Toward a Parallel Genetic Algorithm Approach Based on Collective Intelligence for Combinatorial Optimizacion Problems

F. Hidrobo, Facultad de Ciencias, Univ. de Los Andes, Mérida, Venezuela
J. Aguilar, Facultad de Ingeniería, Univ. de los Andes, Mérida, Venezuela

Abstract

This paper addresses Collective Intelligence routes used in Artificial Life. Preliminary results are used to optimize the performance of Genetic Algorithms. Due to interested in the *behaviour oriented Artificial Intelligence*, specific attention is placed on the biological phenomena that reveals something about Collective Intelligence. Then, we propose a parallel reinforced search algorithm for the GA that use a collective memory of the better structural changes in the individuals through the generations, in order to use that information like trace of search.

1. Introduction

Artificial Life (AL) will have a tremendous impact on the future [2, 7, 9, 10]. Evolution of Artificial Systems is an important component of AL, providing an important modeling tool and an automated design method. *Genetic Algorithms (GAs)* are currently the most prominent and widely used models of evolution in Artificial Life Systems. GA is a method of search and optimization based on the theories of evolution of both Darwin and Mendell [3, 11]. GA proportions a group of important characteristics, as search based on population, combination of information and random mechanisms of decision. The efficiency of the GA is limiting when is applied to "big" problems or when we require good solutions, it is due because they don't secure get the better solution or because they consume considerable times in order to carry out the search of the better solutions.

In this article, we propose a scheme of reinforcement of the search that carries out the GA, as a manner of avoiding the previously mentioned problems. The proposal of reinforcement is based on the follow up of the structural evolution of individuals. In order to carry out the follow up, concepts and case studies of the domain of AL are used, particularly, the concept of *Collective Intelligence* [2, 10] and the case study of the *Colonies of Ants* [4, 5, 6]. This reinforcement approach is implicitly parallel, and is exploited in our implementation. Collective Intelligence studies such as those concerning the actions and inter-relations of a set of simple agents (for example, ants) can carry out the global objectives of the systems where these agents are immersed, with a high degree of efficiency. Each agent cooperates in carrying out tasks to complete those objectives, without central coordination or control, but with existing mechanisms of inter-relation and communication between them. Examples of these systems are the Systems of Insects (for example, the Colonies of Bees and of Ants [4, 5, 6]). In these systems, their agents are not individually intelligent, but their actions as a whole, in order to complete certain objectives of the systems, have demonstrate intelligent behavior (for example, search of sources of foods).

We use these ideas to propose a parallel reinforced search algorithm for the GA. This algorithm creates a *collective memory* of the better structural changes in the individuals through generations, in order to use such information as *trace of search*.

This work is organized as follows. The next section introduces the GA. Then, the theory on Collective Intelligence and Artificial Life are introduced. Subsequently, our approach of reinforcement is detailed, based on the model of Colonies of Ants. In section 5 the graph partitioning problem , the application of the reinforced GA to this problem, results analysis, and comparisons to other results obtained for previous work [8] are introduced. Section 6 presents the Traveling Salesman problem. Finally, the conclusions are presented.

2. Genetic Algorithms

This is an optimization algorithm based on the principles of evolution in biology. A GA follows an "intelligent evolution" process for individuals based on the utilization of evolution operators such as mutation, inversion, selection and crossover [1, 3, 11]. The idea is to find the best local optimum, starting from a set of initial solutions, by applying the evolution operators to successive solutions so as to generate new and better local minima. The procedure evolves until it remains

0-7803-4869-9/98 $10.00©1998 IEEE

trapped in a local minimum.

The use of the GA in the resolution of NP-Complete problems has been limited due to their prolonged execution time in searchs and due to the quality of their solutions (normally, they find optima local, not global). In order to solve these deficiencies directly, one can use parallel approaches [1, 8] and/or incorporate techniques which help in the search process of the GA [1].

2.1. Parallel Approaches for GA

The structure of the GA facilitates the application of different parallel approaches which can help decrease execution times of and improve the quality of the solutions. Two possible parallel approaches are [8]:

• *Divide the solutions space and apply simultaneously the GA in each sub-space.* In this approach, there are N instances of the same problem, where N is the number of space divisions. The objective here is to carry out a parallel search in the solutions space, to obtain several local optima of this manner. This approach should meet the following conditions:

- $S_i \subset S \; \forall i = 1, ..., K.$
- $S_i \cap S_j = \emptyset \; \forall i, j = 1, ..., K$ else $i = j$
- $S = \cup_1^K S_i$

• *Divide the internal structure of the GA in order to simultaneously executeall the possible operations.* The natural decomposition of the internal structure of a GA is in the basic phases: generation of the initial population, evaluation, reproduction, selection and replacement. This approach studies each phase individually. Accordingly, the implicitly parallel phases are the following:

- The generation of the individuals of the initial population.
- The evaluation of each individual.
- The reproduction of the individuals in order to create new individuals.

3. Artificial Life and Collective Intelligence.

AL undertakes those human-made systems that possess some of the key properties of natural life. A major motivation for the field of AL, besides the desire for a firmer theoretical developpment in biology, is the promise it holds for the synthesis of biological phenomena in forms that will be of great practical use in industrial and engineering endeavors [2, 7, 9, 10].

There are a number of open problems that seem especially good candidates which can benefit from the tools that AL is beginning to offer. We are specifically interested in artificial systems that serve as models of living systems for researching open questions in parallel systems. Particularly, we are interested in Collective Intelligence.

Collective Intelligence studies, such as collective cognitive capabilities in Systems of Insects, knowingly that the cognitive capability individual's is limited [2]. This cognitive of capability of a System of Insects allows it to complete objectives that insure its survival/vitality in a hostile environment, with a great efficiency. Examples of these systems are the ant, bee and wasp colonies.

4. Reinforcement of the search in the GA

We propose a reinforced search algorithm based on collective intelligence. The idea is the following: the solutions space is divided and a set of rules is used to define the information transmited between solution sub-spaces. The information to be transmited is composed of structural details of the best individuals found during the search in each one of the sub-spaces. So, the reinforcement approach we propose is based on the follow-up behavior of individuals, to register structural changes which occurred in each one of the solutions sub-space. To carry out these rules we were inspired by the Collective Intelligence, particularly, in the case of Ant Colonies.

Algorithms inspired by Ant Systems are heuristic methods that permit solving combinatorial optimization problems. The procedure is based on the distribution of the search on agents called "ants". That is, agents with very simple capabilities which trying to simulate the behavior of the ants.

Communication between agents (ants) is made through a trace, called **pheromone**. Such that a moving ant leaves a certain quantity of pheromone. Later, the probability that an ant follows a path increases according to the number of ants which have taken that path (a large quantity of pheromone in a path means a large probability it will be visited).

Using these ideas, we will distribute the search in sub-spaces, called agents or ants. In our case, each subspace executes a GA as described in the section II. Also, we follow the structural behavior of better individuals of every sub-population during their evolutions (that is, in each sub-space), through an individual which we call the **trace individual**. So, trace individuals carry the registration of good changes as if they were the **collective memory** of the system. In order to carry out reinforcement tasks, we define special individuals, called **reference individuals**, and use a new type of genetic operator (**trace operator**). We detail those aspects in the next sections.

4.1. Follow-up of the structural behavior of individuals

We propose to define a **trace individual** in each sub-solutions space, to keep the past information, with the following characteristics:

T_{11}	T_{12}	T_{13}	T_{14}	.	.	.	T_{1n}
T_{21}	T_{22}	T_{23}	T_{24}	.	.	.	T_{2n}
T_{p1}	T_{p2}	T_{p3}	T_{p4}	.	.	.	T_{pn}

The columns of the trace individuals represent the elements of each individual, while the rows are the different values that these elements can take. Each T_{jl} element will have a value defined by the following procedure:

- An individual i in time k has a function objective value equal to F_i(k), and in time $k+1$ equal to F_i(k+1).
- We define a improvement value expressed in the following manner:

$$F_t = (F_i(\mathrm{k+1}) - F_i(\mathrm{k}))/\mathrm{c}$$

where c is the total number of elements which have changed in the individual i. As we are interested in the quantity in which individual i has improved, depending on whether or not the objective is to minimize or maximize, the changes are considered interesting if F_t is positive and the objective is maximize, or vice versa if the objective is minimize.
- The value of T_{jl} is incremented in F_t if the element l of individual i has changed to the value j.

Using the mathematical function (F_t) the trace individuals register the contribution of well reproduced individuals (and of each one of their elements) in reference to individuals that they replaced in the original population.

The fundamental concept that is used in the definition of the mathematical function mentioned previously is the same as the main idea that exists in the systems of ants, that is, **pheromone**. This function represents the solutions improvement value through the generations.

4.2. Information Transmission Rules (partial results)

One of the main aspects is how to use the information kept in the trace individuals in order to guide the search process for the paths that each sub-space has determined as interesting. The objective is try to use the information of the trace individuals as an element that guides the search procedure towards better regions than the previous. As such, we propose two mechanisms or rules:

- Generation of **reference individuals** using the information of trace individuals. The reference individuals are composed of the better values identified in trace individuals; this identification uses a probability scheme based on values recorded in the trace individuals. Then, these reference individuals are used to apply genetic operators on them. Then, every GA selects individuals for the reproduction phase from the reference individuals and the best individuals of its sup-space.
- Definition of a new genetic operator, called **trace operator**. This is a modification of some of the operators we are using. To choose the operator to modify, we select one that permits a jump in the solutions space (for example, mutation or inversion). This operator will allow to the values of the individuals elements change toward the better values identified in the trace individuals. The best values are these that had better behavior (larger T_{jl} values).

4.3. Migration of the new individuals

We can define one migration form of the new generated individuals, which depend on the way we have divided the solutions space (see section II): migration of the individual to the partition of the solutions space that it belongs to.

4.4. Reinforced Macro-Algorithm

Incorporating the reinforcement scheme, we propose the following macro-algorithm:

1. Code the individuals which represent the solutions
2. Divide the solutions space using approaches defined in section 2.1
3. Apply the GA in each sub-space. This GA will have memory capabilities (through trace individuals) in order to determine the best characteristics (patterns of common code) that through the generations the individuals have.
4. using the trace individuals, evaluate and reproduce new individuals. In this case, the reproduction phase also establishes what information to transmit between different sub-spaces, using new elements such as the trace operator and reference individuals.
5. Return to step 3 until the system converges.

5. The Graph Partitioning Problem

The problem consists in dividing a graph in several subgraphs [1], so as to minimize the connection costs between them. In a very general way, in order to mathematically formulate the problem, the following definition is necessary: the graphs are sets of nodes joined by arcs. It can be defined as follows:

$$G = (\ N,\ A,),$$

- G is a directed graph,
- $N = \{1, ..., n\}$ is a set of n nodes on which we can associate a weight function $Q : N- > R$. In our studies Q(i)=1 for i=1,.., n,
- $A = \{a_{ij}\}$, are node pairs that define the arcs. It is known as the adjacency matrix.

According to certain constraints, the problem consists dividing the graph in K different subgraphs. The classic constraints are:

• The subgraphs must have a specific size or must have a weight sum of nodes less than a given value.

• The arcs with extremities in different subgraphs must be minimal, or the weight sum of arcs which join nodes in different subgraphs must be minimized.

The cost function associates a real value to every subgraph configuration. We propose the next cost function:

$$F_C = \sum_{i,j \in D} a_{ij} + b\frac{\sum_{z=1}^{K} (N_{G_z} - n/K)^2}{K} \qquad (1)$$

$D = \{i \in G_m\ \&\ j \in G_l\ \&\ l \neq m\ \&\ a_{ij} = 1\}$.
N_{G_z}= number of nodes in subgraph z
b = balance factor $[0, 2]$.
The graph partitioning problem is reduced to find a subgraph configuration with minimum value for the cost function:

$$\mathrm{F} = \mathrm{MIN}(F_C)$$

5.1. Solution of this problem by means of GA

The GA applied in our problem follows the next procedure: we define a research space of n vectors where everyone represents an individual, and every individual represents a possible solution. Each vector has n elements (every element is a node in the graph) and according to the group (sub-graph) to which it belongs (it is assigned) every element has a value between 1...K, . For example, if we assume N = 5 and K= 2 and we could have an individual with the next values: 2, 1, 1, 2, 1. That means for this solution node 1 is in subgraph 2, node 2 is in subgraph 1, node 3 is in subgraph 1 and so on. Furthermore, we use the cost function defined on the first part to determine the cost of every individual. We begin with an initial population of individuals randomly defined and we choose the individuals with minimal cost for generating new individuals using genetic operators. Since the population is constant, we substitute the worst individuals of the initial solution with the best individuals generated. The procedure stops if we exceed a given number of generations without finding a better solution. We use the crossover and mutation operators as genetic operators. The more important implementation details used to solve this problem with our approach are the following:

• The partitioning approach that we use randomly chooses a vector position that defines an individual (for example, the third element) and assigns a different value for this element in each sub-space (between 1 y K).

• For the reinforced Genetic Algorithm:

- The trace individual saves the best values or partitions (between 1 and K) where we must assign every node.

- The **reference individuals** are used in order to match them with better individuals from each sub-population (that is, of each sub-space) using the crossover operations. So, in each crossover operation a reference individual is chosen for crossing with some individual in any GA sub-populations.

- The **trace individual** is a modification of the mutation operator. This operator will allow the performance of a mutation which changes the element values of the individuals towards better values identified in trace individuals. The best values are those that had better behavior (larger value of T_{jl}).

5.2. Results Analysis

The implementation is carried out using C and the PVM (Parallel Virtual Machine) library. The platform used is a eight processors IBM-SP2 of the National Computing Center of the Universidad de los Andes.

The results were compared with previous results obtained in [8], where two parallel GA and a serial GA were presented. One of the parallel GA exploits the implicit parallelism of this technique and the other divides the solutions space in order to perform a search in each one using a GA. Next, we present the results tables for different values of N and K.

• AGS= Serial Implementation.

• AGP= Parallel program using the partitions approach.

• AGPR= Parallel Implementation using the reinforced GA approach.

1. N=30 K=5

ALG	MIN	MAX	PROM	t (sec)
AGS	129	136	131	0.400
AGP	124	126	124.6	1.294
AGPR	124	127	125.8	9.830

2. N=50 K=7

ALG	MIN	MAX	PROM	t (sec)
AGS	271	278	272.2	3.840
AGP	263	268	266	11.760
AGPR	262	266	264.4	50.910

3. N=70 K=9

ALG	MIN	MAX	PROM	t (sec)
AGS	756	765	760.8	14.300
AGP	731	741	737	37.680
AGPR	726	745	740.5	160.750

4. K=5

ALG	N=20	N=50	N=100	N=200
AGS	59	250	1540	1787
AGP	51	246	1520	1772
AGPR	51	244	1517	1772

5. N=100

ALG	K=3	K=4	K=5	K=6	k=7
AGS	1268	1430	1535	1601	1638
AGP	1267	1426	1525	1586	1624
AGPR	1265	1416	1520	1580	1616

We can observe that the execution time of the parallel programs (AGP and AGPR) is incremented due to the tool used (PVM), based on the approach of messages passing. This introduces a new time, well-known as communication time that could be caused for: the number of communications of the processes, the problems of latency of PVM. Besides, the AGRP program always has more execution time than AGP, this additional time is produced for the manipulation that should make AGPR of the trace individuals (Rules of transmission of the information). The main point is the quality of the results, so, as much AGP as AGPR find better solutions that AGS. Table 5 shows that AGPR finds a better solution than AGP. In the other tables these improvements are smaller. The contribution of the parallel implementations in the quality of the solutions is because they can perform a more exhaustive and rigorous search.

6. The Traveling Salesman problem

The Traveling Salesman problem is a classical optimization problem that could be described as: given N cities, the salesman should visit each city one time and the total cost of the journey should be minimal. We can define the cost of the journey as the sum of the distances between the visited cities. This problem can be expressed in the following manner:

$$G = (N, A),$$

- $N = \{1, ..., n\}$ is the graph with n nodes,
- $A = \{a_{ij}\}$ is the adjacency matrix.

We can define D, matrix of distances as:

$$\{d_{ij}\} = \begin{cases} \infty & \text{Si } a_{ij} = 0 \\ l_{ij} & \text{Si } a_{ij} = 1 \end{cases}$$

- l_{ij} = distance between the cities i and j

If we suppose that the cities are numbered from 1 to N, a solution to the problem could be expressed through a state matrix (E) that indicates the order cities are visited. This matrix will contains rows with the order cities visited, and in the columns the cities.

$$\{e_{ij}\} = \begin{cases} 1 & \text{if city j was visited in the position i} \\ 0 & \text{Otherwise} \end{cases}$$

That is, matrix E defines array V with n elements which contains the city that was visited in each position.

$v_j = i$ (If city j was visited in the position i)

Finally, we propose a function that calculates the distance between the cities. This function is:

$$FC = \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{j=1}^{n} l_{ik} e_{ij} e_{kj+1} \qquad (2)$$

The problem consists of finding the journey for the cities that minimize the value of the cost function FC.

6.1. Solution of this problem using GA

In this section, we present what we need to considered to solve this problem using GA. With the previous cost function we use only the inversion and translocation operators as genetic operators,this way all the solutions generated using valid solutions are valid The validity of the initial solutions should be verified in the generation process.

For this problem a solution is a journey that fulfills the restrictions of the problem. Therefore, the representation of the individuals (solutions) is carried out by means of a vector of N elements. This vector contains the orderly journey, that is, position 1 contains the first city visited, the position 2 the second, and so forth. In order to use this representation, we suppose that the cities numbered from 1 to N. The genetic operators for this problem carry out two functions: the genetic evolution and the generation of valid solutions. The operators used are:

- *Inversion.* In this case, we chosen randomly a position (j) (with a value of city c1) and generate randomly a new value for the city (c2) that will be visited in this position. Before, we search the current position of c2 in the vector (position i) and we assign the value c1 at this position.
- *Translocation.* With this operator, we move the elements of the vector. We use a circulate vector to represent the solutions and we move the values of the cities which are visited.

The implementation of the parallel approaches to solve this problem is the same as in the previous section. The main differences are the following:

- The partition scheme for this problem takes a position

of the vector randomly (for example, fourth position) and assigns to each sub-space a different value from the city that will be visited in that position.

• For the reinforced Genetic Algorithm:

- The *trace individuals* contain the best cities for every position.

- The *trace operator* is a modification of the inversion operator. This operator selects randomly a position (l) and changes the value of the position to the best value identified in the trace individuals for this position (larger values of T_{jl} for this position). Then, we search the current position of T_{jl} in the vector (position i) and we assign the current value of the position l at this position.

- The **reference individuals** are selected in order to apply the inversion and translocation operators. Then, these new individuals are sent at the different S_i.

6.2. Results Analysis

The experimental procedure for this problem is the same as that we have used for the graph partitioning problem.

1. Results for 20 cities

ALG	MIN	MAX	AVE	t (sec)
AGS	135	217	174.867	0.120
AGP	130	180	152.500	0.903
AGPR	123	164	147.567	4.382

2. Results for 100 cities

ALG	MIN	MAX	AVE	t (sec)
AGS	603	688	645.200	92.239
AGP	586	640	613.150	108.971
AGPR	583	655	614.400	435.139

3. Results for 200 cities

ALG	MIN	MAX	AVE	t (sec)
AGS	1627	1992	1892.500	469.545
AGP	1539	1931	1654.600	322.120
AGPR	1528	1662	1611.400	1702.155

4. Results for 500 cities

ALG	MIN	MAX	AVE	t (sec)
AGS	5184	5184	5184	1408.570
AGP	4818	4962	4912.700	1108.549
AGPR	4781	5184	4872.909	61093.642

The results obtained for this problem are very similar to those obtained for the previous problem. In this case, the results of parallel approaches (AGP and AGPR), are better than for the serial approach (AGS). For all cases, with the exception of these introduced in the Table 2, the reinforced parallel approach (AGPR) finds better solutions than the parallel version (AGP). On the other hand, with respect to the execution time, we observe that AGPR is always slower than AGP. However, the execution time of AGP is not always slower than AGS (Tables 3 and 4). The improvement time in the parallel approach could be attributed to the size of the problem, which is due to the increase in the granularity of the processes (Tasks).

7. Conclusions

The utilization of a library of message passing (PVM) for the implementation of the parallel approaches of the GA to solve combinatorial problems does not improve the time of execution. It is probable that the reduction in the time of execution can be reached in parallel systems with shared memory or using a library of message passing more efficiently. The contribution of the parallel implementations lies in the quality of the solutions, due to their more rigorous and exhaustive search in the global space.

The approach of reinforced GA indeed allows recording of the history of the evolution of the individuals in the GA. Also, it exploits this information in the phase of reproduction using several mechanisms (generation of reference individuals and trace operators), this permits improvement of the quality of the solution. The deficiency that is observed in the AGPR is in its execution time. The cost of communication is very important in this parallel approach.

References

[1] J. Aguilar, *L'Allocation de Tâches, l'Équilibrage de Charge et l'Optimisation Combinatoire.*, PhD thesis. Université René Descartes - Paris V, 1995.

[2] J. Bonabeau and G. Thereulez, (Editors) *Intelligence Collective,* Hermes, Paris, France, 1994.

[3] L. Davis *Handbook of genetic Algorithms* Van. No strand Reinhold, New York, 1991.

[4] M. Dorigo and L. M. Gambardella *ANT-Q: A Cooperative Learning Approach to Combinatorial Optimization,* Technical Report 95-01. Université Libre Bruxelles, Bruxelles, Belgium, 1995.

[5] M. Dorigo, V. Maniezzo, and A. Colorni *The Ant System: Optimization by colony of cooperating agents* IEEE Transactions on Systems, Man, and Cybernetics-Part B, Vol 26 , No 1, 1996, pp. 1-13.

[6] M. Dorigo, V. Maniezzo, and A. Colorni *The Ant System: Optimization by colony of cooperating agents* IEEE Transactions on Systems, Man, and Cybernetics-Part B, Vol 26 , No 1, 1996, pp. 1-13.

[7] J. Fernandez and A. Moreno *La Vie Artificielle* Seuil, Paris, France, 1995.

[8] F. Hidrobo and J. Aguilar *Esquemas de Paralelización de Algoritmos Genéticos en la Resolución de Problemas de Optimización Combinatoria,* Computación Paralela y Distribuida: Hardware y Software (C. Paéz, J. Aguilar, G. Paéz eds.), CEMISID, Mérida, Venezuela, 1997, pp 39-46.

[9] C. Langton *Artificial Life,* MIT Press, London, England, 1995.

[10] S. Russell and P. Norvig *Artificial Intelligence: A modern approach,* Prentice Hall, New York, USA, 1995.

[11] M. Srinivas and M. Patnaik *Genetic Algorithms: A Survey,* IEEE Computer, June 1994 pp. 17-26.