Proceedings of the 6th WSEAS Int. Conf. on EVOLUTIONARY COMPUTING, Lisbon, Portugal, June 16-18, 2005 (pp111-116)

Distributed Population Based Incremental Learning

Rui Rodrigues
Departamento de Electrotecnia, Faculdade Ciências e Tecnologia,
Universidade Nova de Lisboa,
Quinta da Torre, 2829 – 516 Caparica
Portugal

Abstract - Probabilistic model building genetic algorithms (PMBGA) are powerful search techniques that are used successfully to solve hard computational problems. We exploit the natural bridge island concept in the Parallel PMBGA context. Bridge Island is an island that all the best elements arriving from evolving islands have to pass before its migration to other islands occurs. Some ways of best elements selection to migrate and also some ways of merging information between the migrating element and the receiver element will be tested. The main objective is to compare the relative performance between distributed PBIL and isolated PBIL. The parallel models that will be used are the master-slave topology, island model and coarse-grained.

Key-Words – Evolutionary algorithms, Distribute algorithms, PMBGA, PBIL, Selecting probabilistic information and Merging probabilistic information.

1 Introduction

Genetic algorithms (GA) are stochastic search algorithms based on the Darwin's theories of how the species adapt to environment. Genetic algorithms [1, 2] use selection, crossover and mutation to simulate the way nature evolves, as a model of evolving a population of solutions, giving more chances to the fittest individuals to survive and pass his genetic code to a further generation. The PMBGA is a new type of algorithm based on genetic algorithms. In this kind of algorithm it is not necessary to maintain the population; the algorithm tries to use some statistical information of the best individuals in the population to evolve and generate a new individuals. Many PMBGA exist, as can be seen in the survey of PMBGA [3], and one of the earliest works was the PBIL [4], which will be used in this paper.

Trying to use parallel computation (or distributed in a network of computers) is an opportunity to use a new kind of divide and conquer algorithm and a way to efficiently use computer resources. The field of parallel genetic algorithms is a very dynamic field as is referred in the Survey of Erick Cantú-Paz [5]. This paper presents a new approach to parallel the PBIL; the results can be extrapolated to some other types of PBMGA algorithms. The results obtained from [8, 9] using a parallel version of the PBIL are a motivation to this work.

The bridge island will be an island which that all the best solutions incoming from evolving islands have to pass before going to other islands. Some ways of selecting and merging information will be tested. The bridge island strategy will be compared with an isolate standard PBIL. The parallel

implementation will use the coarse-grained island model.

2 PBIL – A quick introduction

The PBIL [4,8] is one of the earliest PMBGA [3]. The PMBGA uses an explicit translation of some proprieties of the best individuals of the population into a probability vector. This vector will be used to generate new candidates in opposing to GAs that use the entire population. A general PMBGA [3] can be organized as follows:

1. Generate initial population of size M
2. Select N promising solutions where N<=M
3. Calculate joint probability distribution of selected individuals
4. Generate offspring according to the calculated probability distribution and replace parent
5. Go to step 2 until termination criterion is satisfied.

The PBIL uses this algorithm but uses the Univariate Model – it does not consider any dependency among variables, it considers building blocks of order one.

The PBIL algorithm used in this paper can be organized as follows:

1. Initialize probability vector $p=\{p_1, p_2,\ldots, p_n\}$, with 0.5 at each position. Each variable represents probability of value '1' (in binary codification) being present in the same position of the child chromosomes.
2. Sample M>>0 individuals according to probabilities in P and evaluate them.
3. Update probability vector according to fittest individuals $S=\{s_1,s_2,\ldots s_n\}$
   using the following rule:

   $$p_i = p_i \times (1-LR) + s_i \times LR$$

4. Update probability vector away[1] from the worst individuals $B=\{b_1,b_2,\ldots b_n\}$
   using the following rule:

   If $b_i=0$ ➔ $p_i = p_i \times (1-NLR) + NLR \times 1$

   If $b_i=1$ ➔ $p_i = p_i \times (1-NLR) + NLR \times 0$

5. If mutation condition did pass, mutate Probability vector using the
   following rule:

   $p_i = p_i \times (1-MS) + random(0 \text{ or } 1) \times MS$, where MS is the amount to affect the probability vector

6. To maintain some diversity in the generation of new solutions the values of pi are restrict to pi >0.01 and pi <0.99
7. Go to Step 2 until termination criterion is satisfied.

More about the PBIL can be found in [4,8].

3. Distributed PBIL proposed

We can see multiple-deme parallel genetic algorithms as a set of isolated demes organized in a certain topology, where each deme – or island in this paper– can choose a number of chromosomes to migrate to other demes with a certain migration frequency. In the island model, the individuals can migrate to any other deme. Like the multiple-deme parallel genetic algorithms there will be various PBIL algorithms working independently of each other with its probabilistic vector P. In the case of PBIL the probabilistic vector expresses some information of the best past solutions of the deme population. It will be this information that will migrate to other demes trying to influence them.

In the dynamic topology the elements are not restricted to migrate to a fixed number of demes, but instead we can have some rules to choose which will migrate to where. In the work of Lin et al. [6] they measure the distance between population genotypes to help the decision-making and in the work of Marin, Trelles-Salazar, and

[1] Similar idea from the original

Sandoval [7], they gather the best solutions of all demes and choose the best individuals found so far and broadcast them to the slaves' demes. The main difference is that in this paper we focus on new ways to use the bridge island in the context of PBIL.

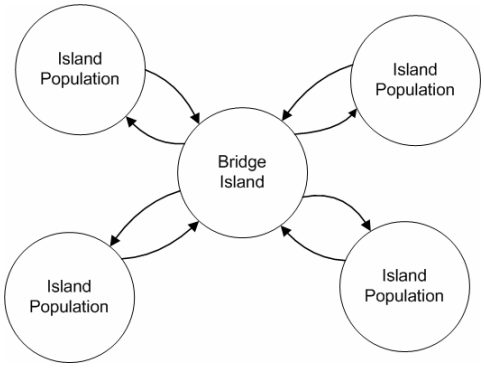

Fig. 1. Bridge Island model

3.1 Selecting Information

In the Distributed version of the PBIL (DPBIL) we will have one deme working with an isolated population that will be affected from time to time by the migration of other elements migrating from other islands.
The bridge model is like the master-slave architecture, but in this model the master will have a dynamic new role in the evolution of the entire population. We can view the bridge island as a place to select who will migrate to where, we will use four strategies:

- **Diversity Island (D)** - The probabilistic vector to migrate to a particular island is the most different element in the sub-population of the bridge island, comparing to the probabilistic vector in the destination island. We will use the distance between elements to evaluate the difference;
- **Randomly(R)** – The selection choice is made randomly, without using any criterion;
- **Tournament Island (T)** –The element received from an island will compete in a tournament to be able to go to another island, the tournament begins only when the players arrive at the bridge island. The element allowed to travel to a new island is chosen like the tournament selection operator in the genetic algorithms – choose two random elements and the best can migrate to a new island. The evaluation of the probabilistic vector is the evaluation of the best most recent chromosome – in the last generation – that the probabilistic vector had generated;
- **Proportional selection (P)** – Another way is to give more chances to the best ones to migrate using its evaluation to know what will be the probability of success of a particular vector.

3.2 Merging Information

There will be many possible ways to influence the deme probabilistic vectors after the migration had occurred:

- **Simple learning (SL)** – Uses a parameter – migration learning – that can transfer part of the information of the migrating probabilistic vector to the probabilistic vector of the island.

(1)
$$P_{island}[i] = (1-\alpha)P_{island}[i] + \alpha P_{migrating}[i],$$
α migration learning

- Heavy learning (HL) – the migration learning parameter will be defined between the relative forces of each probabilistic vector. The force is obtained by each evaluation of the best chromosome from the last generation.

(2)
$$\alpha_1 = Eval[P_{migrating}]/(Eval[P_{island}] + Eval[P_{migrating}])$$
$$\alpha_2 = Eval[P_{island}]/(Eval[P_{island}] + Eval[P_{migrating}])$$
$$P_{island}[i] = \alpha_2 P_{island}[i] + \alpha_1 P_{migrating}[i]$$

- **1-point crossover (C1).** – We will use the crossover operator between the migrating probabil-

istic vector and the probabilistic vector of the island (of course other ways exist). This last operator is the same as the crossover operator in a GA. In this paper only the one-point crossover operator will be used;

- **Heavy 1-point crossover (H1C)** – The probabilistic information that will contribute more is the probabilistic vector that has the best chromosome associated. The percentage that each probabilistic vector contributes to the crossover depends on the relative weight of each probabilistic vector (evaluation of the best chromosome[2]) – the same idea as the heavy learning.

4. Problem

This paper addresses one problem that is a deception problem – Trap Function. We have a trap function that has the global maximum in center of the domain. We will increase the difficulty of the problem, decreasing the influence of the main triangle (with the perc_D parameter). The influence of central triangle will be tested with values very small like 0,1% of the search space.

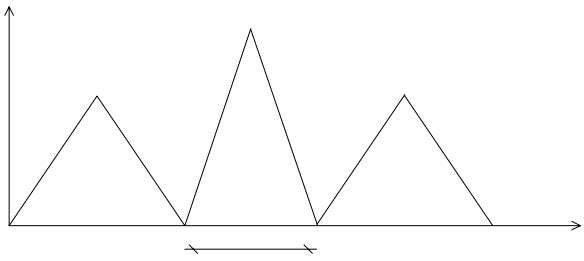

Fig. 2. Evaluation Function

We have some parameters that will characterize the functions:

- Gmax is the value of global maximum
- Lmax is the value of local maximum (the biggest)
- Perc_D is the Percentage of influence of the triangle with maximum in the domain
- L is the dimension of the chromosome (solution)
- D is themajor $x=2^L$ allowed of the function
- N is the number of low triangles

Lmax[i]= Lmax/(1+0.01×(i-1)), i=number of the local maximum counting from the center 1 (more near the central) to N/2.
The codification of the solution uses the gray code, to be easier to the algorithm to evolve.

5 Results

The parameters used in the experiences are showed in the following tables. In table 1 we show the parameters used in the isolated algorithm and in the distributed algorithm. In table 2 we have the parameters of function to be used.

[2] Maybe the best is to use as evaluation of the Probabilistic vector the average evaluations of the chromosomes that were used to construct the probabilistic vector.

Table 1 Parameters used in the experiences

	Isolated PBIL	DPBIL
Population (in each island)	800	200
Number of generations	60	60
Number of best solutions selected	2	2
Learning Rate	0.1	0.1
Number of worst solutions selected	1	1
Negative Learning Rate	0.075	0.075
Probability of mutation	0.02	0.02
Mutating shifted	0.05	0.05
Migration rate	-	16,6%
Migration Learning	-	0.1

Table 2 Function parameters

L(dimension of chromosome)	200
Global maximum	7
N	4
Local maximum[1]	6
Local maximum[2]	5.94
D	2^{200}

We will test the function with Perc_D changing from 0.1% to 0.005%. All the cases tested had 20 restarts to enable some statistical analysis. We will show the average best results and number of times that the best evaluation was best than 6, obtained for all combinations of selection and merging information applied to the problem with the decreasing of influence of the major triangle in the domain of the problem.
Next we will show the graphics related with table 3

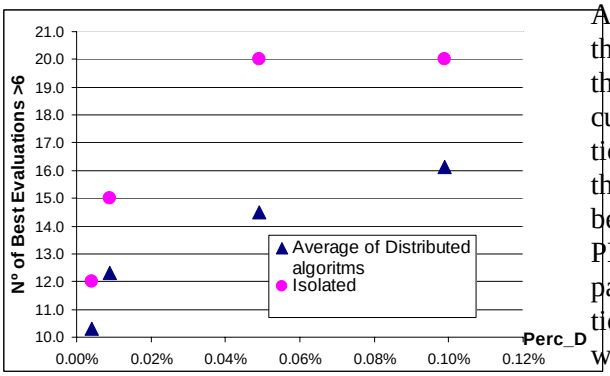

Fig. 3. Evolution of Number of Best Evaluations >6

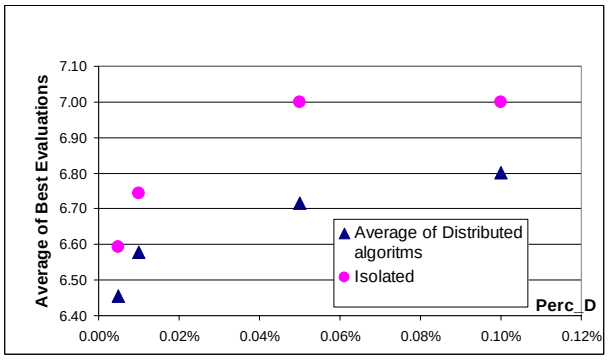

Fig. 4. Evolution of Average of Best Evaluations

Table 3 Average of best evaluation

Perc_D	0.1%	0.05%	0.01%	0.005%
Sel/Merg	Average	Average	Average	Average
R_SL	6.69	6.94	6.51	6.52
R_HL	6.70	6.85	6.54	6.44
R_C1	6.94	6.74	6.45	6.37
R_H1C	6.70	6.77	6.55	6.27
D_SL	6.85	6.70	6.71	6.31
D_HL	6.85	6.64	6.56	6.56
D_C1	6.80	6.80	6.64	6.52
D_H1C	6.98	6.67	6.43	6.70
T_SL	6.80	6.74	6.49	6.56
T_HL	6.80	6.69	6.57	6.34
T_C1	6.75	6.59	6.72	6.58
T_H1C	6.85	6.73	6.56	6.33
P_SL	6.85	6.60	6.58	6.57
P_HL	6.80	6.80	6.58	6.20
P_C1	6.79	6.65	6.73	6.65
P_H1C	6.70	6.55	6.61	6.38
Average	6.80	6.72	6.58	6.46
Isolated Islands	7	7	6.74	6.59

As we can observe in the Figure 3 and 4 that the performance of DPBIL is worst than the isolated counterpart in less difficult problem but we can see by the evolution curve that the tendency is to approach the results of both strategies as the problem becomes harder. The effect of a Distributed PBIL seems - in this problem - that can not pass the positive effects of a large population, but seems that is a valid option when we have distributed computational power available. We can not conclude, as we can see in tables 3 and 4, what of the proposed forms of selecting information and merging

information are the best, the results do not allow that kind of conclusions. But we can see in Table 4 – average all best results form all the problems – that in this problem the diversity selection and the 1-point crossover were the best options.

Table 4 - Average all best results form all the problems

Type of selection		Type of Merging	
R	6.626	SL	6.649
D	6.669	HL	6.619
T	6.631	C1	6.670
P	6.626	H1C	6.611

6 Conclusions

We presented various forms of selecting information and merging information between PBIL algorithms. This work was an effort to solve the problem of finding dynamic topologies that allow the mixing of probabilistic vectors. We can see that, in the case of the PBIL, for particular problem, it is possible to obtain good results using the distributed typology. In the future we can try to find better ways of merging information more related with the probabilistic vector that is the heart of the algorithm. One way to reduce the problem of being trapped into local optima is using the proposed topology. It is still necessary more work to completely validate this way of distributing a PBIL.

References:

[1] Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, 1975

[2] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, 1989

[3] Shakya, Siddhartha, K.: Probabilistic model building Genetic Algorithm (PMBGA): A survey. Technical Report. School of computing, The Robert Gorgon University, 2003

[4] Baluja, S.: Population-based incremental learning. A method for integrating genetic search based function optimization and competitive learning. Pittsburgh, PA: Carnegie Mellon University, Technical Report No. CMU-CS-94-163, 1994

[5] Cantú-Paz, E.: A survey of parallel genetic algorithms. Calculaters Paralleles 10, 1998

[6] Lin, S. C., Punch, W., Goodman: Coarse-diversity Grain Parallel Genetic Algorithms: Categorization and New Approach. In Sixth IEEE Symposium on Parallel and Distributed Processing, IEEE Computer Society Press (Los Alamitos, CA), 1994

[7] Marin, F. J., Trelles-Salazar, O., Sandoval, F.: Genetic algorithms on LAN-message passing architectures using PVM: Application to the rooting problem. In Davidor Y.,Schwefel H. P., Männer, R., Eds., Parallel Problem Solving from Nature, PPSN III, p. 534-543, Springer-Verlang (Berlin), 1994

[8] Baluja, S., Advances in Neural Information Processing Systems (NIPS '96), December, 1996

[9] Baluja, S., David, A. S., Evolution-Based Methods for Selecting Point Data for Object Localization: Applications to Computer Assisted Surgery, CMU-CS-96-183, September, 1996