Parameter Tuning for PBIL Algorithm in Geometric Constraint Solving Systems

R. Joan-Arinyo
Grup d'Informàtica a l'Enginyeria
Universitat Politècnica de Catalunya
robert@lsi.upc.edu

M.V. Luzón
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Granada
luzon@ugr.es

E. Yeguas
Departamento de Informática y Análisis Numérico
Universidad de Córdoba
eyeguas@uco.es

***Abstract*—In previous works we have shown that applying genetic algorithms to solve the Root Identification Problem is feasible and effective.**

The behavior of evolutive algorithms is characterized by a set of parameters that have an effect on the algorithms' performance. In this paper we report on an empirical statistical study conducted to establish the influence of the driving parameters in the Population Based Incremental Learning (PBIL) algorithm when applied to solve the Root Identification Problem. We also identify ranges for the parameters values that optimize the algorithm performance.

Key-words: Parameter optimization, Evolutive algorithms, Geometric Constraint Solving, Root Identification Problem.

I. Introduction

Modern computer aided design and manufacturing systems are built on top of parametric geometric modeling engines. The field has developed sketching systems that automatically instantiate geometric objects from a rough sketch, annotated with dimensions and constraints input by the user. The sketch only has to be topologically correct and constraints are normally not yet satisfied.

Geometric problems defined by constraints have an exponential number of solution instances in the number of geometric elements involved. Generally, the user is only interested in one instance such that besides fulfilling the geometric constraints, exhibits some additional properties. This solution instance is called the *intended solution*.

Selecting a solution instance amounts to selecting one among a number of different roots of a nonlinear equation or system of equations. The problem of selecting a given root was named by Bouma *et al.* in [3] the *Root Identification Problem*.

Luzón *et al.* [10], reported on a new technique to automatically solve the Root Identification Problem for constructive solvers, [7]. The technique overconstrains the geometric problem by defining two different categories of constraints. One category includes the set of constraints specifically needed to solve the geometric constraint problem. The other category includes a set of extra constraints or predicates on the geometric elements which identify the intended solution instance. Once the constructive solver has generated the space of solution instances, the extra constraints are used to drive an automatic search of the solution instances space using genetic algorithms, [11]. The search outputs a solution instance that maximizes the number of extra constraints fulfilled.

Genetic algorithms are characterized by a set of parameters which determine their evolution and for which specific values must be chosen, [1]. Furthermore, when parameters are chosen by trial and error, the relationship between their values and the performance of the algorithm cannot be established.

PBIL algorithm is a method that combines generational mechanisms with simple competitive learning. It is argued that this algorithm is simple and outperforms basic genetic algorithms on a large set of optimization problems.

In this paper we study how PBIL algorithm performs when it is applied to solve the Root Identification Problem. We also explore the possibility of identifying ranges for the values assigned to the parameters for which this algorithm shows an optimal performance.

The remainder of this work is organized as follows. Section II briefly describes the main concepts involved in the Root Identification Problem. Section III briefly describes the PBIL algorithm. Section IV describes the experimental set up. Experimental results are discussed in Section V, leaving Section VI to draw some conclusions and to suggest future work.

II. The Root Identification Problem

The problem we are facing is known as the *Root Identification Problem* and consists in selecting one solution to a system of nonlinear equations among a potentially exponential number of solutions, that is to select one root for each equation in the system.

In two-dimensional constraint-based geometric design, the designer creates a rough sketch of an object made out of simple geometric elements like points, lines, circles and arcs of circle. Then the intended exact shape is specified by annotating the sketch with constraints like distance between two points, distance from a point to a line, angle between two lines, line-circle tangency and so on. Figure 1 shows an example of a geometric constraint problem.

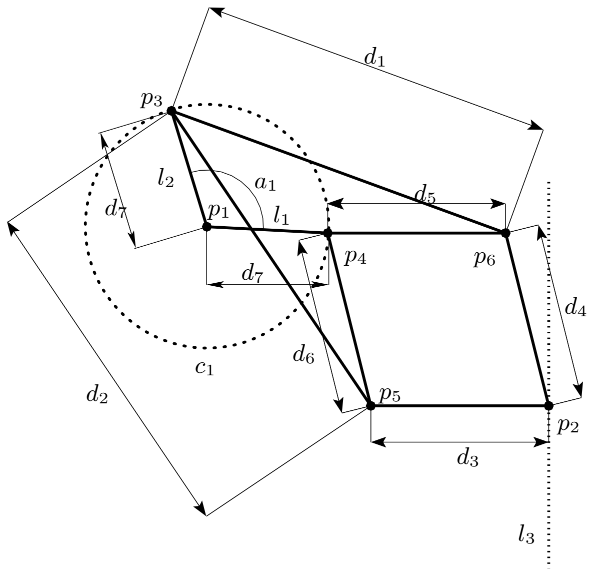

Fig. 1. Geometric problem defined by constraints.

Once the user has defined the sketch and the set of constraints, a geometric constraint solver checks whether the set of geometric constraints coherently defines the object and, if so, determines the position of the geometric elements.

Among all the geometric constraint solving techniques, our interest focuses on the one known as *constructive*, [7].

III. The PBIL Algorithm

In previous works, [15], we conducted a preliminary study to asses the potential behavior of a number of metaheuristics applied to solve the Root Identification Problem. The study considered trajectory-based and population-based metaheuristics. The results shown that the most promising algorithms were clearly in the second category, specifically PBIL, an evolutive probabilistic method, and CHC [6], a genetic algorithm. Due to space limitations we will focus on the former algorithm.

The PBIL algorithm, Baluja [2], is an evolutionary algorithm that uses a probability vector to describe the population of the genetic algorithm. In a binary encoded solutions string, the probability vector specifies the probability of each bit position containing a '1'. The probability of a bit position containing a '0' is obtained by subtracting the probability specified in the vector from 1.0.

In genetic algorithms, operations are defined and performed on the population. In PBIL, operations take place directly on the probability vector which is used to derive a population. The mechanisms used in PBIL are derived from those used in competitive learning. The aim of PBIL is to actively create a probability vector which, with high probability, represents a population of high evaluation vectors. In a manner similar to the training of a competitive learning network, the values in the probability vector are gradually shifted towards representing those in high evaluation vectors.

Figure 2 shows the PBIL algorithm used in this study. According to Baluja, [1], the main parameters affecting the PBIL algorithm evolution are,

- *Population size (N)*: number of samples in the population that must be generated per generation.
- *Mutation probability (MP)*: probability of mutation ocurring in each samples' position. Values are in $[0, 1]$.
- *Mutation shift (MS)*: amount for mutation to affect the probability vector shifting. Values are defined in $[0, 1]$.
- *Learning rate (LR)*: Learning rate that regulates the speed with which the probability vector approaches to the best found solution. Values are in $[0, 1]$.

In general, the length of the string that encodes the individuals in the population is a parameter that depends on the specific problem at hand. As we will see in Section IV, in our study it has been fixed.

```
Procedure PBILalgorithm
  INPUT
    TMAX: Number of iterations
    N : Population size
    LENGTH : Chromosome length
    MP: Mutation probability
    MS: Mutation shift
    LR: Learning rate
  OUTPUT
    P: Probability vector
    BS: Population best chromosome

  # Inicialize probability vector
  for i in [1..LENGTH] do
    P(i) := 0.5
  # Algorithm evolution
  for j in [1..TMAX] do
    for i in [1..N] do
      GenerateSampleVector (P, sample(i))
      EvaluateSample (sample(i), evaluation(i))
    FindBestSample(sample, evaluation, BS)
    # Update probability vector
    for i in [1..LENGTH] do
      P(i) := P(i) * (1.0 - LR) + BS(i) * LR
    # Mutate probability vector
    for i in [1..LENGTH] do
      if (random(0,1] < MP) then
        rshift := ChooseOneRandomly (0.0, 1.0)
        P(i) := P(i) * (1.0 - MS) + rshift * MS
EndProcedure
```

Fig. 2. Basic PBIL algorithm.

The PBIL algorithm returns both the probability vector and the corresponding vector sample with the best evaluation.

IV. Design of the Experiments

To study the behaviour of the PBIL algorithm as a function of the parameters we have applied an empirical methodology along with an statistical analysis of variance of the experimental results.

Since investigating the behavior of the genetic algorithms with all parameters variable would be hard to accomplish, we will focus on those parameters whose influence on the evolutive algorithms are considered as fundamental listed in Section III.

These parameters are called *factors*. The factor *levels* are the set of discrete different values assigned to a factor in an experiment. For

	Values						
N	10	20	30	40	50	60	70
LR	0.05	0.10	0.15	0.20	0.25		
MP	0.01	0.025	0.05	0.075	0.10		
MS	0.01	0.025	0.05	0.075	0.10		

TABLE I
FACTOR LEVELS.

each factor studied, several levels have been considered. They are shown in Table I for PBIL. The levels for the population size have been taken from previous work reported by Yeguas *et al.* in [15]. The levels for the remaining parameters have been chosen as follows. First, for each parameter, a central value has been selected among those suggested by the specific literature (see Baluja, [1]). Then we defined a number of additional levels (four or six) evenly distributed with respect to the central value. Half of them smaller and half of them greater than the central value.

We have considered a representative benchmark including 29 different geometric problems defined by constraints each with 18 geometric elements. Therefore, the length of the string that encodes the individuals in the population is 18 and the size of the search space is bounded by 2^{16} different solution instances, [3].

The run of an algorithm on a problem with an specific assignment of factors level to the parameters is called a *treatment*.

We say that a treatment is successful if and only if the algorithm has found a solution that fulfills all the extra constraints defined to select the intended solution before reaching the maximum number of evaluations allowed, rl_{max}. In our experiments this value was set to 30000 evaluations. For each treatment we recorded the actual number of evaluations performed, the *run-length*.

An *observation* is the mean run length estimated as

$$\widehat{E}(RL) = \frac{1}{k}\Sigma_{i=1}^{k} rl_i + \frac{(n-k)}{k}\ rl_{max}$$

where k is the number of successful runs and rl_i is the run-length of the ith successful run, [8]. The total number of runs was $n = 50$ each triggered with an initial random seed.

To guarantee that the samples fulfil the requirements of normality and variance, [4], it was decided to perform 50 observations for each treatment and each problem. This experimental setup yielded 875 series (one series for each treatment considered) of run-length values, $\widehat{E}(RL)$, each with 50 observations.

To elucidate the influence of the parameters on the algorithms performance, we have conducted a comprehensive statistical analysis of the empirical results by using the ANalysis Of VAriance (ANOVA), [4]. The independent variables are the algorithm parameters and the dependent variable is the mean run-length, $\widehat{E}(RL)$, required to find a solution.

V. RESULTS OF PBIL ALGORITHM

To assess the behaviour of the PBIL algorithm we have conducted unifactorial analysis, multifactorial analysis and post hoc tests.

A. Unifactorial Analysis

We have applied the one way ANOVA analysis to study the effect of each parameter listed in Section III: Population Size, (N), Mutation Probability, (MP), Mutation Shift, (MS), and Learning Rate, (LR) on the mean run-length, $\widehat{E}(RL)$, required for algorithm PBIL to find a solution. Table II shows the ANOVA results corresponding to the behavior of PBIL algorithm for a given problem instance. Starting from the left most column we have: the set of factors considered, the sum of squares, number of degrees of freedom, squared mean, Fisher test statistic F, and significance level. Unifactorial analysis results are listed in rows three through six.

Considering the ANOVA tables for the set of problems in the benchmark, results are consistent and the test of equality of means have shown a significance level smaller than 0.05 for each parameter. This means that variations in each individual parameter level lead to variations in the mean run-length with a 95% confidence level.

Factors	Sum of squares Type III	DOF	Squared Mean	F	Sig
Adjusted Model	6.15E+10	874	7.03E+07	340.764	0.000
Intersection	9.12E+10	1	9.12E+10	442060.412	0.000
N	4.36E+10	6	7.26E+09	35179.451	0.000
LR	2.64E+09	4	6.60E+08	3196.267	0.000
MP	1.19E+08	4	2.98E+07	144.486	0.000
MS	3.72E+08	4	9.31E+07	450.839	0.000
$N * LR$	1.08E+10	24	4.50E+08	2181.185	0.000
$N * MP$	4.01E+07	24	1.67E+06	8.101	0.000
$LR * MP$	2.43E+08	16	1.52E+07	73.445	0.000
$N * LR * MP$	6.08E+08	96	6.33E+06	30.684	0.000
$N * DM$	1.93E+08	24	8.06E+06	39.025	0.000
$LR * DM$	3.05E+08	16	1.91E+07	92.478	0.000
$N * LR * DM$	7.47E+08	96	7.78E+06	37.703	0.000
$MP * DM$	1.19E+08	16	7.44E+06	36.025	0.000
$N * MP * DM$	6.92E+07	96	7.21E+05	3.494	0.000
$LR * MP * DM$	4.16E+08	64	6.49E+06	31.459	0.000
$TP * LR * MP * DM$	1.23E+09	384	3.20E+06	15.520	0.000
Error	8.85E+09	42875	2.06E+05		
Total	1.62E+11	43750			
Adjusted Total	7.03E+10	43749			

R^2 = 0.874 Adjusted R^2 = 0.872

TABLE II
EXAMPLE OF ANOVA TABLE FOR PBIL ALGORITHM.

The analysis of the F statistic test showed that the parameter with the greatest influence on the mean run-length was LR in the 82.7% of the problems studied, N in the 13.7% of the cases, and MS in the 3.4% of the cases.

B. Multifactorial Analysis

The results of multiple factor ANOVA show how each parameter and all the possible combinations of factors affect the algorithm performance and the importance of that influence. In this study we assume that all factors have the same importance. Since we are considering four different factors, see Section III, we have studied the interaction between factors grouping them either in pairs, in triplets or the whole set. The ANOVA Table II, starting in the seventh row, shows the values for multifactorial analysis for the PBIL algorithm applied to a given problem instance in the benchmark.

The significance level in the 70% of the problems in the benchmark is smaller than 0.05, therefore interaction between factors is significative and they have an effect on the mean run-length of the PBIL algorithm. However, this effect is not homogeneous and only those combinations of factors where LR is involved play a noticiable role. In general, the effect on the PBIL algorithm performance is smaller when considering mutifactorial variation than when considering each of the factors separately.

R^2 is the fraction of the total squared error that is explained by the model. Thus values approaching one are desirable, [5]. R^2 values over the benchmark for PBIL ranged from 0.307 through 0.967 with an average of 0.742 and a standard deviation of 0.178. Therefore, the model accounts for most of the factors affecting the algorithm performance.

Homogeneous subsets			
MS levels	1	2	3
0.010	1355.1541		
0.025	1368.2300		
0.050	1407.1335		
0.075		1485.6030	
0.100			1604.6726
Sig.	0.0512	1	1

TABLE III
PBIL. HOMOGENEOUS SUBSETS FOR FACTOR MS.

C. Post Hoc Tests

The results in an ANOVA table serve only to indicate whether means differ significantly or not. They do not indicate which means differ from another. To elucidate this question we applied two methods. First, assuming normality, homogeneity of variance, and independent observations we applied the Tukey Honestly Significant Difference test, [14]. Then, since the Levene test, [9], does not guarantee that population variances are homogeneous, we applied the Games-Howell test, [13]. Results were basically the same. Finally, the results from Tukey's test were grouped in homogeneous subsets such that each subset grouped levels with mean run-length which do not differ significantly. In general, a large number of homogeneous subsets grouping each of them just one level or a few different levels, means that the algorithm performance is largely influenced by the factor values. Homogeneous subsets are sorted according to increasing values of the mean run-length. Thus, the first subset includes the mean run-lengths corresponding to those level factors for which the algorithm shows the best performance.

To illustrate this concept, Table III displays the homogeneous subsets generated for PBIL algorithm and one problem instance considering the mutation shift factor. In what follows, we use the homogeneous subsets to summarize the results obtained for each parameter considering all the problems in the benchmark.

Population size: The number of homogeneous subsets varies with the specific problem instance considered. 75% of the problems generated 6 or 7 subsets (7 was the largest possible number). In the 80% of the problems the first subset included just 1 or 2 levels. Therefore changing the value of the population size has an important effect on the algorithm performance. Concerning the levels included in the first subset, the best performance is achieved for populations in the range [20, 30].

Learning rate: The situation here is similar to that described for the population size. Now the maximum number of different subsets was 5 and the actual number was always either 4 or 5. In general, the first subset includes just one level. Therefore PBIL peformance is sensitive to level changes in LR. PBIL showed the best performace for LR levels in the first subset in the range [0.20, 0.25].

Mutation probability: The number of homogeneous subsets was 3 or more for the 70% of the problems studied while the number of levels included in the first subset strongly depended on the specific problem, sometimes it was just 1 and sometimes the subset included almost all of the levels. In 90% of the cases, level MP = 0.01 in the first subset lead to the best performance. Good performances were also obtained for MP values in the range [0.025, 0.050].

Mutation shift: This factor along with mutation probability appear only in the algorithm mutation step. Thus results are similar. The first subset included several levels, which did not lead to a notorious change in performance. The best performance was reached for MS = 0.01 and for levels in the range [0.025, 0.050].

D. Best Parameter Level Selection

Both one-way and multiple factor ANOVA analysis have shown that different levels of parameters have an effect on the PBIL algorithm peformance. Post hoc tests have delimited ranges for the levels where the PBIL algorithm shows the best performance. The next step is to select a set of specific factors levels which lead to the best PBIL performance. To identify them we decided to conduct an analysis of the mean run-length average values.

Measure	N	LR	MP	MS
Average	28	0.171	0.067	0.040
Standard Deviation	22.93	0.070	0.032	0.026

TABLE IV
BEST FACTOR VALUES FROM ANOVA ANALYSIS.

We applied a Best Treatment analysis. For each problem in the benchmark and for each treatment we computed the average of the mean run-length. Then the best treatment was computed as the average of these values over the set of problems. Results are shown in Table IV. The selected level values are of the same order and close to those suggested in the literature for genetic algorithms optimization, [1], [12].

VI. CONCLUSIONS AND FUTURE WORK

To solve the Root Identification Problem in Geometric Constraint Solving by means of genetic algorithms, specifically PBIL and CHC algorithms, is both feasible and effective.

As expected, the statistical study carried out shows that the influence of the values assigned to the parameters that characterizes the behaviour of the PBIL algorithm is significative. In general, the influence of parameters considered individually is greater than when considering combinations of them. R^2 values show that the model accounts for most of the factors affecting the algorithm performance.

We have identified sets of factors levels for which PBIL shows an optimal performance for the benchmark studied.

Future works will focus on studying the behaviour of the CHC algorithm and comparing the results with the obtained by applying PBIL.

Furthermore, assuming that the run length is a random variable, currently we are conducting experiments to figure out the intrinsic run length distributions for PBIL. This would lead to determine an optimal value for the maximum number of iterations allowed given a specific solution quality required.

VII. ACKNOWLEDGMENTS

This research has been supported by FEDER and CICYT under the project TIN2007-67474-C03-01.

REFERENCES

[1] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA, 1994.

[2] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. Technical Report CMU-CS-95-141, Carnegie Mellon University, Pittsburgh, PA, 1995.

[3] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. Geometric constraint solver. *Computer Aided Design*, 27(6):487–501, June 1995.

[4] G. C. Canavos. *Applied probability and statistical methods*. Little Brown and Company, 1984.

[5] N.R. Draper and H. Smith. *Applied Regression Analysis*. John Wiley and Sons, Inc., New York, NY, third edition, 1998.

[6] L.J. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In Gregory J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 265–283. Morgan Kaufmann Publishers, 1991.

[7] C.M. Hoffmann and R. Joan-Arinyo. A brief on constraint solving. *Computer-Aided Design and Applications*, 2(5):655–663, 2005.

[8] H.H. Hoos and T. Stützle. Evaluating Las Vegas algorithms. Pitfalls and remedies. In *Proceedings of the 14th Conference on Uncertainly in Artificial Intelligence*, pages 238–245. Morgan Kaufmann, 1998.

[9] H. Levene. Contributions to probability and statistics: Essays in honor of Harold Hotelling. pages 278–292. Stanford University Press, 1960.

[10] M. V. Luzón, R. Joan-Arinyo, and A. Soto. Genetic algorithms for root multiselection in constructive geometric constraint solving. *Computer & Graphics*, 27:51–60, 2003.

[11] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

[12] M. Sebag and A. Ducoulombier. Extending population-based incremental learning to continuous search spaces. In *Parallel Problem Solving from Nature-PPSN V*, volume 1498, pages 418–427, 1998.

[13] D.J. Sheskin. *Handbook of parametric and nonparametric statistical procedures*. CRC Press, New York, 1997.

[14] J.W. Tukey. The problem of multiple comparisons. Princeton University, 1953 (Unpublished).

[15] E. Yeguas, M.V. Luzón, and E. Barreiro. Experimentos y análisis de resultados tras la aplicación de metaheurísticas al problema de la selección de la solución deseada. Technical Report IT2/2006, Universidade de Vigo, 2006. Written in Spanish.