lab techniques (wetware), the cellular and population levels with software simulations and the organism level with hardware (robotic) experiments.

What we propose in this paper is a study at population level of evolution through a new evolutionary algorithm. The aim underlying our approach is not to recreate nature as it is, but it is to move from classical genetic algorithms towards artificial evolution of populations composed by individual able to interact among them.

All studies on Genetic and Evolutionary Algorithms started from the Holland’s Simple Algorithm and moved towards several directions. However all developments have in common one original feature : the population(s) size is constant during computation. What we propose is an evolution of the Holland’s Simple Genetic Algorithm in the direction of a population able to grow, up to a maximum limit, and shrink during generations. This is performed by considering the two classical genetic operators, crossover and mutation, not so strictly genetic but, in a little more evolutionary way, like two different ways of reproducing. Moreover the fundamental concept of *competition*, nearby co-operation, is introduced.

In this context, as proposed in [37], the concepts of crossover, mutation and chromosome move towards those of *co-operation* or *sexual reproduction*, *asexual reproduction* and *individual* or *organism*.

The following paragraphs describe the steps of the evolutionary algorithm.

2.2 Initialization

Traditionally the initialization of the population is pure random. However it is possible to initialize a few individuals with a greedy strategy depending on the problem. Small rates, about 10%, of the population initialized in such a way improve speed convergence towards good solutions keeping at the same time good diversity.

2.3 Selection

In the classic Genetic Algorithm, before crossover, the selection [9][13], or reproduction [28], stage is performed. This step wants to model the natural feature that the fitter is an individual the higher the probability to survive. In the traditional approach this is implemented as a roulette wheel with slots weighted in proportion to the fitness values of the individuals. Through such a wheel a linear search is performed [28] or an intermediate population [13] is created. This is a simulation of the competition concept.

With dynamic population a real competition is introduced and the selection stage is simplified. In fact this step is simply performed by picking up the i^{th} individual of the population, for *i* from 1 to population size, and by randomly choosing the second one. Then a coin is tossed and if competition starts then the stronger individual will kill the weaker. This simple strategy mimics in a more nature-like way the evolutionary concept that the fittest individuals will survive through competition. Moreover, while in traditional GAs the probability of reproducing of a chromosome is proportional to its fitness value, here every organism has the same chance of mating and bio-diversity is enhanced.

Finally computational time is improved because the O(*n*), where *n* is the population size, complexity selection routine based on roulette wheel is removed.

2.4 Crossover and Sexual Reproduction

Crossover is that operator which allows two chromosomes to mutually exchange their genetic equipment giving rise to new sons.

In traditional Genetic Algorithms when two chromosome meet then co-operation is performed with a probability p_c a priori defined. If it happened then the two offsprings resulting from the crossover operator would replace their parents, for instance if they were fitter, in the population, else nothing happens and both originals individuals would survive in the population. This mechanism insures the population size to be constant.

The modification we introduce moves towards *sexual reproduction* : co-operation is still performed with a probability p_c, but if it were satisfied then the resulting sons would not replace their parents, they would simply be added to the population. This is much more natural than the genetic crossover because when two individuals mate and have sons these, generally, do not kill their parents as they bear.

However natural selection let only the fittest sons survive and we implemented this in a way that during bisexual reproduction new sons are generated only if fitter than their parents. It implies also that the new healthy sons have a mono-sexual reproduction during the same generation only if the mutated chromosome is fitter. Together with competition, this feature let the population naturally die out because, sooner or later, the fittest individual kills all the others. This situation reminds us of the extinction of species in nature.

Because of physical memory limitation the population cannot grow to infinity, so a limit on the maximum size of the population must be defined. When the population reaches such a limit then co-operation is destructive in the sense that sons will replace their parents if fitter. This ensures, although this limit, the evolution of the specie.

2.5 Competition

Competition has probability 1 - p_c to start. It means that when two individuals meet if they do not co-operate then they fight for survival and the stronger individual kills the weaker. This is essential to let the population not to explode in size.

Co-operation and competition imply that in the first case the population increases of two new individuals while in the latter the population decreases of one.

2.6 Mutation and Asexual Reproduction

Traditionally mutation is that operator which gets one chromosome and randomly changes the alleles of one or more genes. Generally this operator is performed with a probability p_m a priori defined and if it occurred then the mutated chromosome would replace the original one. In this case the population size is preserved because the new solution destroys the original one.

Another point of view is that of considering mutation as non destructive moving towards the concept of *asexual*. In this situation when mutation occurs an organism first clones itself and then mutates. The mutated individual doesn't replace the original one, it is simply added to the population and the population size increases.

Like sexual reproduction when population reaches its maximum value then we perform mutation by replacing the original individual with the mutated one if fitter.

2.7 Termination

The algorithm ends when a maximum number of iterations are reached or when the population dies out.

3 BENCHMARK : THE TRAVELLING SALESMAN PROBLEM

The Travelling Salesman Problem (TSP) is one of the most famous and studied problems. Its formulation is : "given *n* towns find the minimal tour such that each town, except the first, is visited exactly once" . It is known to be a NP hard problem and researchers [3][4] use this problem as benchmark for algorithms and approaches. So to test the capabilities of the dynamic population strategy we customized the algorithm to solve TSP and we compared the results with those obtained with the Simple Genetic Algorithm. In the following paragraphs we describe how the genetic features of the algorithm have been customized to solve TSP.

3.1 Coding

Enumerating the *n* towns with integer numbers then one possible solution is a permutation of *n* numbers. This represents the sequence to follow considering the first gene as the following of the last. In this way we define the length of the genotype as *n*, the number of towns, and we code the i-th gene as an integer *j*, between 1 and *n*, representing that at the i-th step town *j* is visited.

Example : 123456

$$\begin{tikzcd}
1 \arrow[r] & 2 \arrow[r] & 3 \arrow[d] \\
6 \arrow[u] & 5 \arrow[l] & 4 \arrow[l]
\end{tikzcd}$$

3.2 Initialization

Given as *n* the population size and *m* the number of towns then *n-m* individuals are initialized randomly while *m* are initialized according the simple greedy strategy : start from each of the different towns and then choose the nearest town not yet visited until all towns are visited.

3.3 Sexual Reproduction

The crossover operator is implemented in a very simple way. We take one, randomly chosen, crossover point *p*, with $0<p<n-1$, then two new individuals are generated. Each inherits the

genes from 0 to *p-1* directly from one of the two parents and then the remaining genes are filled with the towns not yet visited in the order of the other parent.

Example :

$$\begin{tikzcd}
12|3456 \arrow[drr] & \Rightarrow & 65|1234 \\
65|4321 \arrow[urr] & \Rightarrow & 12|6543
\end{tikzcd}$$

3.4 Asexual Reproduction

Because of the integer coding of the solution and to preserve the constrain that each town must be visited exactly once we simply implemented the mutation operator by swapping the alleles of two randomly chosen genes.

Example :

123**4**56 ⇒ **4**23**1**56

3.5 Fitness

As fitness function we adopted the sum of the Euclidean distance between the connected towns.

4 EXPERIMENTAL RESULTS

In the following part we report the experimental results we obtained by comparing the Simple GA and the Dynamic Population ES on TSP implementing, in both cases, the genetic features as described in the previous paragraph.

The experimentations have been curried out on three classical problems [3][26][27] : Oliver30 (30 towns), Eil51(50 towns) and Eil76(75 towns).

In the following part a few results are exposed. The first table shows the best known real and integer, in brackets, solutions.

Number of towns	Best known solutions
30 Towns	423.74 (420)
50 Towns	427.86 (425)
75 Towns	542.31 (535)

Tab 1. Best known solutions for TSP

The second table shows a comparison among the best results achieved by different strategies. The real solutions, when available, are the ones out of brackets. In the first column there are the results obtained by applying a simple greedy strategy of deep one, the second and the third columns show the outcome of our implementation, as described in the previous paragraph, of the Simple Genetic Algorithm and of the Dynamic Evolution Strategy. The fourth column exhibits the results of the Ant Colonies Systems [5][6][7], the fifth those of a GA specialized for TSP [29]. The results of the sixth column are obtained by applying Evolutionary Programming [30] and the ones of the last two columns are from [31] and are achieved by applying Simulated Annealing and a combination of this with Genetic Algorithms.

Number of towns	Greedy	Simple G.A.	Artificial Evol.	Ant Colonies	Enhan. G. A.	Evol. Progr.	Simul. Anneal.	S.A. + G.A.
30	473.32 (469)	425.94 (423)	424.69 (421)	**423.74 (420)**	N.A. (421)	**423.74 (420)**	N.A. (424)	N.A. (420)
50	505.77 (503)	443.98 (441)	431.17 (428)	427.96 (425)	N.A. (428)	**427.86 (425)**	N.A. (443)	N.A. (436)
75	612.65 (605)	568.95 (563)	564.7 (557)	**542.31 (535)**	N.A. (545)	549.18 (542)	N.A. (580)	N.A. (561)

Tab 2. Comparisons of the best results among different approaches

5 DISCUSSION

The introduction of dynamic population through the concepts of reproduction and competition has a few consequences.

As first result the classic step of selection through the fitness roulette wheel is no more needed because the concept of natural selection is directly implemented. Two observations about this can be done. The first is that this yields an improvement in system resources. In fact in classic GAs the selection step is performed through a O(*n*) complexity routine, where *n* is the population size, or through an intermediate population. With respect to the first case we have an improvement in computational time and with respect to the second case a O(*n*) memory saving is achieved. The second is that, because of the lack in the selection procedure, at each generation each individual is at least selected once, so everyone has at least one chance of reproducing. This gives a great improvement in population diversity.

Dynamic population, obviously, removes the constrain of population size. This is important because population size is one of the parameters which mainly affects the result of traditional genetic algorithms. In fact in the proposed approach population size is determined at each generation, starting from an initial small value, by the rate, resulting from the free competition and co-operation, between borne and dead individuals.

From the implementation point of view the proposed algorithm can be implemented by using dynamic data structures which are much more efficient than allocating, as usually done, static large arrays. Moreover in classic GAs, because of selection and parents'murders in the crossover step, two data structures are required, one for the current population and one for the new population. Dynamic population, implemented with a dynamic structure, with bisexual and mono-sexual strategies, allows to use only one data structure saving a O(*n*), where *n* is the population size, amount of memory. As example in the 50 towns problem for a classic genetic algorithm a population of about 3000 individuals is needed. Considering that the genotype is codified as an array of 50 integers then one individual requires at least $50 \times 4 = 200$ bytes and one population about 600Kb. In practice the larger the problem the larger is the memory saving using dynamic population with respect to classic GAs. This allows the population to grow to large population sizes.

The proposed approach has been compared with others on a classical benchmark, the travelling salesman problem on 30, 50 and 75 cities. From table 2 it is possible to notice that the strategy performs very well. In particular this strategy always performs better than greedy, simple genetic algorithm, simulated annealing and GA combined with the latter.

It is particulary interesting the comparison with the simple genetic algorithm because it has been curried out using the same implementation of the operators used for the dynamic strategy. It is clear the improvement of the solutions found, moreover a remarkable improvement of speed convergence towards good solutions has been noticed.

The solutions found are comparable also with the other considered techniques. In fact the worst result is that of 75 towns in which it has been achieved a result which is about 4% worse than the best one (Ant Colonies).

However this strategy has shown a problem. It is very important to find the right values of crossover and mutation probabilities to prevent premature convergence. In fact if crossover probability is too low, which means that dead rate is high, and the mutations are not enough high then the population will die out in few generations without finding a suitable solution. The right balance between crossover and mutation to prevent such a problem depends on the application.

6 CONCLUSION AND FUTURE WORK

In this paper an evolution of the Holland's Simple Genetic Algorithm is proposed with the aim of moving towards more realistic simulations of natural evolution. The key idea of the proposed strategy is the concept of *dynamic population*. It means that population, instead of being static, can grow and shrink in size during generations. To achieve this task the classic genetic operators, crossover and mutation, are seen as two different ways of generating new sons, without replacing their parents, and real competition among individuals is introduced.

From the efficiency point of view dynamic population allows, with respect to traditional genetic algorithms, a remarkable saving in memory requirement and an improvement in computational time.

This approach has been compared with many techniques on a classic benchmark : the 30, 50 and 75 towns travelling salesman problem. The experimental results have shown a great capability to find very good solutions, at most 4% worse than the best known solutions.

The results obtained, in terms of efficiency and capability to solve hard problems, are encouraging to look further and an outline of future work can be given. The proposed approach can be combined with other techniques, as simulated annealing or hill climbing, to move towards hybrid algorithms or can be specialized with smart reproduction strategies to solve particular problems. Maybe the most interesting line of research is that which stresses the simulation of natural evolution. For example sex may be introduced in the individuals and new co-operation and competition strategies can be developed. Moreover it is possible to go further and move towards artificial life by introducing the concepts of physical space, cellular automata and self-organization.

REFERENCES

1. M. Gorges-Schleuter, *On the power of evolutionary optimization at the example of ATSP and large TSP problems*, Proc. of European Conference on Artificial Life ’97, Brighton, U.K., 1997
2. M. Gorges-Schleuter, *Explicit Parallelism of Genetic algorithms through Population Structures*, Proc. of the Parallel Problem Solving from Nature Conference, 150-159, Springer-Verlag, Berlin, 1991
3. G. Reinelt, *TheTSPLIB*, http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html
4. P. Moscato, TSPBIB , http://www.densis.fee.unicamp.br/~moscato/TSPBIB_home.html

5. A. Colorni and M. Dorigo and V. Maniezzo, *Distributed Optimization by Ant Colonies*, Proc. of ECAL91 European Conference on Artificial Life, 134-142 , Elsevier Publishing, Paris, 1991.
6. A. Colorni and M. Dorigo and V. Maniezzo, *An investigation of some properties of an "Ant Algorithm"*, Proc. of the Parallel Problem Solving from Nature Conference, 509-520, Elsevier Publishing, Brussels, 1992.
7. A. Colorni and M. Dorigo and V. Maniezzo, *The Ant System : Optimization by a colony of cooperating agents*, IEEE Transaction on Systems, Man and Cybernetics part B, vol.26, No.1, 1-13, 1996.
8. V. M. Kureichick, V. V. Miagkikh, A. P. Topchy, *Some new features in genetic solution of the travelling salesman problem*, Proc. of ACEDC96, Plymouth, 1996 .
9. J. H. Holland, *Adaption in Natural and Artificial Systems*, MIT Press, Cambridge, MA, 1975
10. L. D. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991
11. D. Whitley and J. Kauth , *GENITOR : a Different Genetic Algorithm*, Proc. of the Rocky mountains Conference on Artificial Intelligence, Denver, CO, 118-130, 1988
12. D. Whitley , *The GENITOR Algorithm and Selective Pressure*, Proc. of the 3rd International Conference on Genetic Algorithms, Morgan-Kaufmann, 116-121, 1989
13. D. Whitley, *A Genetic Algorithm Tutorial*, Tech. Rep. CS-93-103, Colorado State University, 1993
14. L. Eshelman, *The CHC Adaptive Search Algorithm*, Foundations of Genetic Algorithms, G. Rawlins ed. , Morgan-Kaufmann, 256-283, 1991
15. W. Spears and K. DeJong, *An Analysis of Multi-Point Crossover*, Foundations of Genetic Algorithms, G. Rawlins ed. , Morgan-Kaufmann, 1991

16. G. Syswerda, *Uniform Crossover in Genetic Algorithms*, Proc. of the 3rd International Conference on Genetic Algorithms, Morgan-Kaufmann, 2-9, 1989

17. L. Booker, *Improving Search in Genetic Algorithms*, in Genetic Algorithms and Simulating Annealing, L. Davis ed., Morgan-Kaufmann, 61-73, 1987

18. H. Muhlenbein, *Evolution in time and space – the parallel genetic algorithm*, Foundations of Genetic Algorithms, G. Rawlins ed. , Morgan-Kaufmann, 316-337, 1991

19. H. Muhlenbein, *How genetic algorithms really work : I. Mutation and Hillclimbing*, Parallel Problem Solving from Nature -2- , R. Manner and B. Manderick eds. , North Holland, 1992

20. D. Goldberg, *A note on Boltzmann Tournament Selection for Genetic Algorithms and Population-Oriented Simulated Annealing*, TCGA Report 90003, Dept. Engineering Mechanics, University of Alabama, 1990

21. T. Starkweater and D. Whitley, *Genitor II : a Distributed Genetic Algorithm*, Journal Expt.Theor. Artif. Intell., 2:189-214, 1990

22. P. Bonissone, *Soft Computing Applications : the advent of Hybrid Systems*, Proc. of the Intern. Summer School on Fuzzy Logic Control : Advances in Methodology, 221-243, World Scientific Publishing, Ferrara (Italy), 1998

23. O. Cordon and F. Herrera, and M. Lozano, *A classified review on the combination fuzzy logic – genetic algorithms bibliography*, http://decsai.ugr.es/~herrera/fl-ga.html, Dept. Of Computer Science and AI, Univ. Of Grenada, Spain, 1995

24. J. D. Schaffer and D. Whitley and L. J. Eshelman, *Combinations of Genetic Algorithms and Neural Networks : a Survey of the State of the Art*, Proc. of Intern. Workshop on Combinations of Genetic Algorithms and Neural Networks COGANN92, 1-37, 1992

25. X. Yao, *A review of evolutionary artificial neural networks*, Tech. Rep. , Commonwealth Scientific and Industrial Research Organization, Victoria, Australia, 1992

26. I. Oliver and D. Smith and J. R. Holland, *A study of permutation crossover operators on the travelling salesman problem*, Proc. of the 2nd International Conference on Genetic Algorithms, J.J. Grefenstette ed., Lawrence Erlbaum, Hillsdale (NJ), 224-230, 1987
27. S. Eilon and C.T.D. Watson-Gandy and N. Christofides, *Distribution management : mathematical modeling and practical analysis*, Operational Research Quarterly, 20, 37-53, 1969
28. D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, 1989
29. D. Whitley and T. Starkweather and D. Fuquay, *Scheduling problems and travelling salesman : the genetic edge recombination operator*, Proc. of the 3rd International Conference on Genetic Algorithms, J.D. Shaffer ed. , San Mateo (CA), 133-140, 1989
30. D. Fogel, *Applying evolutionary programming to selected travelling salesman problems*, Cybernetics and Systems : an International Journal, 24, 27-36, 1993
31. F.T. Lin and C.Y. Kao and C.C. Hsu, Applying the genetic approach to simulated annealing in solving some NP-hard problems, IEEE Transactions on Systems, Man and Cybernetics, 23, 1752-1767, 1993
32. J.J. Grefenstette and R. Gopal and B.J. Rosmaita and D. Van Gucht, *Genetic Algorithms for the Travelling Salesman Problem*, Proc. of the an International Conference on Genetic Algorithms and their Applications,160-168, 1985
33. P. Moscato, *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts : Towards Memetic Algorithms*, Caltech Concurrent Computation Program, C3P Report 826, 1989
34. L. M. Rocha, *Evolutionary Systems and Artificial Life*, Tech. Report SSIE 580B, Los Alamos National Laboratory, Los Alamos, New Mexico, 1997
35. *Artificial Life OnLine* : http://alife.santafe.edu

36. C. G. Langton editor, *Artificial Life*, Proceedings of an interdisciplinary workshop on the synthesis and simulation of living systems, Addison Wesley, Los Alamos, New Mexico, 1987

37. D. Jefferson and C. Taylor, Artificial Life as a Tool for Biological Inquiry, in Artificial Life : an Overview, edited by C. G. Langton, MIT press, 1-10, 1995

38. E. Schrodinger, *What is Life? The Physical Aspect of the Living Cell*, Cambridge University Press, 1944