Comparison of Learning Rules for Adaptive Population-Based Incremental Learning Algorithms

F. Bolanos[1], J. E. Aedo[2], and F. Rivera[3]
[1]School of Mecatronics, Universidad Nacional de Colombia - ARTICA, Medellin, Colombia
[2]Electronics Department, Universidad de Antioquia - ARTICA, Medellin, Colombia
[3]Computer Science Department, Universidad de Antioquia - ARTICA, Medellin, Colombia

Abstract— *This paper describes the adaptive approach of the Population-based Incremental Learning (PBIL) algorithm, and proposes several Learning Rules aimed to improve its performance. The assessment of such alternatives was made in terms of both the convergence time and of the quality of the achieved solutions. Two classical optimization problems were used for the tests: The Job Shop Scheduling problem and the Traveling Salesman problem. The obtained results are very promising and suggest that some of the proposed learning rules have a superior performance, without degrading drastically the quality of the solutions.*

Keywords: Optimization, Adaptive algorithms, Incremental Learning, Learning Rules

1. Introduction

The use of population-based algorithms has been successful in solving highly complex optimization problems, as reported on literature [1], [2]. The success of this kind of optimization tools refers to their ability to use a population of potential solutions, to perform a parallel exploration. This parallel approach helps to avoid the local-optimum problem.

Among the different kinds of population-based approaches, Evolutionary Algorithms (EAs) are the most often used. A given EA works with a population of individuals and some rules for changing those individuals in order to reach acceptable solutions. Those rules are inspired on evolutionary processes of biological beings. Multi-Objective Evolutionary Algorithms (MOEAs) are particularly useful when optimizing several figures of merit is required, since such figures of merit are often in conflict with each other [3], [4].

PBIL optimization algorithms work also with a population of potential solutions. The main differences with respect to EAs are related with the representation and the updating process of the population [5]. In PBIL, population is represented by means of a probability array. Each probability value in the array, is related to whether a given attribute must be part or not of the final solution. The way in which the probabilities of the PBIL array are updated, in order to find an optimal, is referred as Learning Process. The updating of each probability in the PBIL array is often performed by means of an approach based on the Hebbian rule [6]. Such updating depends on a parameter called Learning Rate (LR), which controls the speed of the convergence process.

This paper describes several variations of an Adaptive Population-Based Incremental Learning (APBIL) algorithm, which adjust the learning rate parameter dynamically, in order to speeding up the convergence time. The idea is to test several learning rules (i.e. the way in which the learning rate must be modified) in order to find the best trade-off between speed of convergence and quality of the solutions. The propossed learning rules are compared with that one presented in [7], where a bell shape is suggested to change the learning rate as a function of the probability array's entropy. Although optimization algorithms may behave different when used to solve different optimization problems [8], this work provides some insight into the APBIL performance and highlights its main advantages.

This paper is organized as follows. Section II shows some previous works related to PBIL optimization algorithms. Section III shows some fundamentals about the adaptive PBIL algorithm and the learning rules proposed to improve its performance. Section IV describes the Job Shop Scheduling problem, which will be used to test the PBIL optimization approaches. Section V shows the comparison of the results provided by different approaches in terms of speed of convergence and quality of the solutions. Finally, section VI shows the concluding remarks and future work.

2. Related Work

Several works have proposed the use of PBIL for solving optimization problems [9], [10], [11]. In [10], the effect of the LR parameter on the algorithm's performance is analyzed. The design of a Power System Stabilizer (PSS) is used as case of study. This work shows that smaller learning rates, results in a high diversity of the population of solutions. Such a diversity implies a slower convergence time. On the other hand, when LR increases, the exploitation (i.e. improving and intensifying the features of the best solutions found so far) of the design space is favored and the convergence speed grows, but also may lead to find local optimals instead of the global one.

In [12], a comparison between Breeding Genetic Algorithm (BGA) and PBIL approach is performed, for the

design of a Power System Controller. BGA is a modern version of traditional evolutionary algorithms which uses the survival of the fittest as optimization mechanism, but allows a mechanism similar to artificial insemination on living beings, where the offspring may take the best attributes of its parents. Results of this work show that PBIL has almost the same performance when optimizing the power controller. The main advantage of PBIL is that it is computationally simpler and has fewer genetic operators when compared with BGA. Results also show that PBIL algorithm requires less memory and computational resources, which makes it very suitable for online implementations.

In [5], a Dual PBIL (DPBIL) algorithm is presented in order to solve a partitioning problem. As defined on that work, partitioning has the form of a binary optimization problem. The dual form of the PBIL implementation allows to improve the speed of the algorithm for finding optimal solutions. This allows the implementation of DPBIL approaches in dynamic environments, in which the optimization objective's changes over the time.

In [13], a system-level partitioning problem is solved using a PBIL approach. Unlike the DPBIL algorithm, where a probability vector is used, the non-binary nature of the system-level partitioning problem requires using a different representation for the probability array. The results show that by adjusting the learning rate parameter, convergence time is speeded up at the expense of the quality of the obtained solutions. Tuning this parameter becomes a key issue in the PBIL optimization process.

Some efforts have been conducted in order to formalize the PBIL convergence process [14], [15]. In such approaches, PBIL algorithm is modeled by means of a Markov Chain and its behaviour is approximated by using an ordinary differential equation (ODE). The idea is to prove that the corresponding ODE has only stationary points which corresponds to the optimals of the configuration space of the PBIL algorithm. These works has proven that eventually, the ODE and the associated PBIL, will converge to one of those stable points.

3. The PBIL approach

The PBIL algorithm is a stochastic search method that obtains its directional information from the previous best solutions [16]. As mentioned before, PBIL algorithms represent the population of solutions by means of an array of probabilities. In the case of binary problems, the PBIL array takes the form of a vector, which stores a probability value for each attribute of the problem to be optimized. In the case of non-binary problems, it is necessary to work with a probability matrix, in order to take into account the whole solutions space. In both cases, the idea is to update iteratively the probability values in the array, in order to that the population converges to an optimal solution.

A basic version of the APBIL approach is shown in Algorithm 1. As shown on Algorithm 1, all the values in the probability array (namely P) are initialized to $1/N$, in order to take into account all the potential solutions. At each algorithmt's iteration, a new population (Pop) is generated, based on the probabilities of the array, by means of the $Create\_Population$ routine. The attributes with the highest associated probability values, will appear with more frequency in the populationt's individuals. All the individuals of the population are assessed, using the $Evaluate\_Population$ routine. For the sake of choosing the more suitable solutions to the optimization problem at hand, the $Choose\_Best$ routine uses information about the fitness of each potential solution in the population.

In the adaptive approach of the PBIL algorithm, the Learning Rate (LR) parameter must be adjusted in order to allow both exploration and exploitation of the search space. This task is performed by the $Learning\_Rule$ routine. The Entropy (E) of the probability array is calculated and used as an estimation of the population's diversity. Once the LR parameter is calculated, the P array must be updated in order to adjust the probabilities, according to the best solution found in the population. Function $Update\_Array$ is used with this aim.

Entropy decreases as the probabilities in the PBIL array tend to concentrate on single entries of each column of the P array (i.e. when an optimal solution becomes more probable). Then, when entropy value becomes less than a given tolerance, the algorithm may stop. The optimal solution can be easily derived of the probabilities on the P array.

Let's suppose that an optimization problem can be stated as a collection of M attributes, namely Q_1, Q_2, ..., Q_M. For each attribute, there are up to N choices, in order to solve the problem. Figure 1 shows a suitable probability array, for such an optimization problem. In Figure 1, $P_{(i,j)}$ represents the probability of the j attribute to be optimized using the

Algorithm 1 Basic PBIL Algorithm

Input: An $N \times M$ probability array P
Output: An optimized solution

1: $P(i,j) = \frac{1}{N}; \forall\ 1 \leq i \leq N$ and $1 \leq j \leq M$
2: **repeat**
3: $\quad Pop = Create\_Population(P)$;
4: $\quad Fitness = Evaluate\_Population(Pop)$;
5: $\quad Best = Choose\_Best(Pop, Fitness)$;
6: $\quad E = Entropy(P)$;
7: $\quad LR = Learning\_Rule(E)$;
8: $\quad P = Update\_Array(P, Best, LR)$;
9: **until** $\{E < Tolerance\}$;

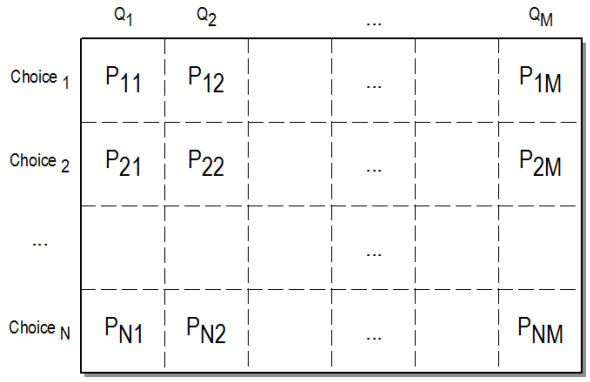

Fig. 1: A non-binary PBIL probability matrix

associated choice i. Since a single column of the matrix of Figure 1 represents the joint probability of all potential choices for a given attribute, the sum over a single column must be equal to one.

Let's suppose that after assessing a given population, it was found that for a given attribute Q_j, the best (optimal) choice is the option k. According to this, the probabilities in the array must be updated using a modified version of the Hebbian rule [17], as shown in Equation (1).

$$P_{(i,j)New} = \begin{cases} P_{(i,j)Old} + \left(1 - P_{(i,j)Old}\right) \times LR, \text{ for } i = k \\ \left(1 - P_{(k,j)New}\right) \times \frac{P_{(i,j)Old}}{1 - P_{(k,j)Old}}, \text{ for } i \neq k \end{cases} \quad (1)$$

In Equation (1), LR corresponds to the learning rate parameter. The higher the LR parameter, the fastest shall be the convergence of the algorithm, at the expense of a poorer quality of the found solutions. The adjusting of the LR parameter requires a good trade-off between quality and speed. Instead of giving to the LR parameter a fixed value, an adaptive approach varies the LR parameter whilst the algorithm converges to an optimal. The best way to do that is by calculating the systemic entropy over the PBIL array of Figure 1 [18]. In this case, systemic entropy, or simply entropy (E), may be calculated as shown in Equation (2).

$$E = -\sum_{i=1}^{N}\sum_{j=1}^{M}\left[P_{(i,j)} \times \log\left(P_{(i,j)}\right)\right] \quad (2)$$

When all problem choices are equally probable, all probabilities on the array have a value of $1/N$. In this situation, the population represented by the PBIL array has the highest diversity. Under these conditions, the maximum value of Entropy can be calculated as shown in Equation (3).

$$E_{Max} = -M \times \log\left(\frac{1}{N}\right) \quad (3)$$

When one of the choices becomes more probable than the others, entropy value decreases, meaning that the array provides less information, since some of the available options have being discarded. A value of 0 for the entropy, implies that all the attributes of the associated solution are completely defined, i.e. the PBIL algorithm has reached a unique solution.

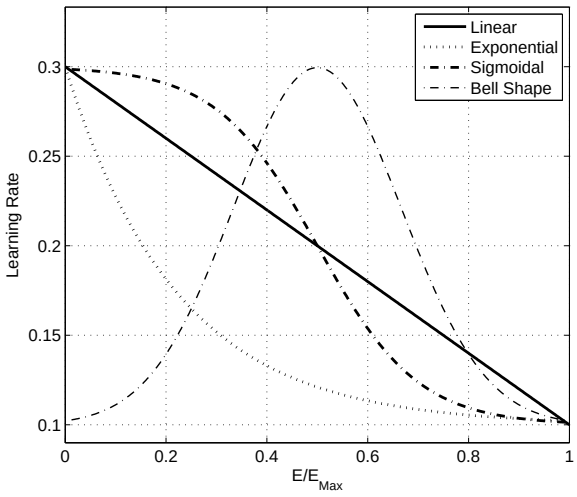

Fig. 2: Learning rules for LR between 0.1 and 0.3

The APBIL algorithm starts with high values of entropy, and they decreases as the APBIL array converges to an optimal attributes combination. Figure 2 shows three learning rules, namely linear, exponential, and sigmoidal. As shown on Figure 2, the learning rate parameter is kept at low values at the beggining of the algorithm (when entropy has maximum values), for the sake of allowing high population's diversity and exploring over all the solutions space. As the entropy decreases, as a result of the probability array updating process, the LR parameter increases, in order to improve the speed of convergence of the algorithm, and to reach the optimal solution quicker. This means that the search process allows high diversity at early iterations of the APBIL algorithm, and once the array seems to be oriented toward a given solution (low entropy), the algorithm changes the learning rate parameter, in order to speed up the convergence. Figure 2 also shows the bell shape learning rule, like that one proposed in [7]. In this case, the learning rate is kept low both at the beginning and at the end of the APBIL algorithm. The idea is to allow high population's diversity in both situations, and avoid the local optimum problem.

Concerning a multi-objective implementation of a APBIL algorithm, the main idea consists of working with several arrays (populations) independently, as is done in the island-model PBIL or IMPBIL approach [19]. As a result, at the end of the algorithm's execution, there will be several optimal solutions with different trade-offs among the objectives being optimized. There will be as much solutions as PBIL

arrays on the implementation. This implies that the learning process must be performed over several arrays, which may reduce the algorithm's performance.

Maintaining different PBIL populations generates additional problems. The first one is that a mechanism is necessary to avoid that several populations converge to the same optimal solution. As explained in [20], this may be done by using a kernel approach based on distance. The idea is that the distances among potential solutions becomes a new objective to maximize in the optimization process, in such a way that the solutions similar to an optimal in a given population, are pruned on the remaining ones.

The second problem concerning multi-objective PBIL is related to the stochastic nature of such algorithm. The random operators that are present in PBIL algorithm can lead to loosing good optimal solutions. A buffering strategy must be implemented in order to store global optimal solutions and to integrate them into the learning process while the algorithms works. Given an optimization objective, if an optimal solution is found and is better than the stored one, the previous solution is discarded and the new best solution is saved.

4. The Job Shop Scheduling optimization problem

Job Shop Scheduling [21] is a combinatorial optimization problem, which was formulated in the middle of the twentieth century. As many combinatorial optimization problems, Job Shop Scheduling is considered a NP problem, which implies that practical algorithms are aimed to find good-enough solutions instead of finding the best solutions. Scheduling is a concept related with a plethora of different optimization problems, with different levels of complexity and diverse structures too.

In order to test our adaptive PBIL algorithms, a Job Shop Scheduling problem will be used. Such a problem implies the scheduling of executable tasks in a multi-processor architecture, with several objectives to be satisfied. Three objectives were considered for optimization: the cost of the solution (N), i.e. the number of Processing Elements (PEs) required for implementing a given scheduling solution; the scheduling penalty (SP), related with tasks priorities and execution delays, and the processor occupancy (PO), which assess the eficiency in the resources usage.

The problem is stated as follows. Let's suppose a set of M independent tasks, which are going to be scheduled for their implementation over a set of up to N PEs. Although the number of tasks is well-known in advance, the number of PEs represents the first objective to be optimized in order to reduce the cost of the solution. The cheaper solution in terms of hardware cost, consists of using a single PE, which will lead to higher delays when executing the tasks. The more expensive solution implies using a single PE for each task. This enhances the performance of the designed system, because response time for each task is minimized. Due to the fact that the target architecture is homogeneous, each PE has the same cost and the execution time for a single task does not change from one specific PE to another.

A given task (T_i) has its own execution time (t_i) and a priority value (Pr_i), which ranges between 1 and Pr_{Max}. If a specific PE is assigned to execute more than one task, it is assumed that tasks will be served sequentially, using their priorities as ordering criterion. Tasks with higher priority are served before those with lower ones. The penalty for delaying a given task is calculated by means of its priority and delay to be executed. Equation (4) shows that the penalty for a specific scheduling scheme is the sum of the penalties for each task, which is defined as the product between its priority and its delay.

$$SP = \sum_{i=1}^{M} D_i \times Pr_i \tag{4}$$

In Equation (4), D_i represents the delay that a specific scheduling solution produces to the task T_i, and Pr_i is its associated priority. The system penalty (SP) is the second objective to be optimized (minimized) in the proposed scheduling problem.

Finally, the third objective to be optimized corresponds to the processor occupancy (PO) which can be simply calculated as the mean percentage of time in which processors are performing useful work (i.e. processing tasks). In this case, the PO objective must be maximized, in order to fully exploit the system's resources.

Figure 3 depicts a specific scheduling solution, in an environment of M tasks, each with a triad formed by its priority, execution time and delay. As can be seen in Figure 3, there can be up to M active PEs (the most expensive and fastest solution), but in a more efficient solution, some PEs are not used at all. Since for each PE, tasks are served on a priority basis, from the figure it can be said that priority of task T_1 is higher than priority of tasks T_i and T_M. Also, task T_1 is served first on PE_1, so the associated delay for such a task is equal to zero. Delays for remaining tasks shall depend on the subsequent execution order.

5. Experimental Results

Four APBIL Learning Rules (namely linear, sigmoidal, exponential and bell-shape) were tested in order to compare them and find the best trade-off between performance and quality for the optimization problems at hand. Such learning rules are depicted in Figure 2 and Table 1 shows their formal specification. In order to perform an equable comparison, this specification represents the only difference among the implementation of the adaptive algorithms. As shown in Table 1, LR depends on the entropy (E) of the PBIL array, as well as on the LR_{Min} and LR_{Max} parameters, which

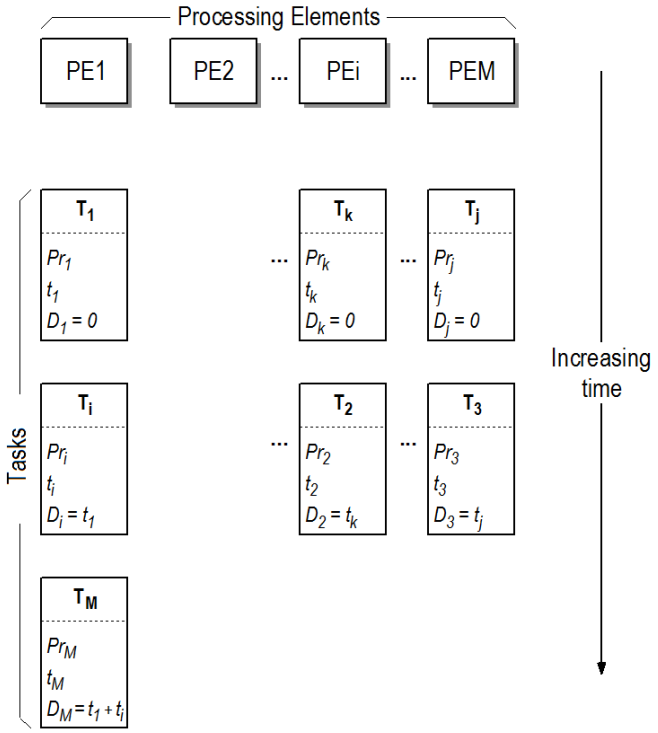

Fig. 3: A given scheduling solution

represent the minimum and maximum values for LR. For all the executions of the APBIL algorithm reported in this section, the LR_{Min} and LR_{Max} values were set to 0.1 and 0.3, respectively.

For comparison purposes, a classical Multi-Objective Evolutionary Algorithm (MOEA) was also implemented [22]. Each algorithm was used to solve the Job Shop Scheduling problem with different sizes (M).

All the algorithms were aimed to simultaneously optimize three objectives, as pointed in previous section: scheduling penalty (SP), processor occupancy (PO) and the number of PEs (N). The algorithms were tested using Matlab on a PC with an Intel Core I7 processor and 8 Gigabytes of memory.

Table 1: Formal specification of learning rules

Linear	$LR = LR_{Max} - \frac{E}{E_{Max}} \times (LR_{Max} - LR_{Min})$
Exponential	$LR = LR_{Min} + (LR_{Max} - LR_{Min}) \times e^{-4.5 \times \frac{E}{Emax}}$
Sigmoidal	$LR = LR_{Max} - \frac{LR_{Max} - LR_{Min}}{1 + e^{-10 \times \left(\frac{E}{Emax} - 0.5\right)}}$
Bell Shape	$LR = LR_{Min} + \frac{LR_{Max} - LR_{Min}}{\sqrt{2 \times \pi}} \times e^{-\frac{\left(\frac{E}{Emax} - 3\right)^2}{2}}$

Figure 4 depicts the mean convergence time for the

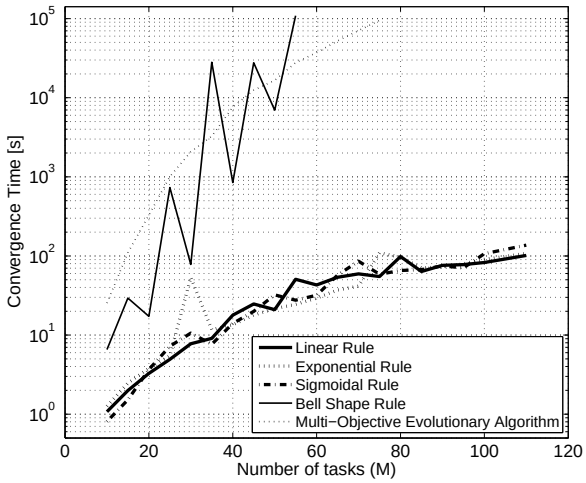

Fig. 4: Mean convergence time for the different optimization strategies

mentioned algorithms, as a function of the optimization problem's size (M). As can be seen, convergence times for MOEA and bell-shaped algorithms, becomes restrictive when dealing with higher values of M. In such cases, the execution of the algorithm was forced to stop.

Learning rules for the adaptive approaches in Figure 4, are the same depicted in Figure 2. As can be seen, there is no remarkable difference between the monotonically-increasing learning rules (linear, exponential and sigmoidal), except for a peak given on the exponential rule, for a problem size of thirty tasks. On the contrary, MOEA and bell-shaped PBIL has very poor convergence times, often several orders of magnitud above the proposals with monotonically-increasing learning rules.

With respect to the bell-shaped learning rule, it does not seems logical to decrease the learning rate at the end of the process of convergence. When a PBIL algorithm is performing the last stages of the space exploration, the probabilities in the algorithm's array tend to be concentrated on single positions of each column, which points toward the optimal solution. The results in Figure 4 suggest that there is no need to guarantee population's diversity at final stages of the PBIL algorithm's execution.

The MOEA algorithm was implemented using a vector representation for each solution on the population. Each entry in the solution's vector represents the implementation resource for a given system's task. This means that each vector on the population has a lenght of M elements. A single crossover operator was used for recombination, and the mutation was made by means of a change of a single value on a given vector position. A round-robin method and a $\mu + \lambda$ strategy was used for selection purposes [23]. As can be seen in Figure 4, MOEA's performance is very poor

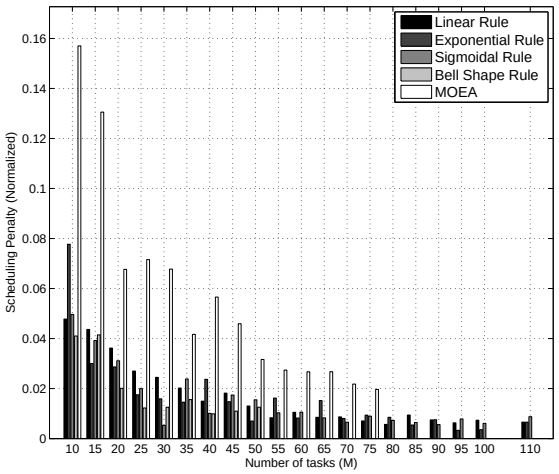

Fig. 5: Scheduling penalty for several optimization approaches

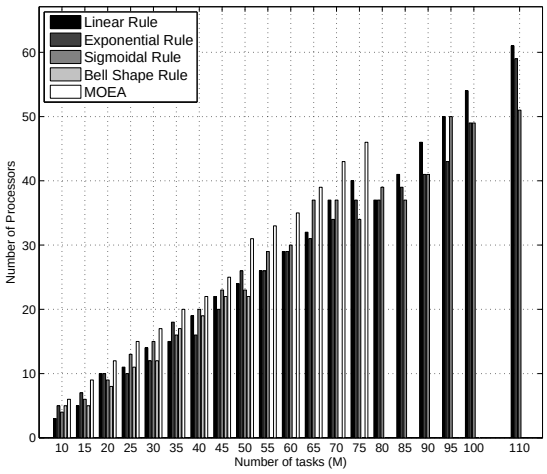

Fig. 7: Number of processors for several optimization approaches

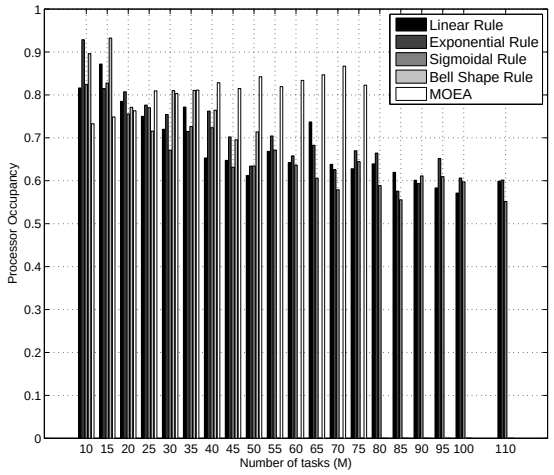

Fig. 6: Processor occupancy for several optimization approaches

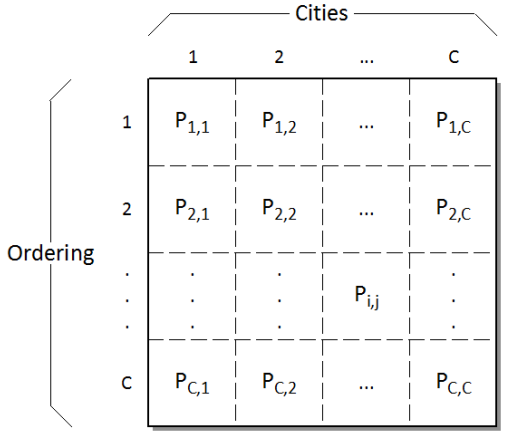

Fig. 8: A PBIL probability array representation for the TSP problem

when compared with the best APBIL algorithms because it does not have an adaptive behavior.

None of the results shown in Figure 4 would be relevant without a comparison between the quality of the solutions obtained with each approach. Figures 5, 6 and 7 show the best solutions found by each optimization algorithm, according to the objectives defined in the previous section. As can be seen, there is no remarkable differences among the PBIL algorithms, although MOEA strategy shows better solutions concerning processor occupancy. However, MOEA's quality concerning the remaining criteria (SP and N) is always poorer than those for PBIL approaches.

For the sake of providing a further insight into the performance of the proposed learning rules, a traveling salesman problem (TSP) optimization algorithm was implemented. TSP is one of the most famous NP-complete problems [24]. Given C cities, the goal is to find a minimum lenght tour which visits each city exactly once. The PBIL array used to represent such an optimization scheme is shown in Figure 8. As can be seen, the PBIL array takes the form of a $C \times C$ probability matrix. In that figure, $P_{i,j}$ represents the probability of visiting city j in the i_{th} place.

Three APBIL approaches were implemented in order to solve TSPs of several sizes. These approaches correspond to the three monotonically-increasing learning rules described before, i.e. linear, sigmoidal and exponential. The TSPs were

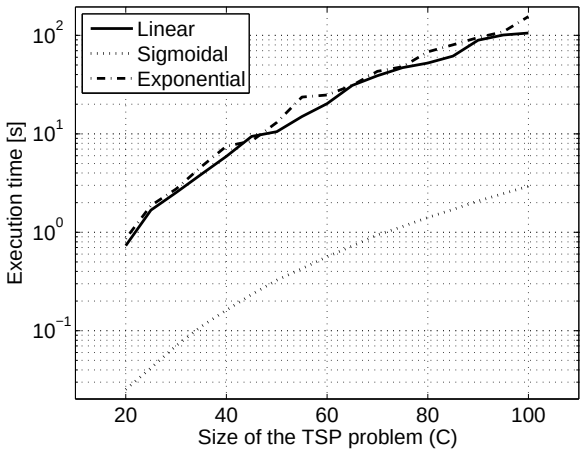

Fig. 9: Execution time for the different TSPs

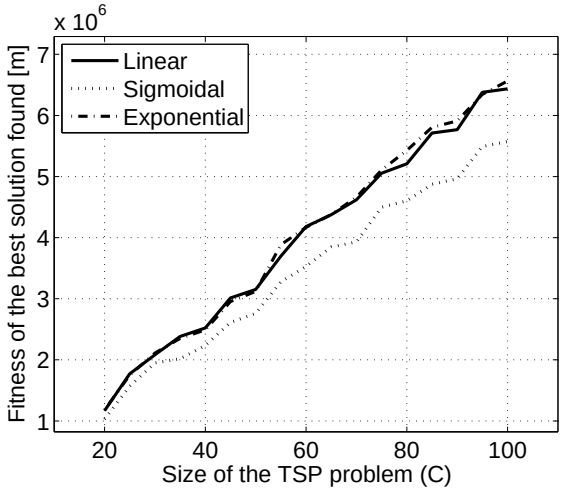

Fig. 10: Total traveled distance for several TSP problems

created by distributing randomly a total of C cities in an area of 10000 square kilometers. In order to assess the quality of the solutions provided by the APBIL search, the total traveled distance by the salesman, measured in meters, was used as a fitness value.

Figure 9 shows the mean convergence times for each APBIL implementation, for different sizes of the TSPs. Again, the only difference among APBIL implementations was the learning rule used to update the LR parameter. The sizes of the TSPs ranges from 20 to 100 cities. As mentioned in [24], complexity of the TSP problem grows very quicly with the value of C, so using larger sizes may be restrictive for simulation.

Figure 9 shows that sigmoidal learning rule has a better performance when compared with linear and exponential rules, for TSP optimization problems. Execution times related with sigmoidal learning rule are always almost an order of magnitud below the remaining alternatives, and such a trend prevails over all the range of the TSP size.

Figure 10 shows the total traveled distance for the three implementations of the APBIL algorithm. As can be seen on that figure, total traveled distance of the solutions obtained from sigmoidal learning rule are up to 16 % below of traveled distance of the remaining approaches.

According to the previous discussion, keeping low values of the LR parameter allows high population's diversity. As can be seen on Figure 2, the sigmoidal learning rule allows high population's diversity at early stages of the exploration process (i.e. when entropy values are near to its maximum). Such diversity avoids the local optimum problem, and then improves the quality of the found solutions. However, as entropy decreases and the algorithm approximates to optimal values, the sigmoidal learning rule increases the LR parameter even more than the other rules, which implies that sigmoidal rule improves exploitation at the end of the process. Such exploitation speeds up the search process, as shown on Figure 10. Figure 2 shows that sigmoidal learning rule is the second one in favoring the exploration at the begining of the search process, and also is the one which most favors the exploitation at the end of the convergence process. Such a combination of features seems to be the reason to explain why sigmoidal learning rule resulted to be both the strategy with best quality as well as the fastest one.

6. Conclusions

An adaptive PBIL strategy has been tested using several learning rules. The monotonically-increasing learning rules have shown promising results in order to speed up the convergence process. The results show that a linear or sigmoidal relationship between the learning rate and the PBIL array's entropy seems to be the best option.

Results derived from the TSP optimization suggest that sigmoidal learning rule is the best approach, in order to speed up the convergence time. However, presented results must be taken as an initial insight, in order to derive the features of the learning rules tested in this paper, which will be intended as future work.

Acknowledgment

The authors would like to thank to ARTICA, to COLCIENCIAS, to ICT Ministry of Colombia, to National University of Colombia and to University of Antioquia, for their support in the development of this work.

References

[1] H. Bai and B. Zhao, "A survey on application of swarm intelligence computation to electric power system," in *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, vol. 2, 2006, pp. 7587 –7591.

[2] N. Pindoriya, S. Singh, and K. Lee, "A comprehensive survey on multi-objective evolutionary optimization in power system applications," in *Power and Energy Society General Meeting, 2010 IEEE*, 2010, pp. 1 –8.

[3] A. Godit' andnez, L. Espinosa, and E. Montes, "An experimental comparison of multiobjective algorithms: Nsga-ii and omopso," in *Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2010*, 28 2010-oct. 1 2010, pp. 28 –33.

[4] G. B. Lamont and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Norwell, MA, USA: Kluwer Academic Publishers, 2002.

[5] L. jun Fan, B. Li, Z. quan Zhuang, and Z. qian Fu, "An approach for dynamic hardware /software partitioning based on dpbil," in *Natural Computation, 2007. ICNC 2007. Third International Conference on*, vol. 5, 2007, pp. 581 –585.

[6] M. Schmidt, K. Kristensen, and T. Randers Jensen, "Adding genetics to the standard pbil algorithm," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 2, 1999, pp. 3 vol. (xxxvii+2348).

[7] H. Pang, K. Hu, and Z. Hong, "Adaptive pbil algorithm and its application to solve scheduling problems," in *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*, 2006, pp. 784 –789.

[8] B. Babu and A. Gujarathi, "Multi-objective differential evolution (mode) for optimization of supply chain planning and management," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, sept. 2007, pp. 2732 –2739.

[9] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Pittsburgh, PA, USA, Tech. Rep., 1994.

[10] K. A. Folly and G. K. Venayagamoorthy, "Effects of learning rate on the performance of the population based incremental learning algorithm," in *Proceedings of the 2009 international joint conference on Neural Networks*, ser. IJCNN'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 3477–3484. [Online]. Available: http://portal.acm.org/citation.cfm?id=1704555.1704773

[11] Q. Zhang, T. Wu, and B. Liu, "A population-based incremental learning algorithm with elitist strategy," in *Proceedings of the Third International Conference on Natural Computation - Volume 03*, ser. ICNC '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 583–587. [Online]. Available: http://dx.doi.org/10.1109/ICNC.2007.126

[12] S. Sheetekela and K. Folly, "Power ystem controller design: A comparison between breeder genetic algorithm and population based incremental learning," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, july 2010, pp. 1 –8.

[13] F. Bolanos, J. Aedo, and F. Rivera, "System - level partitioning for embedded systems design using population - based incremental learning," in *CDES*, H. R. Arabnia and A. M. G. Solo, Eds. CSREA Press, 2010, pp. 74–80.

[14] H. Li, S. Kwong, and Y. Hong, "The convergence analysis and specification of the population-based incremental learning algorithm," *Neurocomputing*.

[15] R. Rastegar and A. Hariri, "The population-based incremental learning algorithm converges to local optima," *Neurocomputing*, vol. 69, no. 13-15, pp. 1772 – 1775, 2006, blind Source Separation and Independent Component Analysis - Selected papers from the ICA 2004 meeting, Granada, Spain, Blind Source Separation and Independent Component Analysis. [Online]. Available: http://www.sciencedirect.com/science/article/B6V10-4J9X2VD-2/2/503395260d19018514ab89fa4c646659

[16] E. Hughes, "Optimisation using population based incremental learning (pbil)," in *Optimisation in Control: Methods and Applications (Ref. No. 1998/521), IEE Colloquium on*, Nov. 1998, pp. 2/1 –2/3.

[17] R. White, "Competitive hebbian learning," in *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, vol. ii, July 1991, p. 949 vol.2.

[18] L. Wang, L. Ma, Q. Bian, and X. Zhao, "Rough set attributes reduction based on adaptive pbil algorithm," in *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference on*, 2010, pp. 21 –24.

[19] J. M. Chaves-Gonzalez, D. Dominguez-Gonzalez, M. A. Vega-Rodriguez, J. A. Gomez-Pulido, and J. M. Sanchez-Perez, "Parallelizing pbil for solving a real-world frequency assignment problem in gsm networks," in *Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008)*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 391–398. [Online]. Available: http://portal.acm.org/citation.cfm?id=1343596.1343822

[20] B. W. Silverman, *Density estimation: for statistics and data analysis*, Chapman and Hall, Eds., London, 1986.

[21] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[22] M. Castillo Tapia and C. Coello, "Applications of multi-objective evolutionary algorithms in economics and finance: A survey," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, sept. 2007, pp. 532 –539.

[23] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing (Natural Computing Series)*. Springer, October 2008. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/3540401849

[24] S. Baluja, "An empirical comparison of seven iterative and evolutionary function optimization heuristics," Tech. Rep., 1995.