Parameter Tuning for PBIL and CHC Algorithms to Solve the Root Identification Problem in Geometric Constraint Solving

R. Joan-Arinyo
Grup d'Informàtica a l'Enginyeria
E.T.S. d'Enginyeria Industrial de Barcelona
Universitat Politècnica de Catalunya
robert@lsi.upc.edu

M.V. Luzón
Departamento de Informática
Escuela Superior de Ingeniería Informática
Universidad de Vigo
luzon@uvigo.es

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

January 15, 2007

Abstract

Evolutive algorithms are among the most successful approaches for solving a number of problems where systematic search in huge domains must be performed. One problem of practical interest that falls into this category is known as *The Root Identification Problem* in Geometric Constraint Solving, where one solution to the geometric problem must be selected among a number of possible solutions bounded by an exponential number. In previous works we have shown that applying genetic algorithms, a category of evolutive algorithms, to solve the Root Identification Problem is both 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 PBIL and CHC evolutive algorithms when applied to solve the Root Identification Problem. We also identify ranges for the parameter values that optimize the algorithms performance.

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

1 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 [5] the *Root Identification Problem*.

Several approaches to solve the Root Identification Problem have been reported in the literature. Examples are: selectively moving the geometric elements, conducting a dialogue with the constraint solver that identifies interactively the intended solution, and preserving the topology of the sketch input by the user. For a discussion of these approaches see, for example, Bouma *et al.* [5], and Luzón [19], and references therein.

Luzón *et al.* [18, 19], and Barreiro [3], reported on a new technique to automatically solve the Root Identification Problem for constructive solvers, [14]. 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. See Goldberg, [13], and Michalewicz, [21]. The search outputs a solution instance that maximizes the number of extra constraints fulfilled.

Genetic algorithms are a category of evolutive algorithms that are characterized by a set of parameters which determine the evolution of the algorithms and for which specific values must be chosen. According to Baluja [1] and Sebag *et al.* [22], values for these parameters are commonly chosen in practice by trial and error, tuned by hand, or taken from other fields.

Unfortunately, these methods do not guarantee to produce optimal parameter settings, and because experimental results are always problem dependent, values that proved to be useful in other applications may not perform well on untried tasks. See Wolpert and Mcready [26]. Furthermore, when parameters are chosen in these ways, the relationship between the parameter values and the performance of the algorithm cannot be established.

Population-Based Incremental Learning (PBIL), Baluja [1], and *Cross generational elitist selection Heterogeneus recombination and Cataclismic mutation* (CHC), Eshelman [11], are two evolutive algorithms that have received a large amount of attention as general purpose function optimizers. PBIL algorithm is a method that combines generational mechanisms with simple competitive learning. It is argued that this algorithm is simple and outperforms genetic algorithms on a large set of optimization problems. CHC is a nontraditional genetic algorithm whose crossover operation is highly disruptive that results in a search ability more effective than that of traditional genetic algorithms by balancing

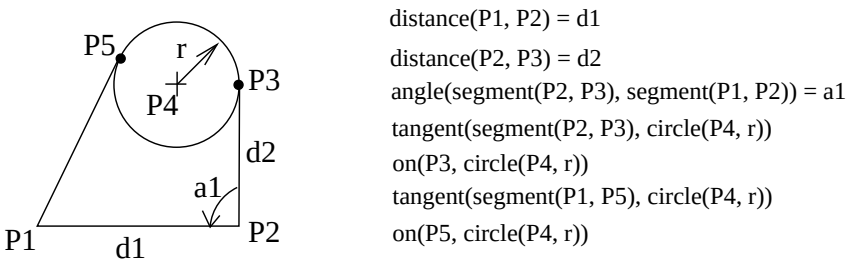

Figure 1: Geometric problem defined by constraints.

diversity and convergence.

In this paper we study how PBIL and CHC algorithms perform when they are applied to solve the Root Identification Problem. We also explore the possibility of identifying ranges for the values assigned to the parameters for which these algorithms show an optimal performance.

The remainder of this work is organized as follows. Section 2 briefly describes the main concepts involved in the Root Identification Problem. Section 3 briefly describes the PBIL and CHC algorithms. Section 4 describes the experimental set up. Experimental results are discussed in Section 5 and Section 6. Section 7 briefly compares PBIL and CHC performances, leaving Section 8 to draw some conclusions.

2 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. We first briefly describe the context where this problem arises. Then we give the criteria we use to define the *solution instance model*, that is, the solution of interest we want to select.

2.1 Constructive Geometric Constraint Solving

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 sketch of a constraint-based design. A geometric constraint solver then checks whether the set of geometric constraints coherently defines the object and, if so, determines the position of the geometric elements.

Many techniques have been reported in the literature that provide powerful and efficient methods for solving systems of geometric constraints. For example, see Hoffmann and Joan-Arinyo [14] and references therein for an extensive analysis of work on constraint solving. Among all the geometric constraint solving techniques, our interest focuses on the one known as *constructive*.

Constructive solvers have two major components: the *analyzer* and the *constructor*. The analyzer symbolically determines whether a geometric problem defined by constraints is solvable. If the prob-

$$
\begin{aligned}
P_1 &= point(0,0)\\
P_2 &= point(d_1,0)\\
\alpha_1 &= direction(P_1,P_2)\\
\alpha_2 &= adif(\alpha_1,a_1)\\
P_3 &= rc(line(P_2,\alpha_2),circle(P_2,d_2),i_1)\\
\alpha_3 &= direction(P_2,P_3)\\
\alpha_4 &= asum(\alpha_3,\pi/2)\\
Q_1 &= rc(line(P_2,\alpha_4),circle(P_2,r),i_2)\\
P_4 &= rc(line(Q_1,\alpha_3),circle(P_3,r),i_3)\\
Q_2 &= midpoint(P_1,P_4)\\
r_1 &= distance(P_1,Q_2)\\
P_5 &= cc(circle(P_4,r),circle(Q_2,r_1),i_4)
\end{aligned}
$$

Figure 2: Construction plan for the object in Figure 1.

lem is solvable, the output of the analyzer is a sequence of construction steps, known as the *construction plan*, that describes how to place each geometric element in such a way that all constraints are satisfied. After assigning specific values to the parameters, the constructor interprets the construction plan and builds an object instance, provided that no numerical incompatibilities arise.

The specific construction plan generated by an analyzer depends on the underlying constructive technique and on how it is implemented. For example, the ruler-and-compass constructive approach is a well-known technique where each constructive step in the plan corresponds to a basic operation solvable with a ruler, a compass and a protractor. In practice, this simple approach solves most useful geometric problems. Figure 2 shows a construction plan for the object of Figure 1, generated by the ruler-and-compass geometric constraint solver reported by Joan-Arinyo and Soto in [16].

Function names in the plan in Figure 2 are self explanatory. For example function $adif$ denotes subtracting the second angle from the first one and $asum$ denotes the addition of two angles while rc and cc stand for the intersection of a straight line and a circle, and the intersection of two circles, respectively.

Fudos and Hoffmann, [12], show that a well constrained geometric constraint problem has, in general, an exponential number of solutions with respect to the number of geometric elements in the problem. For example, consider a geometric constraint problem that properly places n points with respect to each other. Assume that the points can be placed serially, each time determining the next point by two distances from two already placed points. In general, each point can be placed in two different locations corresponding to the intersection points of two circles. For n points, therefore, we could have up to 2^{n-2} solutions.

Following Bouma *et al.* [5], the problem of selecting one solution to the geometric constraint solving problem is known as the *Root Identification Problem*, a solution for which all the extra constraints hold is an *intended solution instance*.

2.2 Root Identification Problem and Evolutive Algorithms

Work by Booker *et al.* [4], and by Eiben and Ruttkay, [10], proved that evolutive algorithms are an efficient and effective method to solve general problems when they can be expressed as optimization problems.

In this context, Luzón *et al.*, [18, 20], showed how the Root Identification Problem can be formulated as an optimization problem and applied evolutive algorithms to solve it. In this technique, the user annotates the geometric problem with two categories of constraints. One 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, represented by the construction plan, the extra constraints are used to drive an automatic search of the solution instances space using genetic algorithms. The search outputs a solution instance that maximizes the number of extra constraints fulfilled. See Michalewicz, [21], for a detailed study of genetic algorithms and Goldberg, [13], for their application in search and optimization.

In this work we study the behavior of PBIL and CHC algorithms when they are applied to solve the Root Identification Problem as well as to explore the possibility of identifying ranges for the parameters' values for which these algorithms show an optimal performance.

3 The Evolutive Algorithms Studied

In previous works, [27, 28], 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, a probabilistic method, and CHC, a genetic algorithm. For the sake of completness, in what follows we give a brief description for each of these two algorithms.

3.1 The PBIL 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. Figure 3 shows an example with three small populations of 5 bit solution vectors. The population size is 4. Notice that the first and third representations for the population are the same, although the solution vectors each represents are entirely different.

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.

Population 1	**Population 2**	**Population 3**
0 0 1 1 0	1 0 1 0 0	1 0 1 0 0
1 1 0 0 1	1 1 0 0 1	0 1 0 1 1
1 1 0 0 0	1 1 0 0 0	1 0 1 0 0
0 0 1 1 1	1 1 0 0 1	0 1 0 1 1
Representation 1	**Representation 2**	**Representation 3**
0.5 0.5 0.5 0.5 0.5	1.0 0.75 0.25 0.0 0.5	0.5 0.5 0.5 0.5 0.5

Figure 3: The probability representation in PBIL algorithm.

Figure 4 shows the PBIL algorithm used in this study. According to Baluja, [1], the main parameters affecting the PBIL algorithm evolution are,

- *Population size*: number of samples in the population that must be generated per generation.
- *Mutation probability*: probability of mutation ocurring in each samples' position. Values are in $[0, 1]$.
- *Mutation shift*: amount for mutation to affect the probability vector shifting. Values are defined in $[0, 1]$.
- *Learning rate*: 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 4, in our study it has been fixed.

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

3.2 The CHC Algorithm

The CHC algorithm is a nontraditional genetic algorithm which combines a conservative selection strategy that always preserves the best individuals found so far with a radical, highly disruptive recombination operator that produces offsprings that are maximally differents from both parents.

Figure 5 shows the main procedures involved in the CHC algorithm used in this study. For a general description see Eshelman [11]. The parameters affecting the evolution of the CHC algorithm for which the user must provide a value are:

- *Population size*: number of individuals in the population evolving in the search process.
- *Divergence rate*: percentage of the best found solutions used as a pattern to construct the new population in the restart stage. Values are in $[0, 1]$.
- *Difference threshold*: maximum similarity degree allowed between two individuals in the population in the crossover stage. Values range in $[0,\mathrm{L}/2]$, where L is the length of the string that encodes the individuals in the population.

```
Procedure PBILalgorithm
  INPUT
        TMAX : Number of iterations to allow learning
        N : The population size. Number of samples to produce per generation
        LENGTH : Length of encoded individuals
        MP : Probability of mutation ocurring in each position
        MS: Amount for mutation to affect the probability vector
        LR: Learning rate
  OUTPUT
        P : Probability vector
        BEST_SAMPLE : Population sample with highest evaluation for P

  # Inicialize probability vector
  for i in [1..LENGTH] do
        P(i) := 0.5

  # Algorithm evolution
  for j in [1..TMAX] do

        # Generate Samples
        for i in [1..N] do
           GenerateSampleVector (P, sample(i))
           EvaluateSample (sample(i), evaluation(i))

        # Find vector sample corresponding to the best evaluation
        FindBestSample(sample, evaluation, BEST_SAMPLE)

        # Update probability vector
        for i in [1..LENGTH] do
           P(i) := P(i) * (1.0 - LR) + BEST_SAMPLE(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
```

Figure 4: Basic PBIL algorithm.

Parameter	Values						
Population size	10	20	30	40	50	60	70
Learning rate	0.05	0.10	0.15	0.20	0.25		
Mutation probability	0.01	0.025	0.05	0.075	0.10		
Mutation shift	0.01	0.025	0.05	0.075	0.10		

Table 1: Factor levels for PBIL algorithm.

- *Best individuals*: number of individuals to be replaced when the population is reinitialized.

As in the case of the PBIL algorithm the length L has been fixed in our study. The CHC algorithm returns the individual in the population with the best evaluation.

4 Design of the Experiments

To study the behaviour of the PBIL and CHC algorithms 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 that have been listed in Section 3.

These parameters are called *factors*. The factor *levels* are the set of discrete different values assigned to a factor in an experiment. For each factor studied, several levels have been considered. They are shown in Table 1 for PBIL and in Table 2 for CHC. The levels for the population size have been taken from previous work reported by Yeguas *et al.* in [28]. 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], for PBIL and Eshelman, [11], for CHC). 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 search space is bounded by 2^{16} different solution instances.

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

Parameter	Values						
Population size	10	20	30	40	50	60	70
Divergency rate	0.20	0.25	0.30	0.35	0.40	0.45	0.50
Difference threshold	2	3	4	5	6		
Best individuals	1	2	3				

Table 2: Factor levels for CHC algorithm.

Procedure CHCalgorithm
INPUT
TMAX : Number of iterations allowed
N : The population size
LENGTH : Length of encoded individuals
D : Difference threshold
DR : Divergence rate
M : Number of individuals in the population to be replaced
OUTPUT
C : Best individual in population

t := 0
InicializePopulation (P_t)
EvaluatePopulation(P_t)

# Algorithm evolution
while not termination condition is satisfied **do**
t := t + 1
SelectForRecombination (P_{t-1}, C_t)
Recombine (C_t, C'_t)
EvaluatePopulation(C'_t)
SelectForSelection (P_{t-1}, C'_t, P_t)
if Equals (P_{t-1}, P_t) **then**
DecrementDifferenceThreshold (D)
if $D < 0$ DivergePopulation
D := DR * (1.0 - DR) * LENGTH
EndProcedure

Procedure DivergePopulation
INPUT
N : Population size
LENGTH : Length of encoded solution
DR : Divergence rate
P_t : Current population
P_{t-1} : Previous population
M : Number of individuals in the population to be replaced
OUTPUT
P : Population

Replace M individuals in P_t with the best M members of P_{t-1}
for all but the best M members of P_t **do**
Flip DR * LENGTH bits at random
EvaluatePopulation(P_t)
EndProcedure

Figure 5: Basic CHC algorithm.

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 iterations allowed, rl_{max}. In our experiments this value was set to 30000 iterations. For each treatment we recorded the actual number of iterations 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, [15]. 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, [6], it was decided to perform 50 observations for each treatment and each problem. This experimental setup yielded 875 series for PBIL and 735 series for CHC (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), [6, 7, 8]. The independent variables are the algorithm parameters and the dependent variable is the mean run-length, $\widehat{E}(RL)$, required to find a solution.

5 Results of PBIL Algorithm

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

5.1 Unifactorial Analysis

We have applied the one way ANOVA analysis to study the effect of each parameter listed in Section 3.1: 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 3 shows the ANOVA results corresponding to the behavior of PBIL algorithm for a given problem instance. Starting from the left most column, first 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.

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. Next we discuss in more detail the effect of each factor on the mean run-length.

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 3: Example of ANOVA table for PBIL algorithm.

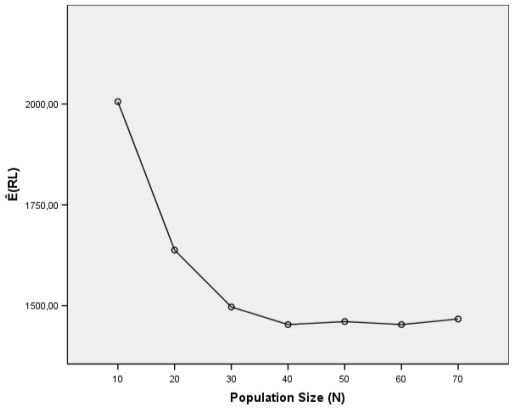

Figure 6: PBIL. Mean run-length, $\widehat{E}(RL)$, versus population size.

5.1.1 Population Size

Figure 6 shows values of the mean run-length versus the population size corresponding to one geometric problem in the benchmark.

The ANOVA box plot in Figure 7 shows the mean run-length for the set of geometric problems in the benchmark versus the population size. The ANOVA box plot is used to assess normality and to select those factor levels for which the algorithm shows the best performance. See [7]. The factor levels are in the X axis. Five different values are represented on the Y axis. The thin vertical line shows the run-length range bounded by the minimum and the maximum values. The bold horizontal line inside the boxes indicates the median of the run-length. The box itself indicates where most of the cases lie: the lower side is the first quartile while the upper side is the third quartile. Points that do not seem to belong to the data set are called *outliers* and are marked individually as either an asterisc or a small circle depending on how far from the median they lie. Ideally, for normal distributions, the rectangle is in the middle of the range, the median line is in the middle of the rectangle and is coincident with the average value. If most of the box is on one side or the other of the median line, this indicates that the run-length is not normal for that factor level.

Results in Figure 7 show that the best performance of PBIL corresponds to a population size in the range [15, 30]. Then as the population size increases, the algorithm performace decreases and the standard deviation widens. When the population is smaller than 15, the performance decreases. This is consistent with the fact that the population is unable to diversify and explore some promising regions in the solution space. See Baluja, [1].

5.1.2 Mutation Probability

As in the case above, Figure 8 shows values of the mean run-length versus the mutation probability corresponding to one of the geometric problems in the benchmark, and Figure 9 depicts the box plot for all the problems in the benchmark for the mutation probability factor.

The run-length medians are almost constant therefore the mutation probability has little influence on the PBIL algorithm performance. However, dispersion increases with the mutation probability. This means that the diversity generated by updating the current population with the best instance from the

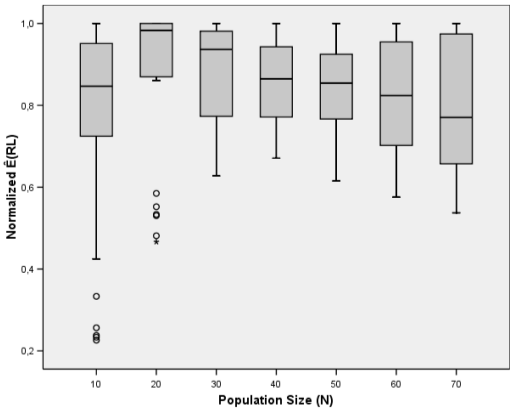

Figure 7: PBIL. Normalized mean run-length versus population size.

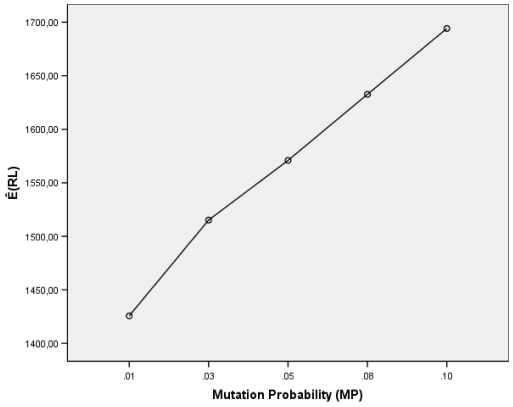

Figure 8: PBIL. Mean run-length, $\widehat{E}(RL)$, versus mutation probability.

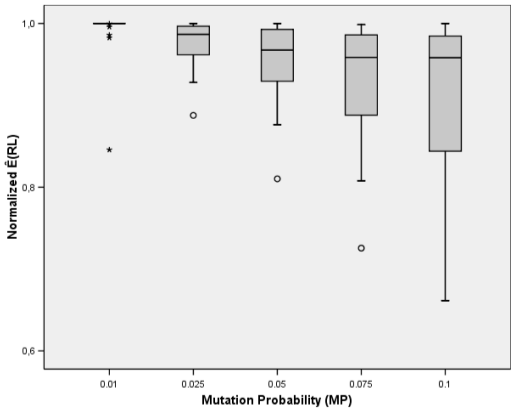

Figure 9: PBIL. Normalized mean run-length versus mutation probability.

previous generation allows to explore the solucion space. Therefore, randomly generated diversity in PBIL does not play an important role in our application.

5.1.3 Mutation Shift

Figure 10 shows values of the mean run-length versus the mutation shift corresponding to one of the geometric problems in the benchmark, and Figure 11 shows the box plot for the set of geometric problems in the benchmark. Note that the behaviour is similar to that shown by the mutation probability.

5.1.4 Learning Rate

Figure 12 and Figure 13 show the same concepts as in the previous factors referred here to the learning rate. Now, PBIL shows the best performance for learning rates in the higher values studied, that is, in the range [0.2, 0.25]. Notice that, for these learning rate values, the standard deviation is minimum.

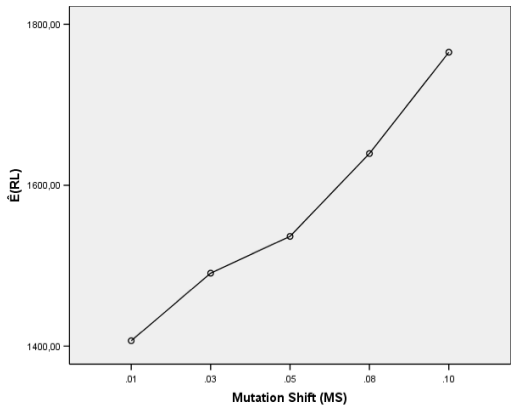

Figure 10: PBIL. Mean run-length, $\widehat{E}(RL)$, versus mutation shift.

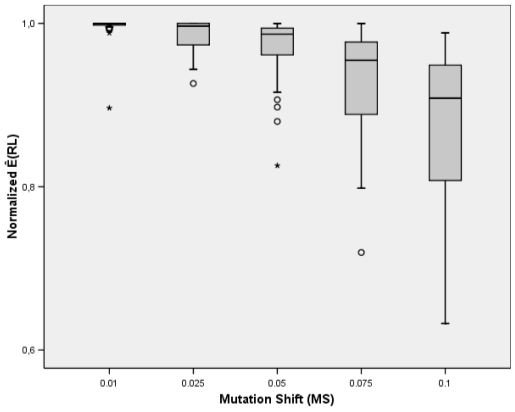

Figure 11: PBIL. Normalized mean run-length versus mutation shift.

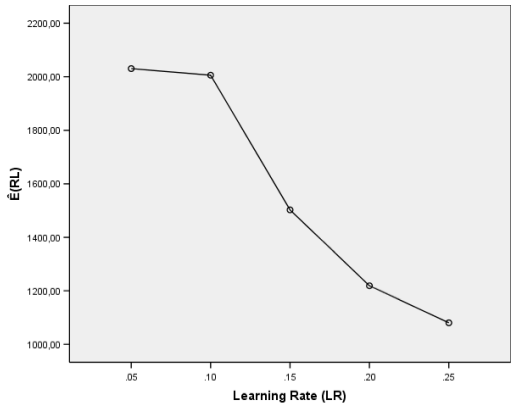

Figure 12: PBIL. Mean run-length, $\widehat{E}(RL)$, versus learning rate.

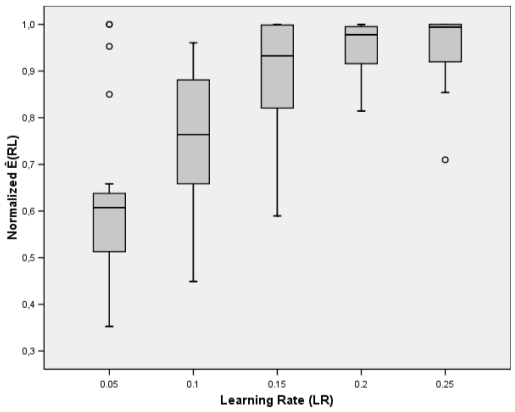

Figure 13: PBIL. Normalized mean run-length versus learning rate.

Parameter	Media	Standard Dev.
N	0.4317	0.2942
LR	0.5382	0.2466
MP	0.0287	0.0357
MS	0.0537	0.0479
$N * LR$	0.3769	0.2384
$N * MP$	0.0111	0.0219
$LR * MP$	0.0362	0.0341
$N * LR * MP$	0.0379	0.0533
$N * MS$	0.0098	0.0172
$LR * MS$	0.0387	0.0382
$N * LR * MS$	0.0389	0.0554
$MP * MS$	0.0258	0.0251
$N * MP * MS$	0.0067	0.0066
$LR * MP * MS$	0.0453	0.0420
$N * LR * MP * MS$	0.0463	0.0413

Table 4: Averaged partial Eta squared for interaction parameters in PBIL algorithm.

5.2 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 3, we have studied the interaction between factors grouping them either in pairs, in triplets or the whole set. The ANOVA Table 3, 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.

To measure the degree of association between effect of simultaneously changing the level of several factors and the mean run-length, we figured out the partial Eta squared. This measures the proportion of the effect plus the error variance that is attributable to the factors interaction, [24]. Table 4 includes the averaged partial Eta squared and its standard deviation computed for the benchmark. Clearly, the factors whose interaction plays a significative role are the population size (P) and the learning rate (LR). Notice that this result is congruent with those described above.

R^2 is the fraction of the total squared error that is explained by the model. Thus values approaching one are desirable, [9]. 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	1355.154165		
0.025	1368.230008		
0.050	1407.133550		
0.075		1485.603044	
0.100			1604.672698
Sig.	0.051253	1	1

Table 5: PBIL. Example of homogeneous subsets for factor MS.

.

5.3 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, [25]. Then, since the Levene test, [17], does not guarantee that population variances are homogeneous, we applied the Games-Howell test, [23]. Results were basically the same. Finally, the results from Tukey's test were grouped in homogeneous subsets such that each subset groups 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 factors for which the algorithm shows the best performance.

To illustrate this concept, Table 5 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].

ANOVA	Measure	N	LR	MP	MS
One-way	Average	33	0.202	0.021	0.018
	Standard Deviation	22.54	0.059	0.028	0.014
Multifactorial	Average	28	0.171	0.067	0.040
	Standard Deviation	22.93	0.070	0.032	0.026

Table 6: PBIL. Best level values from ANOVA analysis.

.

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 however levels in the lowest rang did not lead to a significative change in performance. The best performance was reached for MS = 0.01 and for levels in the range [0.025, 0.050].

5.4 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.

We performed this analysis on the results yielded by the one-way ANOVA and on the results coming from the multifactorial ANOVA. In the first case, interactions between factors are not considered and we applied a Mean Analysis. Here for each problem in the benchmark and each parameter, we identified the parameter level that yielded the smallest mean run-length. Then the best level was figured out as the average of these levels over the set of problems.

In the second case interaction between factors are considered and 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 6. As expected, level values selected depend on the source of the data. However, they are of the same order and close to those suggested in the literature for genetic algorithms optimization, [1, 22].

6 Results of CHC Algorithm

To assess the behavior of the CHC algorithm we have conducted the same set of experiments as in the case of PBIL algorithm: unifactorial analysis, multifactorial analysis and post hoc tests, over the same benchmark.

Factors	Sum of squares Type III	DOF	Squared Mean	F	Sig
Adjusted Model	7.53E+11	734	1.03E+09	280.107	0
Intersection	8.90E+11	1	8.90E+11	242836.433	0
N	6.15E+11	6	1.02E+11	27959.848	0
DR	5.55E+08	6	9.26E+07	25.272	0
D	8.05E+10	4	2.01E+10	5495.595	0
M	3.37E+08	2	1.68E+08	45.971	0
$N * DR$	1.77E+09	36	4.92E+07	13.443	0
$N * D$	5.16E+10	24	2.15E+09	587.418	0
$DR * D$	1.28E+08	24	5.31E+06	1.451	0.071
$N * DR * D$	5.32E+08	144	3.70E+06	1.009	0.453
$N * M$	1.03E+09	12	8.57E+07	23.392	0
R $DR * M$	4.01E+07	12	3.34E+06	0.912	0.534
$N * DR * M$	2.49E+08	72	3.46E+06	0.943	0.614
$D * M$	5.57E+07	8	6.96E+06	1.9	0.055
$N * D * M$	2.05E+08	48	4.28E+06	1.168	0.198
$DR * D * M$	2.04E+08	48	4.24E+06	1.158	0.211
$N * DR * D * M$	1.34E+09	288	4.64E+06	1.267	0.001
Error	1.32E+11	36015	3.66E+06		
Total	1.77E+12	36750			
Adjusted Total	8.85E+11	36749			

R^2 = 0.851 Adjusted R^2 = 0.848

Table 7: Example of ANOVA table for CHC algorithm.

6.1 Unifactorial Analysis

We have applied the one way ANOVA analysis to study the effect of each parameter listed in Section 3.2, Population Size (N), Divergence Rate (DR), Difference Threshold (D), and Best Individuals (M), on the mean run-length, $\widehat{E}(RL)$, required to find a solution. Table 7 shows an example of ANOVA table corresponding to the behavior of CHC algorithm for a problem instance 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% level of confidence.

6.1.1 Population Size

Figure 14 shows values of the mean run-length versus the population size corresponding to one geometric problem in the benchmark.

The ANOVA box plot in Figure 15 shows the mean-run length for the set of geometric problems in the

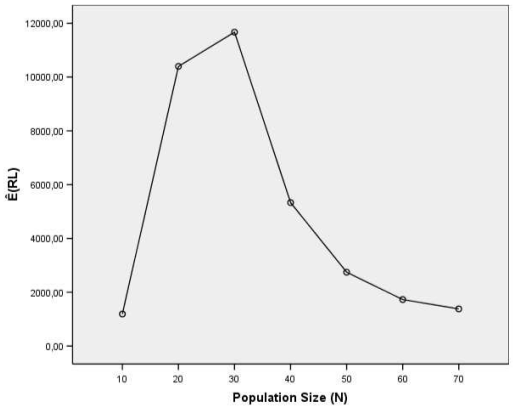

Figure 14: Mean run-length, $\widehat{E}(RL)$, versus population size.

benchmark versus population size. We can see that the normalized mean run-length reaches a local maximum for populations with N = 10 and N = 40.

6.1.2 Divergence Rate

The one-way ANOVA shows that in the 75% of the problems in the benchmark divergence rate has a significance level smaller than 0.05, that is, the divergence rate has an influence on the CHC algorithm performance with a 95% of confidence for these problem instances.

Divergence rate DR is involved in the calculations only when the population is reinitialized after finding a local optimum. See Figure 5. For our benchmark, more than 80% of the algorithm runs reached a solution without reinitializing the population. This explains why in 25% of the cases studied divergence rate does not have an influence on the CHC algorithm performance. To validate or to reject this result we should study the CHC performance feeding it with larger problems with more local optima.

Figure 16 shows values of the mean run-length versus the divergence rate size corresponding to one

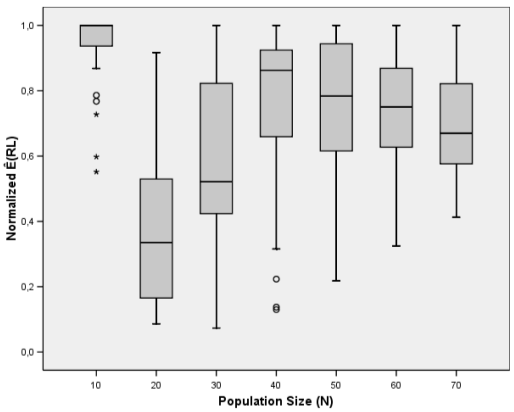

Figure 15: Normalized mean run-length versus population size.

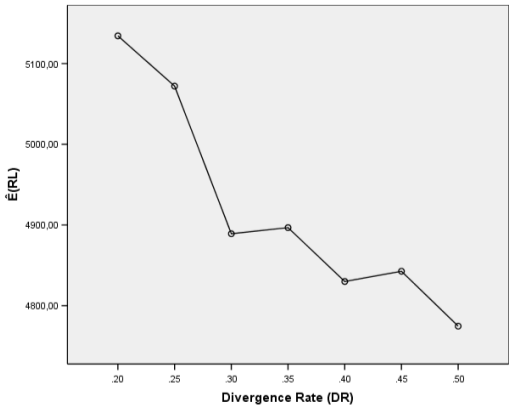

Figure 16: CHC. Mean run-length, $\widehat{E}(RL)$, versus divergence rate.

geometric problem in the benchmark.

The ANOVA box plot in Figure 17 shows the mean-run length for the set of geometric problems in the benchmark versus divergence rate. Clearly, the CHC performance increases with the divergence rate whereas the dispersion decreases. The rational for this behavior is that after finding a local optimum, diversification favours the algorithm to jump into new regions in the search space.

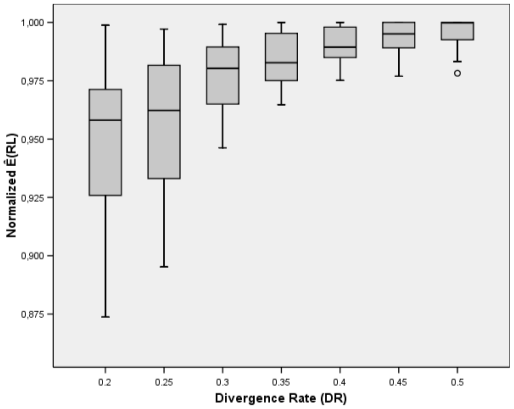

Figure 17: CHC. Normalized mean run-length versus divergence rate.

6.1.3 Difference Threshold

One-way ANOVA analysis shows that difference threshold significance is always smaller than 0.05. We can discard with a probability of 95% the hypothesis that the means for the expectation of factor are equal for each parameter level. Therefore, difference threshold have significant influence in the CHC performance, independently of the problem instance.

Figure 18 shows values of the mean run-length versus the difference threshold corresponding to one geometric problem in the benchmark.

The ANOVA box plot in Figure 19 shows the mean run-length for the set of geometric problems in

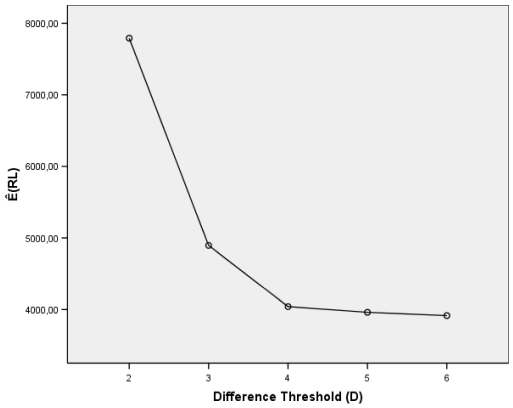

Figure 18: CHC. Mean run-length, $\widehat{E}(RL)$, versus difference threshold.

the benchmark versus difference threshold. CHC performance increases with the difference threshold whereas the dispersion decreases. Similarly to what happens with the divergence rate, difference threshold comes into action after reinitializing the population. Therefore, as expected, they both show a coherent behavior.

6.1.4 Best Individuals

One-way ANOVA analysis shows that best individuals significance is smaller than 0.05 only in the 66% of the problems studied. This factor only has an effect when the CHC algorithm reinitializes the population. This explains the high number of cases where the factor does not make a significative influence on the algorithm performance.

Figure 20 shows values of the mean run-length versus the best individuals for one geometric problem in the benchmark.

The ANOVA box plot in Figure 21 shows the mean run-length for the set of geometric problems in the benchmark versus the best replacement factor. CHC shows the best performance when only the

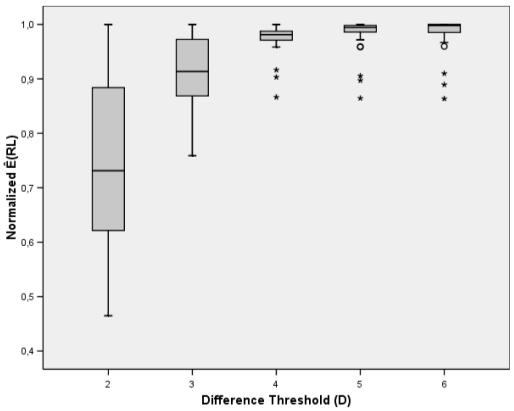

Figure 19: CHC. Normalized mean run-length versus difference threshold.

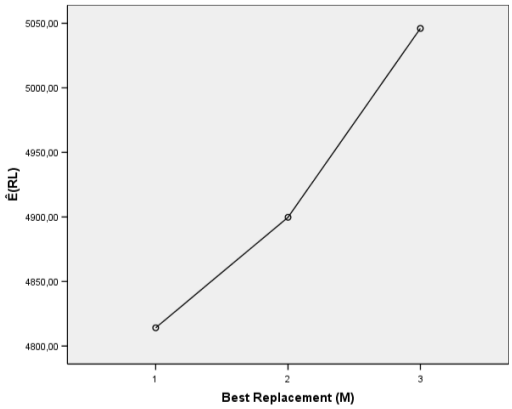

Figure 20: CHC. Mean run-length, $\widehat{E}(RL)$, versus best replacement.

best individual from the previous generation is carried over the reinitialized population. The effect of this factor on the algorithm performace is almost unnoticeable.

6.2 Multifactorial Analysis

Results yielded for one instance problem by multiple factor ANOVA for CHC algorithm are illustrated in Table 7 starting in row seven. The situation depicted in this table can be extended to most of the problems in the benchmark: most of the factors interactions do not have a significative influence in the algorithm performance. Notice that in these cases, always factors involved in the population reinicialization are present, M, DR and D.

As in the case of PBIL, we computed for CHC the averaged partial Eta squared and its standard deviation for the benchmark. Values are those given in Table 8. Factors whose interaction plays a significative role are the population size (N) and difference threshold (D).

R^2 values over the benchmark for CHC ranged from 0.226 through 0.851 with an average of 0.586 and a standard deviation of 0.197.

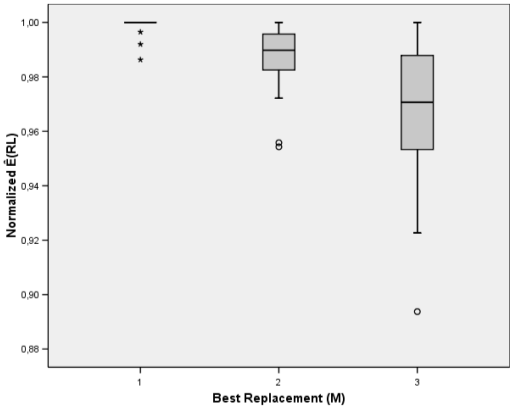

Figure 21: CHC. Normalized mean run-length versus best replacement.

Parameter	Average	Standar Dev.
N	0.5290	0.2038
DR	0.0032	0.0039
D	0.1366	0.1616
M	0.0024	0.0051
$N * DR$	0.0070	0.0056
$N * D$	0.1538	0.0992
$DR * D$	0.0009	0.0004
$N * DR * D$	0.0043	0.0012
$N * M$	0.0055	0.0092
$DR * M$	0.0004	0.0002
$N * D * M$	0.0020	0.0006
$D * M$	0.0003	0.0003
$N * D * M$	0.0016	0.0006
$DR * D * M$	0.0014	0.0004
$N * DR * D * M$	0.0080	0.0011

Table 8: Averaged partial Eta squared for interaction parameters in CHC algorithm.

6.3 Post Hoc Tests

Once we have determined that means are significatively different, we applied to CHC algorithm the same post hoc studies as we did to PBIL algorithm, see Section 5.3. Results were again basically the same.

Results from Tukey's test were grouped in homogeneous subsets. Table 9 displays the homogeneous subsets generated for CHC algorithm and one problem instance considering the best individuals factor. We use them to summarize the results obtained for each parameter considering all the problems in the benchmark as follows.

Population size: The number of homogeneous subsets found in the 75% of the problems was 6 or 7. For most of the other 25% of the problems the number of subsets was 5. In 65% of the problems the number of levels in the first subset was 1. In the rest of problems it was 2 or 3. Therefore, changes in this factor are significative. The best performance is achieved for N = 10 (75% of the problems) and for N in the range [40, 60].

Homogeneous subsets		
M levels	1	2
1	4814.133134	
2	4899.720181	4899.720181
3		5045.998193
Sig.	0.359680	0.051369

Table 9: CHC. Example of homogeneous subsets for factor M.

.

ANOVA	Measure	N	D	DR	M
One-way	Average	26	4.760	0.457	1.138
	Standard Deviation	20.96	1.431	0.051	0.441
Multifactorial	Average	21	4.207	0.316	1.621
	Standard Deviation	15.97	1.567	0.077	0.820

Table 10: CHC. Best level values from ANOVA analysis.

.

Divergency rate: On the one hand, only for a small number of problems in the benchmark the number of homogeneous subsets was greater than 2. On the other hand, in the 83% of the problems the first subset included 5 or more different levels (the maximum allowed was 7). Therefore, the influence of this parameter on the algorithm performance is rather small, as suggested by the fact that it is involved only in the reinitialization step. The best performance corresponds to levels in the first subset always within the range [0.4, 0.5].

Difference threshold: Generally, the number of homogeneous subsets for each problem is 3 (the maximum allowed was 5). The first subset included several levels but, for most of the problems in the benchmark, the lowest levels showed a clear influence on the algorithm performance. In the 80% of the problems, the best performance was achieved for D levels in the range [4, 6]. However, performance changes for different values in the range were unnoticeable.

Best individuals: Similarly to the divergency rate, this factor is involved only in the reinitialization step. Only for a small number problems in the benchmark the number of homogeneous subsets was greater than 2 and the first homogeneous subset included, in general, 2 or 3 different levels (3 was the maximum allowed). Levels in the first subset were, basically, 1 and 2. The best performance was achieved for M = 1.

6.4 Best Parameters Level Selection

To select the best set of parameters levels for CHC algorithm, we applied the same procedures as in Section 5.4. Results are shown in Table 10. These results are homogeneous for each parameter and close to those recommended by Eshelman, [11].

7 PBIL versus CHC

Values considered in this section are derived from multifactorial ANOVA analysis. Figure 22 plots for each problem in the benchmark, the smallest mean run-length that allowed PBIL and CHC algorithms to find a solution instance. We can see that, except for problems 4 and 5, CHC algorithm clearly outperforms PBIL algorithm.

Figure 23 plots the mean run-length average yielded by the best parameters levels selected in Section 5 and Section 6 for each problem in the benchmark. CHC outperforms PBIL in the 75% of the problems. The largest difference in performance shows up in problem number 9 for which both PBIL and CHC show the worst performance.

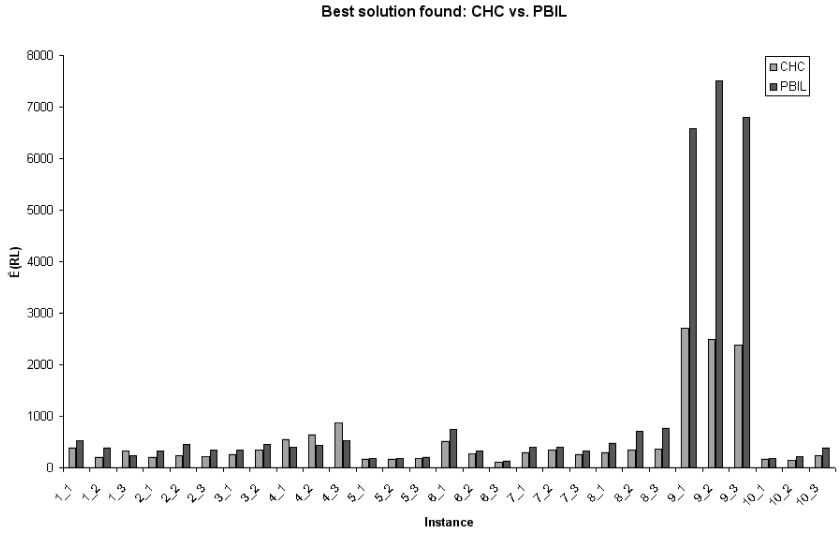

Figure 22: Smallest mean run-length for each problem instance.

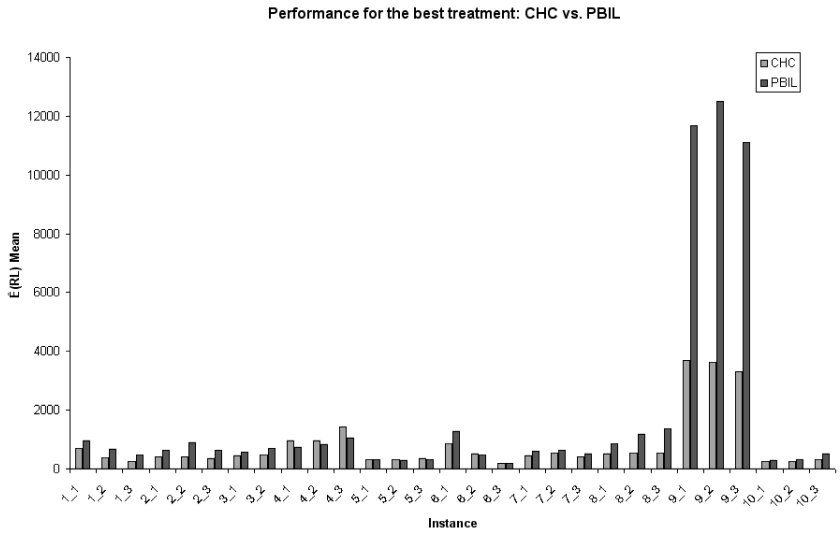

Figure 23: Mean run-length average for the best parameters levels versus problem instance.

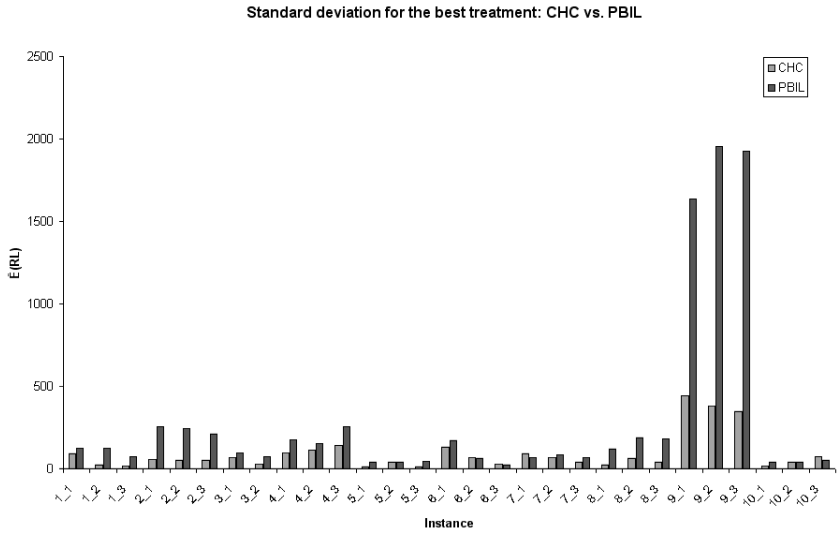

Figure 24: Standard deviation of the mean run-length for the best parameters levels versus problem instance.

Figure 24 plots for each problem in the benchmark the standard deviation for the mean run-length corresponding to the parameters levels that lead to the best performance for both PBIL and CHC algorithms given in Figure 22. Clearly, CHC shows intervals for the best values which are narrower than those of PBIL.

8 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 these algorithms 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 algorithms performance.

We have identified sets of factors levels for which PBIL and CHC, respectively, show an optimal performance for the benchmark studied. For the problems in this benchmark, we can conclude that CHC performs better than PBIL.

Assuming that the run length is a random variable, currently we are conducting experiments to figure out the intrinsic run length distributions for PBIL and CHC. This would lead to determine an optimal value for the maximum number of iterations allowed.

9 Acknowledgments

This research has been supported by FEDER and CICYT under the projects TIN2004-06326-C03-01 and TIN2004-06326-C03-02. We thank E. Barreiro for providing us with the benchmark used in this work.

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] E. Barreiro. *Modelización y optimización de algoritmos genéticos para la selección de la solución deseada en resolución constructiva de restricciones geométricas*. PhD thesis, Universidade de Vigo, June 2006. Written in Spanish.

[4] L.B. Booker, D.B. Fogel, D. Whitley, and P.J. Angeline. Recombination. In T. Bäck, D.B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C3.3, pages C3.3:1–C3.3:10. Institute of Physics Publishing Ltd and Oxford University Press, 1997.

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

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

[7] G.M. Clarke and D. Cooke. *A Basic Course in Statistics*. Arnold, 1998.

[8] P.R. Cohen. *Empirical Methods for Artificial Intelligence*. MIT Press, 1995.

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

[10] A.E. Eiben and Zs. Ruttkay. Self-adaptivity for constraint satisfaction: Learning penalty functions. In *Third IEEE World Conference on Evolutionary Computation*, pages 258–261, Nagoya, Japan, 1996. IEEE Service Center.

[11] 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.

[12] I. Fudos and C.M. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics*, 16(2):179–216, April 1997.

[13] D.E. Goldberg. *Genetic Algorithms in Search, Optimization Machine Learning*. Addison Wesley, 1989.

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

[15] 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.

[16] R. Joan-Arinyo and A. Soto-Riera. Combining constructive and equational geometric constraint solving techniques. *ACM Transactions on Graphics*, 18(1):35–55, January 1999.

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

[18] 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.

[19] M.V. Luzón. *Resolución de Restricciones Geométricas. Selección de la Solución Deseada*. PhD thesis, Dept. Informática, Universidad de Vigo, December 2001. Written in Spanish.

[20] M.V. Luzón, A. Soto, J.F. Gálvez, and R. Joan-Arinyo. Searching the solution space in constructive geometric constraint solving with genetic algorithms. *Applied Intelligence*, 22:109–124, 2005.

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

[22] 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.

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

[24] B.G. Tabachnick and L.S. Fidell. *Using multivariate statistics*. Harper & Row, New York, NY, 2nd edition, 1989.

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

[26] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transaction on Evolutionary Computation*, 1(1):67–82, 1997.

[27] E. Yeguas, M.V. Luzón, and E. Barreiro. Estudio e implementación de metaheurísticas para solucionar el problema de la selección de la solución deseada. Technical Report IT1/2006, Universidade de Vigo, 2006. Written in Spanish.

[28] 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.