A Model to Measure the Performance of Evolutionary Algorithms Applied to Solve the Root Identification Problem

R.Joan-Arinyo
Grup d'Informàtica a l'Enginyeria
Universitat Politècnica de Catalunya
Av Diagonal 647, 8a. 08028 Barcelona
M. V. Luzón
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Granada
Periodista Daniel Saucedo Aranda s/n, 18071, Granada, Spain,
E. Yeguas
Departamento de Informática y Análisis Numérico
Universidad de Córdoba
Campus Rabanales. 14071 Córdoba - Spain

Abstract

The existence of a model for representing the performance of metaheuristics applied to solve problems with high computational requirements is paramount to determine the solution quality given a certain avaliable run-time and vice versa.

In this work, a statistical model is proposed to describe the performance of evolutionary algorithms applied to solve the Root Identification Problem. Given an unknown problem size, a parameter setting and a performance model are estimated for two well-known evolutionary algorithms, Population-Based Incremental Learning (PBIL) and Cross generational elitist selection Heterogeneus recombination and Cataclismic mutation (CHC). The performance model is validated over a benchmark corresponding to huge search spaces.

***Keywords*: Parameter optimization, Performance model, Evolutionary algorithms, Geometric Constraint Solving, Root Identification Problem, PBIL and CHC**

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 the *Root Identification Problem* [10].

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, [10], and [45], and references therein. Not only it is important to have in mind the interactive nature of computer aided design and manufacturing systems, [39, 60], but also the high difficulty to solve the problem, [24].

A new technique to automatically solve the Root Identification Problem for constructive solvers, [29], was reported in the literature: [45, 46] and [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, [30]. The search outputs a solution instance that maximizes the number of extra constraints fulfilled.

Genetic algorithms are a category of evolutionary algorithms. Evolutionary algorithms (or, as a matter of fact, any heuristic method) are characterized by a set of parameters which determine the evolution of the algorithms and for which specific values must be chosen. One of the main difficulties of applying an evolutionary algorithm to a given problem is to decide on an appropiate set of parameter values. Those values are commonly chosen in practice by trial and error, tuned by hand, or taken from other fields, [43].

Furthermore, the evolution of metaheuristics in the search process supposes to take decisitions where randomness appears. The knowledge of the random vari-

able run-time can provide valuable information to analyze and characterize an algorithm behavior. It will make possible to define a performance model for each algorithm and to obtain indications for improving the algorithm performance and comparing algorithms in a suitable way. See [8, 49, 62].

When an unknown large problem size needs to be solved, it is very useful and paramount in most of the cases to have an idea of the solution quality which an algorithm can reach with a certain available run-time. In the inverse sense, it is necessary to know the required run-time to reach a certain solution quality. The application of a metaheuristic would not be blind at all if a reliable model can approximate its empirical results.

In previous works, [64, 65], two algorithms, among a large set of different population-based and single-point metaheuristics, [49, 8], have been demonstrated to yield the best results when they are applied to the Root Identification Problem: *Population-Based Incremental Learning* (PBIL), [6], and *Cross generational elitist selection Heterogeneus recombination and Cataclismic mutation* (CHC), [22].

PBIL and CHC are two evolutionary 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 diversity and convergence.

In this paper we propose a performance model for evolutionary algorithms when they are applied to solve the Root Identification Problem. The performance model is particularized to PBIL and CHC algorithms. It is composed by a parameter setting and a machine-independent run-time analysis (run-length vs. success probability). A method for predicting the performance model is presented and validated with a benchmark constituted by huge problem sizes.

The remainder of this work is organized as follows. Section 2 briefly describes the main concepts involved in the Root Identification Problem and how evolutionary algorithms can be applied to solve it. Section 3 briefly describes the PBIL and CHC algorithms. Section 4 presents a parameter setting for the performance model. The performance model is described in Section 5. Section 6 presents a method for predicting the behaviour model and validates it with a benchmark constituted by huge problem sizes, leaving Section 7 to draw some conclusions and to suggest future work.

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. Finally, we explain how evolutionary algorithms can be applied to solve the Root Identification Problem.

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. Fig. 1 shows a mechanism as a geometric constraint problem. This mechanism, known as the Peaucellier's linkage, [16], transforms the circular motion of point p_3, along circle c_1, into the translation of point p_2 along the straight line l_3. As a geometric constraint problem, Peaucellier's linkage includes six points, p_i, $1 \leq i \leq 6$, and two straight segments l_1, l_2. The set of geometric constraints are those listed in Fig. 2 which includes point to point distance constraint, $dpp()$, coincidence, $on()$, and angle between two straight segments, $angle()$.

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.

Many techniques have been reported in the literature that provide powerful and efficient methods for solving systems of geometric constraints. For example, see [29] 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 problem 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, for example, computing the square root of a negative value.

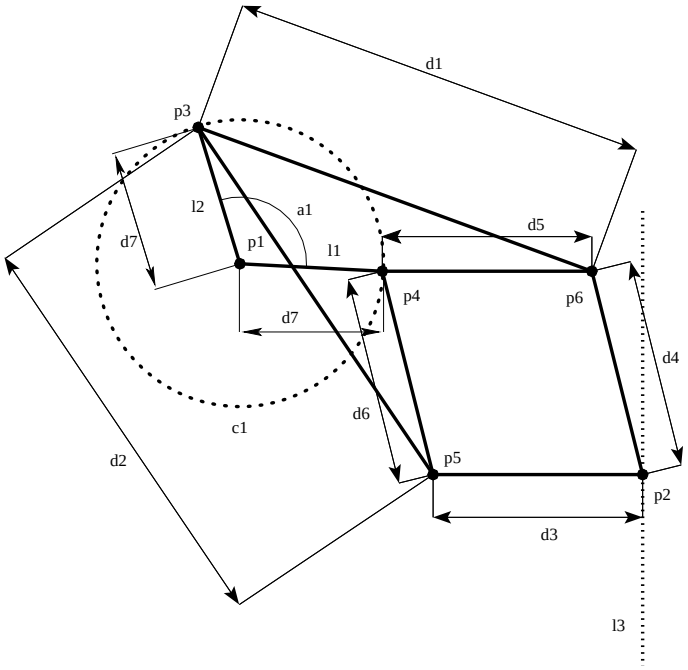

Figure 1: Paucellier's linkage: Geometric problem defined by constraints

2.2 The Construction Plan

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. Fig. 3 shows a construction plan for the object in Fig. 1, generated by the ruler-and-compass geometric constraint solver reported in [39].

Function names in the plan in Fig. 3 are self explanatory. For example, function $line2P()$ defines a straight line through two points, $circleCR()$ defines a circle

1.	dpp(p_3, p_6, d_1)	8.	dpp(p_4, p_5, d_7)
2.	dpp(p_3, p_5, d_2)	9.	on(p_1, l_1)
3.	dpp(p_2, p_5, d_3)	10.	on(p_4, l_1)
4.	dpp(p_2, p_6, d_4)	11.	on(p_1, l_2)
5.	dpp(p_4, p_6, d_5)	12.	on(p_3, l_2)
6.	dpp(p_4, p_5, d_6)	13.	angle(l_1, l_2, a_1)
7.	dpp(p_4, p_5, d_7)		

Figure 2: Geometric constraints for Peaucellier's linkage

1.	$p_1 = pointXY(0,0)$	9.	$p_5 = icc(x_2, x_3)$
2.	$p_3 = pointXY(0,d_7)$	10.	$x_4 = circleCR(p_3, d_1)$
3.	$l_2 = line2P(p_1, p_3)$	11.	$x_5 = circleCR(p_4, d_5)$
4.	$l_1 = lineAP(l_2, p_1, -a_1)$	12.	$p_6 = icc(x_4, x_5)$
5.	$x_1 = circleCR(p_1, d_7)$	13.	$x_6 = circleCR(p_5, d_3)$
6.	$p_4 = ilc(l_1, x_1)$	14.	$x_7 = circleCR(p_6, d_4)$
7.	$x_2 = circleCR(p_3, d_2)$	15.	$p_2 = icc(x_6, x_7)$
8.	$x_3 = circleCR(p_4, d_6)$		

Figure 3: Construction plan for Peaucellier's linkage

by its center and radius, and functions $ilc()$, $icc()$ denote line-circle intersection and circle-circle intersection respectively.

In [24] is shown 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. See Figure 4. Therefore, for n points, once the first two points have been placed, we could have up to 2^{n-2} solutions. Possible different locations of geometric elements corresponding to different roots of systems of nonlinear algebraic equations can be distinguished by enumerating the roots with an integer index. For a more formal definition see [23, 45].

Following [10], 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*.

The problem of finding an specific real solution to a geometric constraint problem has been classified as NP-complete in [24].

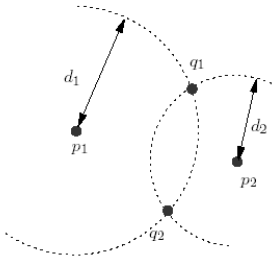

Figure 4: Possible placements of a point

2.3 Root Identification Problem and Evolutionary Algorithms

Geometric constraint solving is paramount in parametric modeling, which is at the heart of Computer Aided Design and Manufacturing systems. In this field, user-system interactivity in real time is a must, [39, 60], therefore we need to devise techniques to efficiently solve the Root Identification Problem.

Evolutionary algorithms are an efficient and effective method to solve general problems when they can be expressed as optimization problems, as was proved in [21] and [9].

In this context, in [46, 44] is shown how the Root Identification Problem can be formulated as an optimization problem and how metaheuristics, in particular, evolutionary algorithms, can be applied 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 metaheuristics, specifically genetic algorithms. The search outputs a solution instance that maximizes the number of extra constraints fulfilled. A detailed study of genetic algorithms and their application in search and optimization is presented in [26] and [47].

In this work we study the behavior of evolutionary algorithms when they are applied to solve the Root Identification Problem. A performance model is proposed and validated. The study is particularized to CHC and PBIL algorithms.

2.4 Components of Our Root Identification Evolutionary Algorithms

Two are the general concepts in evolutionary algorithms that need to be specified according to the specific problem we have at hand: the fitness function and the coding scheme.

2.4.1 Fitness Function

We will solve the Root Identification Problem by overconstraining the geometric constraint problem: the intended solution instance to a well constrained problem is specified by defining a set of predicates on the geometric elements. As extra constraint, the user can apply the usual geometric constraints or specific topological constraints, like

$$PointOnSide(p; line(p_i; p_j); side)$$

which means that point p must be placed on one of the two open half spaces defined by the straight line through points p_i, p_j, oriented from p_i to p_j . Parameter *side* takes values in $\{right, left\}$.

In our case, only $PointOnSide()$ and $Clockwise()$ predicates have been used. The latter indicates the clockwise orientation of three points. Let $NPred$ be the number of extra constraints,

$$Predicate() = \{PointOnSide(), Clockwise()\}$$

defined in a concrete instance I of the problem. The proposed fitness function $Fitness$ consists on counting the number of additional predicates which a given solution instance I fulfills.

$$Fitness(I) = \sum_{k=1}^{NPred} eval(Predicate_k, I) \tag{1}$$

where $eval(Predicate_k, I)$ evaluates the solution instance I with respect to the k-th predicate. It returns value 1 if the solution I fulfills the predicate and 0 otherwise.

The main aim of the evolutionary algorithms is to maximize F, [45]. When a solution is not feasible, that is, numerical incompatibilities arise in the constructor, the value assigned to the fitness function is -1. Hence, the larger the $Fitness$ value, the better (since this means that the solution is more similar to the intended exact shape required by the user).

2.4.2 Coding Scheme

Recall that we consider ruler-and-compass constructive geometric constraint solving. In this context, geometric operations correspond to quadratic equations, thus each constructive step has at most two different roots. Let s_j denote the integer parameter associated by the solver with the j-th intersection operation, either $ilc()$ or $icc()$, occurring in the construction plan. Since we are interested only in solution instances that actually are feasible, we only need to consider integer parameters s_j taking value in the set of signs $S = \{-1, +1\}$ that identifies each intersection point. Clearly, each solution can be coded as a binary string of a certain length L.

The number of geometric elements in the problem determines the number of different solutions. Assume that L is the total number of $ilc()$ plus $icc()$ intersection operations in the construction. We define the *index* associated with the construction plan as the ordered set $\mathcal{I} = \{s_1, \ldots, s_j, \ldots, s_L\}$ with $s_j \in S, 1 \leq j \leq L$. Moreover, the construction plan can be expressed as a first order logic formula, [32]. Therefore the cartesian product of sets, $\mathcal{I} = S^L$, defines the space where the solution instances to the geometric constraint problem belong to. The set of indexes define the solutions to the geometric constraint problem.

3 The Evolutionary Algorithms Studied

In previous works, [64, 65], 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 single-point and population based metaheuristics according to the classification given in [8].

The single-point search methods applied were local search algorithms, [1], simulated annealing, [40], tabu search, [25] and multistart local search, [48, 51]. The population-based search methods applied were genetic algorithms, [30, 54], estimation of distribution algorithms, [42], and ant colony optimization, [19].

The results showed that the most promising algorithms were clearly in the population-based category. Specifically PBIL, a estimation of distribution algorithm, 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, [6], is a general heuristic search algorithm inspired by the standard genetic algorithm defined in [30]. PBIL is an evolutionary algorithm which includes many of the features from the standard genetic algorithm such as binary string representation, the notion of individuals, fitness measures and mutations. Contrary to the standard genetic algorithm it does not maintain a population of individuals but instead PBIL contains a Probability Vector (PV). At each generation a new population of individuals is sampled according to the probabilities specified in the PV. The population is evaluated and the PV is updated by dragging it towards the best individual in the population. This is repeated for each generation. The effect is that the sampled individuals in the following generations will have an increasing resemblance with the best individuals found in the previous generations. Eventually the PV converges towards some local (or global) optimal solution.

In a binary encoded solutions string, the PV 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 PV from 1.0. Fig. 5 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 order to prevent the values of the PV from converging quickly to values close to 1.0 or 0.0 the PV undergoes a small amount of random mutation at each generation. This avoids premature convergence. The PV is initialized with all entries set to 0.5 that causes the first population to be created at random. In contrast to the standard genetic algorithm the PBIL compresses the population into a single PV.

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 5: The probability representation in PBIL algorithm.

In genetic algorithms, operations are defined and performed on the population. In PBIL, operations take place directly on the PV 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 PV 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 PV are gradually shifted towards representing those in high evaluation vectors.

Fig. 6 shows the PBIL algorithm used in this study: the basic algorithm extended by a simple restart for the PV when stagnation is detected. According to [6] the main parameters affecting the PBIL algorithm evolution are,

- *Population size* (PS): number of samples in the population that must be generated per generation.
- *Mutation probability* (MP): probability of mutation ocurring in each PV position. Values are in $[0, 1]$.
- *Mutation shift* (MS): amount for mutation to affect the PV shifting. Values are defined in $[0, 1]$.
- *Learning rate* (LR): learning rate that regulates the speed with which the PV approaches to the best found solution. Values are in $[0, 1]$. A small learning rate leads to more exploration of the search space. On the other hand, a higher learning rate tends to lead to more exploitation of the information gained during the previous search.

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

```
Procedure PBILalgorithm
  INPUT
    TMAX: Number of iterations
    PS : Population size
    L : Chromosome length
    MP: Mutation probability
    MS: Mutation shift
    LR: Learning rate
  OUTPUT
    PV: Probability vector
    BG: Best global solution found

  # Initialize probability vector
  for i in [1..L] do
    PV(i) := 0.5

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

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

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

    # Update best global solution
    BG := UpdateBestSol(BS, BG)

    # Update probability vector
    for i in [1..L] do
      PV(i) := PV(i) * (1.0 - LR) + BS(i) * LR

    # Mutate probability vector
    for i in [1..L] do
      if (random(0,1] < MP) then
        rshift := ChooseOneRandomly (0.0, 1.0)
        PV(i) := PV(i) * (1.0 - MS) + rshift * MS

    # Check stagnation and restart probability vector
    if ( Stagnation(PV) ) then
      for i in [1..L] do
        PV(i) := 0.5
EndProcedure
```

Figure 6: Pseudocode for PBIL algorithm.

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. The four main components of the algorithm are shown as follows:

- *An elitist selection*: the PS members of the current population are merged with the offspring population obtained from it and the best PS individuals are selected to compose the new population. In case that a parent and an offspring have the same fitness value, the former is preferred to the latter.

- *A highly disruptive crossover, HUX*: it crosses over exactly half of the non matching alleles, where the bits to be exchanged are chosen at random without replacement. This way, it guarantees that the two offspring are always at the maximum Hamming distance from their two parents, thus proposing the introduction of a high diversity in the new population and lessening the risk of premature convergence. The recombination, disruptive crossover, replaces classical mutation plus crossover in a genetic algorithm. This crossover assures ergodicity of the procedure, [53].

- *An incest prevention mechanism*: During the reproduction step, each member of the parent (current) population is randomly chosen without replacement and paired for mating. However, not all these couples are allowed to cross over. Before mating, the Hamming distance between the potential parents is calculated and if half this distance does not exceed a difference threshold D, they are not mated and no offspring coming from them is included in the offspring population. The aforementioned threshold is usually initialized depending on the chromosome length. The recommended value in [22] is $L/4$, being L the chromosome length. If no offspring is obtained in one generation, the difference threshold is decremented by one.

  The effect of this mechanism is that only the more diverse potential parents are mated, but the diversity required by the difference threshold automatically decreases as the population naturally converges.

- *A restart process*: it substitutes the standard genetic algorithm mutation and it is only applied when the population has converged. The difference threshold is considered to measure the stagnation of the search, which happens when it has dropped to zero and several generations have been run without introducing any new individual in the population. The population is reinitialized by considering the best M individuals as the first chromosomes of the new population and generating the remainding chromosomes by randomly flipping a percentage (divergence rate DR) of the bits belonging to the best global individual. The recommended values in [22] for the restart parameters are: $M = 1$ and $DR = 0.35$.

Fig. 7 shows the main procedures involved in the CHC algorithm used in this study. For a general description see [22]. The parameters affecting the evolution of the CHC algorithm for which the user must provide a value are:

- *Population size* (PS): number of individuals in the population evolving in the search process.
- *Divergence rate* (DR): percentage of the best global solution found used as a pattern to construct the new population in the restart stage. Values are in $[0, 1]$.
- *Difference threshold* (D): maximum similarity degree allowed between two individuals in the population in the crossover stage. Values range in [0,L/2], where L is the length of the string that encodes the individuals in the population.
- *Best individuals* (M): number of best individuals considered when the population is reinitialized.

The CHC algorithm returns the individual in the population with the best evaluation.

4 Parameter settings for CHC and PBIL

In previous works, [33, 36], we conducted a preliminary study to tune CHC and PBIL algorithms in order to obtain the best performance when they are used to solve the Root Identification Problem. The problem size considered there was represented by a search space limited by 2^{18} solutions. An empirical methodology along with an statistical ANalysis Of VAriance (ANOVA), [11, 15, 13], was applied.

In this paper, a parameter setting for each problem size and evolutionary algorithm is estimated. To do so, a simple statistical method for predicting parameter settings for CHC and PBIL is used: the simple linear regression method, [20, 61].

The sample used to obtain expressions for parameter setting estimation is constituted by the best parameter settings obtained in previous works by ANOVA. The studies made in [33, 36] were complemented with a new work, [34], corresponding to search spaces limited by 2^{19} and 2^{20} solutions.

We first briefly describe the simple linear regression method. Then, we present the experimental design to obtain a parameter setting for each problem size and evolutionary algorithm. Finally, the parameter setting models for each algorithm will be presented.

```
Procedure CHCalgorithm

  INPUT
        TMAX: Number of iterations
        PS: Population size
        L: Chromosome length
        D: Difference threshold
        DR: Divergence rate
        M: Number of best individuals for restarting
  OUTPUT
        C: Best individual in population

  t := 0
  InitializePopulation (P_t)
  EvaluatePopulation(P_t)
  # Algorithm evolution
  while not termination condition is satisfied do
        t := t + 1
        # Incest prevention
        SelectForRecombination (P_{t-1}, C_t)
        # HUX crossover
        Recombine (C_t, C'_t)
        EvaluatePopulation(C'_t)
        # Elitist selection
        SelectForSelection (P_{t-1}, C'_t, P_t)
        if Equals (P_{t-1}, P_t) then
          DecrementDifferenceThreshold (D)
        if D < 0 then
          # Restart process
          DivergePopulation
          InitializeConvergenceCount(D,DR,L)
EndProcedure

Procedure DivergePopulation
  INPUT
        PS : Population size
        L : Chromosome length
        DR : Divergence rate
        P_t : Current population
        P_{t-1} : Previous population
        M : Number of best individuals for restarting
  OUTPUT
        P : Population

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

Figure 7: Pseudocode for CHC algorithm.

4.1 The simple linear regression method

Regression models, [17, 27], study the quantitative stochastic relationship among an interest variable and a set of explanatory variables. When the relationship among an interest variable, response variable or dependent variable (Y) and a set of independent or regressor variables ($X_1, X_2, \ldots, X_n$) is studied, a stochastic relationship exists among the response variable and the regressor variables if the next expression is satisfied,

$$Y = m(X_1, X_2, \ldots, X_n) + \varepsilon \tag{2}$$

being m the unknown regression function and ε a random variable of mean zero (the observation error).

The basic objective of studying a regression model is to estimate the regression function, m, and the probabilistic model followed by the random error ε, that is, to estimate the distribution function F_ε for the error variable. That estimation is made by a sample of the variables on study, $\{((X_{1,i}, X_{2,i}, \ldots, X_{n,i}), Y_i)\}_{i=1}^{M}$.

Linear regression, [20, 61], is a form of regression analysis in which the relationship between one or more independent variables and another variable, called dependent variable, is modeled by a least squares function, called linear regression equation. This function is a linear combination of one or more model parameters, called regression coefficients. A linear regression equation with one independent variable represents a straight line. The results are subject to statistical analysis.

The most simple regression model is the simple regression linear model, which studies the linear relationship between a response variable (Y) and a regressor variable (X) using a sample $\{(x_i, Y_i)\}_{i=1}^{M}$. The next model is followed,

$$Y_i = \alpha_0 + \alpha_1 x_i + \varepsilon_i \quad \forall i = 1, 2, \ldots, M \tag{3}$$

where α_0 (intercept) and α_1 (slope) are the parameters of the model.

Next, the basic assumptions for applying the simple linear regression model are detailed:

- The regression function is linear,

$$m(x_i) = E(Y/x_i) = \alpha_0 + \alpha_1 x_i \quad \forall i = 1, 2, \ldots, M \tag{4}$$

  or, equivalently,

$$E(\varepsilon_i) = 0 \quad \forall i = 1, 2, \ldots, M \tag{5}$$

- The variance is constant (homocedasticity),

$$Var(Y/x_i) = \sigma^2 \quad \forall i = 1, 2, \ldots, M \tag{6}$$

or, equivalently,

$$Var(\varepsilon_i) = \sigma^2 \quad \forall i = 1, 2, \ldots, M \tag{7}$$

- A normal distribution is followed,

$$Y/x_i \sim N(\alpha_0 + \alpha_1 x_i, \sigma^2) \quad \forall i = 1, 2, \ldots, M \tag{8}$$

or, equivalently,

$$\varepsilon_i \sim N(0, \sigma^2) \quad \forall i = 1, 2, \ldots, M \tag{9}$$

- The observations Y_i are independent. Under the normality hypothesis, this is equivalent to

$$Cov(Y_i, Y_j) = 0, \quad \text{if } i \neq j \tag{10}$$

This hypothesis in function of the errors and under the normality hypothesis is equivalent to

$$Cov(\varepsilon_i, \varepsilon_j) = 0, \quad \text{if } i \neq j \tag{11}$$

In the simple linear regression model three parameters must be estimated: the regression line coefficients, α_0 and α_1, and the poblational variance, σ^2. Multiple methods for computing those estimators exist, e. g. the maximum verosimility method, [4], or the least squares method, [52]. In our case, the latter will be used.

The minimum squared estimators, denoted as $\hat{\alpha}_0$ and $\hat{\alpha}_1$, are obtained, defining the residuals as the difference between observed value and expected value, $\{e_i = y_i - \hat{y}_i\}_{i=1}^M$, by minimizing the sum of the squared residuals,

$$\Psi(\alpha_0, \alpha_1) = \sum_{i=1}^{M} e_i^2 = \sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} (y_i - (\hat{\alpha}_0 + \hat{\alpha}_1 x_i))^2 \tag{12}$$

The estimators are then defined by the next expressions,

$$\begin{array}{l} \hat{\alpha}_0 = \bar{y} - \hat{\alpha}_1 \bar{x} \\ \hat{\alpha}_1 = \frac{s_{XY}}{s_x^2} \end{array} \tag{13}$$

being $\bar{x}$ and $\bar{y}$ the sample means for X and Y, respectively, s_x^2 the sample variance for X and s_{XY} the sample covariance between X and Y.

The residual variance, $\hat{s}_R^2$, is used as the poblational variance estimator, σ^2,

$$\hat{s}_R^2 = \frac{1}{M-2}\sum_{i=1}^{M} e_i^2 \tag{14}$$

where $M-2$ are the degrees of freedom of the residuals.

To assess if a significant relationship between X and Y exists, the next contrast have to be verified,

$$C_1 : \begin{Bmatrix} H_0 & \alpha_1 = 0 \\ H_a & \alpha_1 \neq 0 \end{Bmatrix} \tag{15}$$

If the contrast C_1 does not allow to reject the null hypothesis H_0 (the slope of the regression line is null), the linear relationship between X and Y will not exist.

In the same way, the contrast C_0 have to be verified,

$$C_0 : \begin{Bmatrix} H_0 & \alpha_0 = 0 \\ H_a & \alpha_0 \neq 0 \end{Bmatrix} \tag{16}$$

Different possibilities exist to test these hypothesis. The most extended one is the T-test, [41].

Moreover, to verify if the model is significant, it is necessary to decompose the response variable variability in the variability explained by the model and the residual variability. Under the hypothesis that a linear relationship between the response variable and the regressor variable exists, the objective is to test the next hypothesis contrast,

$$C_F : \begin{Bmatrix} H_0 & E(Y/X = x) = \alpha_0 \\ H_a & E(Y/X = x) = \alpha_0 + \alpha_1 x \end{Bmatrix} \tag{17}$$

If the null hypothesis is accepted, the regressor variable does not have any influence and no linear relationship exists between both variables. On the contrary case, the response variable is linearly dependent with respect to the regressor variable. That hypothesis contrast is tested by ANOVA, where F statistic gives a global measure of the regression goodness.

It is important to have a measure which indicates the goodness–of–fit, describing how well the regression line fits the set of observations. The determination

coefficient, R^2, is usually used for it and is defined as follows,

$$R^2 = \frac{scE}{scG} = \frac{\sum_{i=1}^{M}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{M}(y_i - \bar{y})^2} \tag{18}$$

The numerator is called explained sum of squares (scE) and the denominator is denoted as global sum of squares (scG). Given that $scE < scG$, the R^2 value belongs to the interval $[0, 1]$.

Besides, the Pearson correlation coefficient r measures the strength and direction of a linear relationship between the X and Y variables and is defined as follows,

$$r = sign(\hat{\alpha}_1)\sqrt{R^2} \tag{19}$$

The r value ranges from -1 to 1. If $r = 1$ exists an exact increasing linear relationship. If $r = -1$ exists an exact decreasing linear relationship.

4.1.1 Problems for fitting a simple linear regression model

Fitting a simple linear regression model presents different problems due to either no linear relationship among the variables or a violation of the basic hypothesis. Those problems are the following:

- *Lack of linearity.* Either the relationship between the variables is not linear or outstanding explanatory variables have not been included in the model. However, there are some cases where the relationship among the variables is not linear but the linear regression model is still valid. The relationship can become linear by either a variable X or a variable Y or both X and Y variables transformation. Transforming the model variables can solve problems like lack of normality or heterocedasticity, problems which will be described next. See Table 1.

- *Presence of outliers.* An observation with a large residual is called an atypical datum. The standarized residuals (r_i), computed by a normalisation of the ordinary residuals (e_i), are used to identify the observations having large residuals. It is easy to detect the outliers by observing a scatter plot of the sample and the corresponding fitted line.

- *Lack of normality.* Model residuals do not follow a normal distribution. Box plots, Percentile–Percentile plots and Histograme of standarized residuals provide information about their distribution. See [59] and [28] to check details about those graphical plots.

- *Heterocedasticity.* The variance of the residuals is not constant. The plot of standarized predictions versus standarized residuals $(\hat{y}_i, r_i)$ gives the necessary information about the model hypothesis verification: linearity, homocedasticity and outliers. See [59] and [28].

Model name	Model equation	X Transform.	Y Transform.
Simple	$Y = \alpha_0 + \alpha_1 X$	X	Y
Hyperbolic	$Y = \alpha_0 + \alpha_1 / X$	$1/X$	Y
Inverse	$Y = 1/(\alpha_0 + \alpha_1 X)$	X	$1/Y$
Double inverse	$Y = 1/(\alpha_0 + \alpha_1 / X)$	$1/X$	$1/Y$
Exponential	$Y = e^{\alpha_0 + \alpha_1 X}$	X	$\ln Y$
Logarithmic	$Y = \alpha_0 + \alpha_1 \ln X$	$\ln X$	Y
Multiplicative	$Y = \alpha_0 X^{\alpha_1}$	$\ln X$	$\ln Y$
X Sqrt	$Y = \alpha_0 + \alpha_1 \sqrt{X}$	$\sqrt{X}$	Y
Y Sqrt	$\sqrt{Y} = \alpha_0 + \alpha_1 X$	X	$\sqrt{Y}$
S Curve	$Y = e^{\alpha_0 + \alpha_1 / X}$	$1/X$	$\ln Y$

Table 1: Most common transformations for X and Y variables in linear regression model.

- *Autocorrelation.* Dependency among observations exists. The Durbin–Watson test, [5], can be used to detect it.

4.2 Experimental design

Given a problem size and an evolutive algorithm, specific values must be chosen for a set of parameters which determine the evolution of the algorithm. One of the main difficulties of applying an evolutionary algorithm to a given problem is to decide on an appropiate set of parameter values. Those values are commonly chosen in practice by trial and error, tuned by hand, or taken from other fields, [43].

In previous works, [33, 36, 34], the studied algorithms, CHC and PBIL, have been statistically optimized to be applied to the Root Identification Problem by ANOVA. A randomly generated representative benchmark was considered, [63]. The search spaces were limited by 2^{18}, 2^{19} or 2^{20} solutions. In [33, 36], 29 instances with a search space limited by 2^{18} solutions were considered, while in [34], 12 instances were considered for each of the search space sizes constituted by 2^{19} and 2^{20} solutions.

To obtain a parameter setting for each problem size and each algorithm, the simple linear regression model has been used having the best parameters selected in previous works as sample. For each different parameter, a different simple linear regression model will be yielded. The independent variable X will be the problem size and the dependent variable Y will be the parameter value. Therefore, for the CHC algorithm, four different models will allow to estimate a parameter setting, one model per parameter: Population Size (PS), Divergence Rate (DR), Difference Threshold (D) and Best Solutions (M). In this way,

for the PBIL algorithm, four different models will also be considered, one per parameter: Population Size (PS), Learning Rate (LR), Mutation Probability (MP) and Mutation Shift (MS).

Before estimating the values for α_0 and α_1, see Equation 3, by the least squares method in each model, the simple linear regression model assumptions must be tested.

First, the scatter plots representing the available sample for each parameter must be analyzed. If a lack of linearity appears, we will try to apply some transformations over the independent variable, the dependent variable or both variables, see Table 1. When a scatter plot representing the original or transformed variables shows linearity, homocedasticity and no presence of outliers, the estimators $\hat{\alpha}_0$ and $\hat{\alpha}_1$ will be computed, see Equation 13. The normality assumption will not be violated because of the sample size, taking into account the Limit Central Theorem, [11].

After estimating the regression line coefficients for each parameter, the model will have to be validated. The goodness of the model have to be analyzed by the determination and the Pearson correlation coefficients. See Equations 18 and 19, respectively. Then, the statistical significance of the estimated coefficients must be validated individually by the T-test, see Equations 15 and 16, and as a whole by ANOVA, see Equation 17.

Finally, the residuals analysis will test a posteriori the model hypothesis. The observation of the Normal Percentil–Percentil plots for the standarized residuals, the standarized residual histogrames and the standarized predictions vs. standarized residuals plots will allow to demonstrate those hypothesis, [28, 59].

4.3 Parameter setting models

Tables 2 and 3 present respectively the parameter setting models for CHC and PBIL obtained by the simple linear regression method.

As can be seen, several transformations have been required to keep the linearity assumption. In CHC algorithm, only a logarithmic transformation has been applied to Divergence Rate parameter. In PBIL algorithm, an exponential transformation has been applied to Learning Rate parameter and a X squared root transformation has been applied to mutation parameters: Mutation Probability and Mutation Shift. See Table 1.

After that transformations, the problem size versus parameter value scatter plots kept the model assumptions. The validation of the goodness of the different models gives as a result the determination and the Pearson correlation coefficients presented in Tables 4 and 5. The R^2 values are large and close to 100% and the r values are close to -1 or 1. Each model is well-supported by their correspondent sample. Finally, the T-test, ANOVA and residuals analysis assessed the model validity.

Parameter (P)	Parameter setting model equation
Population Size (PS)	$\widehat{PS} = \lfloor 19.780700 + 0.380564T \rfloor;\ \forall T > 0;\ T \in N$
Difference Threshold (D)	$\widehat{D} = \lfloor 0.841461 + 0.211591T \rfloor;\ \forall T > 0;\ T \in N$
Divergence Rate (DR)	$\widehat{DR} = \min(0.135549 + 0.062776 \ln T, 1);\ \forall T > 0;\ T \in N$
Best Individuals (M)	$\widehat{M} = \lfloor 1.360900 + 0.013641T \rfloor;\ \forall T > 0;\ T \in N$

Table 2: Parameter setting models for CHC.

See [37] for a detailed and exhaustive explanation of the complete regression analysis. The study of each parameter nature supposes the final parameter setting equations presented in Tables 2 and 3.

5 Modelling PBIL and CHC behavior by Run-Length Distributions

The evolution of metaheuristics in the search process supposes to take decisitions where randomness appears. The knowledge of the random variable run-time can provide valuable information to analyze and characterize the algorithm behavior. It will be possible to obtain indications for improving the algorithm performance and comparing algorithms in a suitable way. See [8, 49, 62].

To acquire empirical knowledge about the run-time distribution (RTD) of an algorithm when it is used to solve a specific problem instance, an empirical RTD can be estimated by the results from several algorithm runs, [31]. Probably, the empirically observed RTD can be aproximated to a well-known distribution function corresponding to probability theory by goodness-of-fit methods, [56, 28, 17].

If a similar behavior in every instance studied is observed, the underlying RTD will characterize the run-time behavior over the sort of problem instances afforded. Thus, the RTD analysis can provide indications for improving an algorithm and the comparison among algorithms can be easier.

In the present Section the run-time behavior for evolutionary algorithms, particularized to CHC and PBIL algorithms, is analyzed when they are applied to

Parameter (P)	Parameter setting model equation
Population Size (PS)	$\widehat{PS} = \lfloor 7.739150 + 1.650500T \rfloor;\ \forall T > 0;\ T \in N$
Learning Rate (LR))	$\widehat{LR} = e^{-1.633900 - 0.002972T};\ \forall T > 0;\ T \in N$
Mutation Probability (MP)	$\widehat{MP} = \min(0.011871 + 0.003236\sqrt{T}, 1);\ \forall T > 0;\ T \in N$
Mutation Shift (MS)	$\widehat{MS} = \min(0.017716 + 0.001375\sqrt{T}, 1);\ \forall T > 0;\ T \in N$

Table 3: Parameter setting models for PBIL.

Parameter	R^2 (%)	r
Population Size (PS)	93.8806	0.9689
Difference Threshold (D)	84.6110	0.9198
Divergence Rate (DR)	64.5952	0.8037
Best Individuals (M)	82.9036	0.9105

Table 4: Goodness of the model for CHC parameters.

solve the Root Identification Problem. Then, a theoretical statistical distribution which describes that behavior is identified as a behaviour model for each algorithm. Finally, the properties which determine the algorithm behaviour and performance are described by the identified model.

5.1 Algorithm performance measure: Run-time vs. Run-length distributions

Metaheuristics, specifically CHC and PBIL, make use of randomness during their runs. The run-time is a random variable and knowledge about its distribution will be obtained by empirically taking samples after running an algorithm several times. Based on such samples, assumptions can be made about the run-time. Those assumptions can be validated by statistical hypothesis tests.

The metaheuristic run-time depends on the problem instance. The search space is different for each instance. Thus, a run-time distribution must be estimated for each instance and algorithm in order to detect differences in the run-time behavior over the set of instances. However, it does not avoid that the conclusions extracted can determine the run-time distribution type corresponding to the global problem, as has been demonstrated in [31].

Instead of actually measuring algorithm runs in terms of CPU-time, it is often preferable to use representative operation counts as a more machine independent measure of an algorithm's performance, [3]. In this way, using a suitable cost model for the algorithm operations facilitates the comparison among different architectures. In such a case, run-length distributions (RLDs) instead of run-time distributions are considered. Henceforth, we will present the performance

Parameter	R^2 (%)	r
Population Size (PS)	80.9857	0.8999
Learning Rate (LR)	72.3596	-0.8506
Mutation Probability (MP)	97.9928	0.9899
Mutation Shift (MS)	97.8747	0.9893

Table 5: Goodness of the model for PBIL parameters.

of algorithms by RLDs.

For our problem, the computational cost of an evolutionary algorithm corresponds, basically, to computing the number of extra predicates fulfilled by the set of solution instances in the population, [45, 44]. Therefore we choose computing the number of extra predicates fulfilled by a solution instance as the run-length unit.

It is important to notice that most of the algorithms present a parameter which limits the run-time. In our case, the maximum number of evaluations allowed before declaring a run as a failure is the maximum run-length, rl_{max}. Since our context requires real time interaction, [39, 60], the limit value rl_{max} has to be set to a suitable value.

Practically, we measure empirical RLDs by running the respective algorithm for n times on a given problem instance up to some suitable cutoff value and recording for each successful run the number of steps (number of evaluations) required to find a solution. The empirical run-length distribution is the cumulative distribution associated with the observations. More formally, let $rl(j)$ denote the run length for the j-th successful run over an instance I, then the cumulative empirical RLD is defined by the next Equation,

$$\hat{P}_I(rl \leq i) = \mid \{j \parallel rl(j) \leq i\} \mid /n \tag{20}$$

In our case, a successful run is a run where a solution which fulfills all the extra predicates defined by the user is found.

As a result, the averaged empirical RLD for an algorithm over a set T of NI instances is defined as follows,

$$\hat{P}_T(rl \leq i) = \frac{1}{NI} \sum_{I=1}^{NI} \hat{P}_I(rl \leq i) \tag{21}$$

The validity of the averaged empirical RLD depends on the closeness among the different empirical RLDs. The averaged empirical RLD will be used to summarize the performance of an algorithm on a specific problem size T.

5.2 Goodness-of-fit

To characterize in a general way the evolution and the performance of evolutionary algorithms when they are applied to solve the afforded problem, a set of graphical and analytical tests will be used to check the goodness-of-fit of empirical RLDs to well-known continuous statistical distributions.

Each random variable, [11], in our case the run-time represented by the run-length, has a probability distribution which characterizes it completely. Let

X be a random variable, the probability distribution for X can be described univocally by its cumulative distribution function $F(x)$, defined as follows,

$$F(x) = Pr[X \leq x] \tag{22}$$

for any x in $\mathbb{R}$. A distribution is called continuous if its cumulative distribution function is continuous. Because of the continuous nature of run-time, continuous distributions will be used.

If the definition of empirical RLD, see Equation 20, is checked, it can be seen that it is equivalent to the cumulative distribution function definition. Therefore, if a continuous statistical distribution fits the results of the runs of an evolutionary algorithm on a specific instance, the corresponding cumulative distribution function fits to the algorithm empirical RLD.

The goodness-of-fit of a statistical model describes how well it fits a set of observations, [17]. Measures of goodness-of-fit typically summarize the discrepancy between observed values and the values expected under the model in question. Such measures can be used in statistical hypothesis testing, to test whether two samples are drawn from identical distributions, or whether outcome frequencies follow a specified distribution. In our case, we wish to know if the observed values, corresponding to the run-length of an algorithm, say PBIL or CHC, on the Root Identification Problem, fit to a well-known theoretical continuous distribution.

The goodness-of-fit tests measure the compatibility of a given random sample with a specific theoretical probability distribution function. We usually distinguish between two different types of tests: graphical tests, [28], and cuantitative tests, [56].

Graphical tests determine if the observed data fit to a hypothetical distribution based on a subjective visual test of the data. The most common graphical test is the probabilities plot: Percentile-Percentile plot or Quantile-Quantile plot.

In quantitative tests is tested the null hypothesis of being equivalent the data's underlying distribution and a statistical distribution. Given a theoretical statistical distribution, to decide if the null hypothesis is rejected, the parameters of such a distribution are estimated from the observed data by a parameter estimation method. The preferred method for parameter estimation is the maximum verosimility method, [4]. Among the most common quantitative tests for goodness-of-fit are: the Chi-square test, [57], the Kolmogorov-Smirnov test, [12] and the Anderson-Darling test, [58]. These three tests have been used in this work.

5.3 Finding a statistical model for empirical RLDs

We want to model an algorithm performance when it is used to solve the Root Identification Problem. The aim is to generate the empirical RLDs and to test

Problem size	2^{18}	2^{19}	2^{20}	2^{25}	2^{30}	2^{35}
Instances	29	12	12	10	10	10

Table 6: Features of the benchmark used for finding a statistical model.

if they fit to a cumulative distribution function corresponding to a well-known continuous statistical distribution function. In this way, different instances corresponding to a set of problem sizes will be randomly generated, see Table 6. This benchmark is available in [63].

A set of statistical continuous distributions has been selected taking into account the nature of the random variable analyzed, [11, 35]. Those distributions are: Exponential, Weibull, Gamma, Erlang, Normal, Laplace and Cauchy. To have enough sample in order to obtain the empirical RLD for each instance, the number of runs for each algorithm on each specific instance is $n = 50$.

For each different theoretical statistical continuous distribution and empirical RLD, graphical and cuantitative goodness-of-fit tests has been applied: Percentile-Percentile test representing graphical tests and Chi-squared test, Kolmogorov-Smirnov test and Anderson-Darling test representing cuantitative tests.

The results of the goodness-of-fit tests applied to the benchmark are presented in [35] for search spaces bounded by 2^{18}, in [34] for search spaces bounded by 2^{19} and 2^{20}, and in [38] for the remaining problem sizes. For all the problem instances constituting the benchmark there is a distribution for which quantitative methods do not allow to reject the null hypothesis with a 95% confidence level. The distribution which models the performance for CHC and PBIL is the Gamma distribution. Percentile-Percentile tests have shown visually such fit. For an example of the resulting fit, see Figure 8.

Taking into account the visual closeness among the different empirical RLDs for each problem size, an averaged empirical RLD for each problem size, see Equation 21, and algorithm has been computed. Then, the goodness-of-fit for each averaged empirical RLD to the aforementioned theoretical statistical distributions has been tested in the same way that the individual empirical RLDs. Again, the distribution which models the averaged empirical RLDs has been the Gamma distribution. Figure 9 presents the averaged Gamma RLD for a problem size (as an example, the problem instances with a search space bounded by 2^{35} solutions have been taken) and the different Gamma RLDs which correspond to the different instances. In [34] and [38], the complete analysis and goodness-of-fit test application for all the problem sizes considered in the benchmark are presented.

Figure 10 presents the averaged Gamma RLD for all the problem sizes constituting the benchmark. Generally, the larger the problem size the larger the run-length needed to obtain a same success probability (solution quality). The cumulative distribution function of the Gamma distribution which fits to the av-

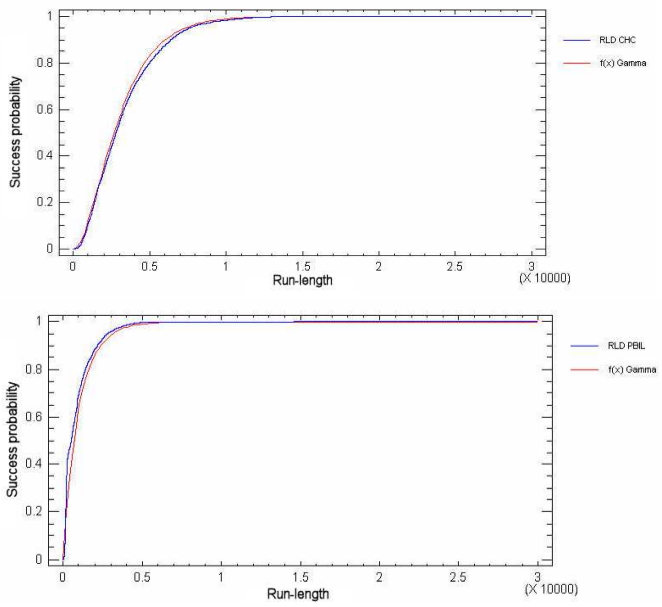

Figure 8: Examples of fit between empirical RLDs and the cumulative distribution function of a theoretical statistical distribution: CHC (top) and PBIL (bottom).

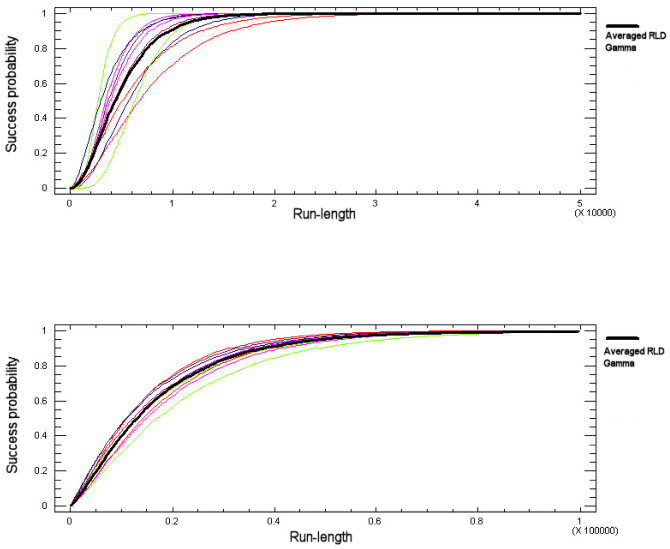

Figure 9: Example of averaged Gamma RLD for a problem size: CHC (top) and PBIL (bottom).

eraged empirical RLD for a problem size is proposed as the performance model for the algorithm applied in each case.

5.4 The Gamma performance model

In probability theory and statistics, the Gamma distribution, [17], is a two-parameter family of continuous probability distributions. It has a scale parameter θ and a shape parameter k. If k is an integer then the distribution represents the sum of k independent exponentially distributed random variables, each of which has a mean of θ.

The probability density function of the Gamma distribution can be expressed in terms of the Gamma function parameterized in terms of a shape parameter k and a scale parameter θ,

$$f(x;k,\theta) = x^{k-1}\frac{e^{-x/\theta}}{\theta^k\,\Gamma(k)} \text{ for } x > 0 \tag{23}$$

where $k > 0$ is the shape parameter, $\theta > 0$ is the scale parameter and $\Gamma(k)$ is the Gamma function applied on k parameter.

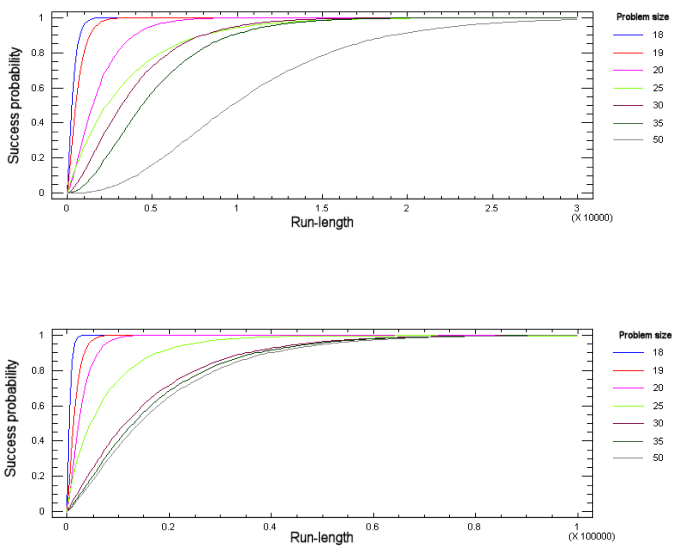

Figure 10: Averaged Gamma RLDs for all problem sizes: CHC (top) and PBIL (bottom).

The Gamma function is an extension of the factorial function to real and complex numbers. For a complex number z with positive real part it is defined by the next expression,

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, \mathrm{d}t \tag{24}$$

which can be extended to the rest of the complex plane, excepting the non-positive integers.

Alternatively, the Gamma distribution can be parameterized in terms of an inverse scale parameter $\beta = 1/\theta$.

The cumulative distribution function, used here to represent a RLD, can be expressed in terms of the incomplete Gamma function,

$$F(x; k, \beta) = \int_0^x f(u; k, \beta)\, du = \frac{\gamma(k, x\beta)}{\Gamma(k)} \tag{25}$$

where k and β are respectively the shape and inverse scale parameters and $\gamma(k, x\beta)$ is the lower incomplete Gamma function applied on k and $x\beta$.

The incomplete Gamma function is defined as an integral function of the same integrand as the Gamma function. There are two varieties of the incomplete Gamma function: the upper incomplete Gamma function is for the case that the lower limit of integration is variable and the lower incomplete Gamma function can vary the upper limit of integration.

The upper incomplete gamma function is defined as:

$$\Gamma(a, x) = \int_x^\infty t^{a-1}\, e^{-t}\, dt \tag{26}$$

The lower incomplete gamma function is defined as:

$$\gamma(a, x) = \int_0^x t^{a-1}\, e^{-t}\, dt \tag{27}$$

In both Equations 26 and 27, a is a complex parameter with a positive real part.

That a random variable X is Gamma-distributed with inverse scale β and shape k is denoted $X \sim Gamma(k, \beta)$. In our case, the random variable considered is the run-length, which will be denoted as RL.

Given a problem size T belonging to the benchmark described before, an averaged Gamma RLD has been estimated by the maximum verosimility method used in quantitative goodness-of-fit tests. Let $F(rl; \hat{k}_T, \hat{\beta}_T)$ the corresponding cumulative distribution function and $Gamma(\hat{k}_T, \hat{\beta}_T)$ the distribution.

Taking into account the estimated Gamma distribution for a problem size, the following descriptive parameters related to the run-length random variable can be defined:

- Mean ($\overline{RL}_T$):

$$\overline{RL}_T = \hat{k}_T/\hat{\beta}_T \tag{28}$$

- Standard deviation (σ_{RL_T}):

$$\sigma_{RL_T} = \sqrt{\hat{k_T}}/\hat{\beta}_T \tag{29}$$

- j-th Percentile (P_T^j):

$$P_T^j = j/100 \int_0^\infty t^{rl-1} e^{-t}\,\mathrm{d}t \tag{30}$$

- Median (Me):

$$Me = P_T^{50} = 1/2 \int_0^\infty t^{rl-1} e^{-t}\,\mathrm{d}t \tag{31}$$

Notice that the computation for the percentile and the median are based on the Gamma function definition given above.

Furthermore, given the averaged RLD Gamma for a problem, that is the cumulative distribution function, we can obtain the success probability (solution quality) which will be obtained for a certain run-length (run-time) of an algorithm. Also, in a inverse way, the run-length needed for an algorithm for obtain a certain success probability (solution quality) can be computed.

Formally, let T be a problem size, $F_T(rl; \hat{k}_T, \hat{\beta}_T)$ the cumulative function distribution which models the performance of an algorithm A on the problem size T and rl_{suc} the run-length available for that algorithm. The estimated success probability p_{suc} given by the Gamma model will be defined by the following equation,

$$p_{suc} = F_T(rl_{suc}; \hat{k}_T, \hat{\beta}_T) = \int_0^{rl_{suc}} f(u; \hat{k}_T, \hat{\beta}_T)\,du = \frac{\gamma(\hat{k}_T, rl_{suc}\hat{\beta}_T)}{\Gamma(\hat{k}_T)} \tag{32}$$

On the other hand, let T be a problem size, $F_T^{-1}(rl; \hat{k}_T, \hat{\beta}_T)$ the inverse of the cumulative function distribution which models the performance of an algorithm A on the problem size T and p_{suc} the solution quality required. The estimated run-length rl_{suc} given by the Gamma model will be defined by the following equation,

$$rl_{suc} = F^{-1}(p_{suc}; \hat{k}_T, \hat{\beta}_T) = \{x_{suc} : F(x_{suc}; \hat{k}_T, \hat{\beta}_T) = p_{suc}\} \tag{33}$$

Calculating the Gamma direct and inverse cumulative distribution functions and likewise calculating the Gamma function and incomplete Gamma functions is computationally efficient and can be made by different aproximated algorithms and iterative methods for optimization. See [50] as a main reference in the literature. See [14, 55] for details about the Gamma function and [2, 18] for details about the incomplete Gamma functions.

6 Prediction and validation of the performance model

In the previous Section, the Gamma model has been demonstrated to fit to the empirical RLDs and to the averaged empirical RLDs corresponding to search space sizes up to 2^{50} solutions. It would be very useful if, given a problem size, a performance model was available to provide an idea of the time requirements needed to reach a certain solution quality, and likewise to provide an aproximated solution quality to be obtained for a certain time.

In this Section, some expressions will be proposed to predict the performance model for evolutionary algorithms, particularized to CHC and PBIL algorithms, when they are solving the Root Identification Problem. Then, the predicted performance models will be validated for search space sizes up to 2^{100} solutions.

6.1 Prediction of the Gamma performance model

As have been said in Section 5.4, the Gamma model is defined in terms of two parameters: a scale factor, β $(1/\theta)$, and a shape factor, k. If an expression for predicting the value of each parameter according to the problem size is obtained, the performance model can be estimated.

The irregular variations in the Gamma parameters for search space sizes up to 2^{50} solutions, [37, 38], make such prediction difficult. However, if two of the basic descriptive measures for the models like the mean and the standard deviation are observed, see Tables 7 and 8, a linear relationship is clearly noticeable. The Gamma scale and shape parameters can be defined in terms of the Gamma model mean and standard deviation, see Equations 35 and 34, so the Gamma model will be predicted by defining expressions for those basic descriptive measures.

- Mean ($\overline{RL}$):

$$\overline{RL} = \frac{k}{\beta} \tag{34}$$

- Standard deviation (σ_{RL}):

$$\sigma_{RL} = \frac{\sqrt{k}}{\beta} \tag{35}$$

Size	Mean	Std. dev.
2^{18}	348.544335	292.998905
2^{19}	632.154507	520.875174
2^{20}	1885.949367	1545.081771
2^{25}	3417.728408	3509.732858
2^{30}	3914.569717	3035.836581
2^{35}	5126.840255	3418.691291
2^{50}	10829.827060	6219.448981

Table 7: CHC. Values of the basic statistical measures in the averaged Gamma model for search space sizes up to 2^{50}.

Taking into account the existing linearity between the problem size and each basic statistical measure, the simple linear regression method has been applied to obtain the prediction expressions. The details, including the verification of a priori and a posteriori assumptions, are presented in [38]. Tables 9 and 10 present respectively the corresponding expressions for CHC and PBIL.

6.2 Validation of the Gamma performance model

Given an unknown large problem size, a Gamma RLD has been estimated to model the performance for each evolutionary algorithm, in our case CHC and PBIL. The aim now is to validate the reliability of the estimated models: model reliability to estimate the solution quality for a certain time and, in an inverse sense, model reliability to estimate the time required for a certain solution quality. First, the experimental design is presented. Then, the results of the validation are discussed.

Size	Mean	Std. Dev
2^{18}	563.158103	471.796710
2^{19}	1620.623656	1320.078059
2^{20}	2873.277778	2306.704310
2^{25}	7280.584924	8149.599308
2^{30}	15724.173050	14965.771580
2^{35}	17145.441020	15305.531910
2^{50}	18502.257000	16217.613280

Table 8: PBIL. Values of the basic statistical measures in the averaged Gamma model for search space sizes up to 2^{50}.

Parameter (P)	Linear model equation	$\hat{\alpha}_{0,P}$	$\hat{\alpha}_{1,P}$
Mean ($\overline{RL}$)	$\overline{RL} = \hat{\alpha}_{0,\overline{RL}} + \hat{\alpha}_{1,\overline{RL}}T$	-4936.36	308.173
Std. dev. (σ_{RL})	$\sigma_{RL} = \hat{\alpha}_{0,\sigma_{RL}} + \hat{\alpha}_{1,\sigma_{RL}}T$	-2107.86	169.024

Table 9: Linear regression coefficients estimation for basic statistical measures for CHC Gamma model.

6.2.1 Experimental design

To assess the Gamma model reliability, a benchmark containing four different search space sizes of up to 2^{100} solutions has been selected. For each of the sizes 2^{60}, 2^{70}, 2^{80} and 2^{100}, $NI = 10$ different instances have been randomly generated. The benchmark is available in [63]. Notice the huge computation requirements of such problem sizes.

The comparison between the empirical results coming from the algorithm runs and the results provided by the model, will give an idea of the reliability of such model. To select a parameter setting for each algorithm run, the expressions provided in Section 4 have been used. Because a suitable sample is needed to obtain significant empirical results, $n = 50$ runs of each algorithm over each instance have been made.

To make the analysis easier, a good solution quality has been selected, $p_{suc} = 0.9$. Given an algorithm, CHC or PBIL, a Gamma model is available for each problem size T, $Gamma_T(\hat{k}_T, \hat{\beta}_T)$, see Section 6.1. Each Gamma model presents a cumulative distribution function F_T (RLD) which defines the algorithm performance: run-length vs. success probability, see Section 5.

To validate a Gamma model $Gamma_T$ for each run j of an algorithm over an instance I of a problem size T, two differences have to be tested. First, the difference between the run-length estimated by the model to reach a solution quality $p_{suc} = 0.9$ and the run-length empirically required by the algorithm to obtain such a solution quality has to be computed. Second, the difference between $p_{suc} = 0.9$ and the quality obtained by the algorithm when the run-length limit is that estimated by the model to reach $p_{suc} = 0.9$ has also to be com-

Parameter (P)	Linear model equation	$\hat{\alpha}_{0,P}$	$\hat{\alpha}_{1,P}$
Mean ($\overline{RL}$)	$\overline{RL} = \hat{\alpha}_{0,\overline{RL}} + \hat{\alpha}_{1,\overline{RL}}T$	-8135.06	612.462
Std. Dev. (σ_{RL})	$\sigma_{RL} = \hat{\alpha}_{0,\sigma_{RL}} + \hat{\alpha}_{1,\sigma_{RL}}T$	-6767.08	538.613

Table 10: Linear regression coefficients estimation for basic statistical measures for PBIL Gamma model.

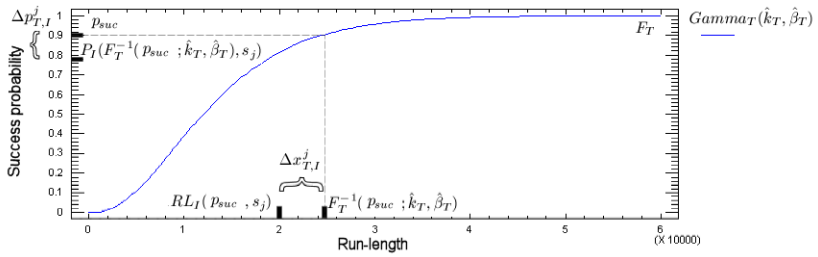

Figure 11: Differences between the Gamma model and the empirical results for an algorithm.

puted. Those differences have to be normalized in order to make comparisons among different problem sizes and algorithms. The first difference is presented in the following Equation,

$$\Delta x^j_{T,I} = \frac{|F_T^{-1}(p_{suc};\hat{k}_T,\hat{\beta}_T) - RL_I(p_{suc},s_j)|}{F_T^{-1}(p_{suc};\hat{k}_T,\hat{\beta}_T)} * 100,\ j \in \mathbb{N} \tag{36}$$

where $F_T^{-1}(p_{suc};\hat{k}_T,\hat{\beta}_T)$ is the run-length estimated by the Gamma model and $RL_I(p_{suc},s_j)$ is the required empirical run-length by an algorithm run over an instance I with an initial random seed s_j.

In an inverse sense, the second difference is defined as follows,

$$\Delta p^j_{T,I} = \frac{|p_{suc} - P_I(F_T^{-1}(p_{suc};\hat{k}_T,\hat{\beta}_T),s_j)|}{p_{suc}} * 100,\ j \in \mathbb{N} \tag{37}$$

where $P_I(F_T^{-1}(p_{suc};\hat{k}_T,\hat{\beta}_T),s_j)$ is the solution quality empirically obtained by an algorithm run over the instance I with an initial random seed s_j and having as a run-length limit to reach the required solution quality, p_{suc}, that estimated by the model, $F_T^{-1}(p_{suc};\hat{k}_T,\hat{\beta}_T)$.

Figure 11 graphically presents the differences for each algorithm run described in Equations 36 and 37.

Taking into account the random nature of the runs of an algorithm and the random nature of the instances generated for each problem size, the differences between the Gamma model and the empirical results of an algorithm must be expressed in terms of the $n = 50$ runs considered for each instance and the $NI = 10$ instances considered for each problem size T. To do so, based on Equations 36 and 37, the mean value for all the differences $(\Delta x^j_{T,I}, \Delta p^j_{T,I})$ corresponding to an instance I has been taken and denoted respectively as $\overline{\Delta x_{T,I}}$ and $\overline{\Delta p_{T,I}}$. Then, for each problem size, the averaged value of the differences per instance has been taken and denoted respectively as $\overline{\Delta x_T}$ and $\overline{\Delta p_T}$. See

Equations 38 and 39.

$$\overline{\Delta x_T} = \frac{1}{NI}\sum_{I=1}^{NI}\overline{\Delta x_{T,I}} \quad ; \quad \overline{\Delta x_{T,I}} = \frac{1}{n}\sum_{j=1}^{n}\Delta x_{T,I}^{j} \tag{38}$$

$$\overline{\Delta p_T} = \frac{1}{NI}\sum_{I=1}^{NI}\overline{\Delta p_{T,I}} \quad ; \quad \overline{\Delta p_{T,I}} = \frac{1}{n}\sum_{j=1}^{n}\Delta p_{T,I}^{j} \tag{39}$$

The variability among the difference values has been computed as the standard deviation among the difference values per instance I, denoted as $\sigma_{\Delta x_{T,I}}$ and $\sigma_{\Delta x_{T,I}}$. To summarize that variability, the averaged value has been calculated for each problem size T, denoted respectively as $\sigma_{\Delta x_T}$ and $\sigma_{\Delta p_T}$. See Equations 40 and 41.

$$\sigma_{\Delta x_T} = \frac{1}{NI}\sum_{I=1}^{NI}\sigma_{\Delta x_{T,I}} \quad ; \quad \sigma_{\Delta x_{T,I}} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(\overline{\Delta x_{T,I}} - \Delta x_{T,I}^{j}\right)^2} \tag{40}$$

$$\sigma_{\Delta p_T} = \frac{1}{NI}\sum_{I=1}^{NI}\sigma_{\Delta p_{T,I}} \quad ; \quad \sigma_{\Delta p_{T,I}} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(\overline{\Delta p_{T,I}} - \Delta p_{T,I}^{j}\right)^2} \tag{41}$$

For each algorithm, the larger the reliability of a Gamma model for a certain problem size, the lesser the values of the differences $\overline{\Delta x_T}$ and $\overline{\Delta p_T}$. Based on Equations 36, 37, 38 and 39, it is formally expressed as follows,

$$\lim_{\overline{\Delta x_T}\to 0} RL_I(p_{suc}, s_j) = F_T^{-1}(p_{suc}; \hat{k}_T, \hat{\beta}_T) \quad \forall j \in \mathbb{N}, \forall I \tag{42}$$

$$\lim_{\overline{\Delta p_T}\to 0} P_I(F_T^{-1}(p_{suc}; \hat{k}_T, \hat{\beta}_T), s_j) = p_{suc} \quad \forall j \in \mathbb{N}, \forall I \tag{43}$$

To summarize, for each algorithm (CHC and PBIL) and problem size $T = \{2^{60}, 2^{70}, 2^{80}, 2^{100}\}$ the corresponding Gamma model, $Gamma_T(\hat{k}_T, \hat{\beta}_T)$, has been identified. To validate such model, $n = 50$ runs has been made for each instance corresponding to a set of $NI = 10$ instances and the summary difference values $\overline{\Delta x_T}$, $\overline{\Delta p_T}$, $\sigma_{\Delta x_T}$ and $\sigma_{\Delta p_T}$ have been obtained.

6.2.2 Validation results

Table 11 presents the Gamma models to be validated. The estimation of the Gamma parameters has been described in Section 6.1. Tables 12 and 13 present

Model	CHC		PBIL	
$Gamma_T$	$\hat{k}_T$	$\hat{\beta}_T$	$\hat{k}_T$	$\hat{\beta}_T$
$Gamma_{2^{60}}$	2.84654	2.10015E-04	1.25413	4.38315E-05
$Gamma_{2^{70}}$	2.92692	1.75942E-04	1.26086	3.62971E-05
$Gamma_{2^{80}}$	2.98416	1.51346E-04	1.26560	3.09728E-05
$Gamma_{2^{100}}$	3.06025	1.18243E-04	1.27185	2.39470E-05

Table 11: Gamma models to be validated.

the validation of the Gamma models for CHC. The former shows the differences between the Gamma models and the empirical results in the run-length needed for a given solution quality. The latter shows the differences between the Gamma models and the empirical results in the solution quality obtained for the run-length predicted by the Gamma models to reach the good solution quality required ($p_{suc} = 0.9$).

All the differences are homogeneous both in mean and in standard deviation for all the problem sizes considered. The difference value for the different instances is in most of the cases under the 10% and slightly higher for the remaining ones. The mean difference is for all the problem sizes under the 10%. It is important to notice that as the problem size increases, the mean difference does not increase. The variability of the differences is similar among the different instances and problem sizes. There is no notorious differences between the success probability results and the run-length results.

The same results for PBIL are presented in Tables 14 and 15. There is no notorious differences between PBIL and CHC results. Table 16 presents a summary of the differences between the Gamma models and CHC and PBIL empirical results. The Gamma model fits to the CHC and PBIL behaviour without significant differences. The variability is very limited and the model is very reliable for each problem size if the instance is changed. Furthermore, increasing the

Δx_T	Problem size							
CHC	2^{60}		2^{70}		2^{80}		2^{100}	
Instance	$\overline{\Delta x_{2^{60},I}}$	$\sigma_{\Delta x_{2^{60},I}}$	$\overline{\Delta x_{2^{70},I}}$	$\sigma_{\Delta x_{2^{70},I}}$	$\overline{\Delta x_{2^{80},I}}$	$\sigma_{\Delta x_{2^{80},I}}$	$\overline{\Delta x_{2^{100},I}}$	$\sigma_{\Delta x_{2^{100},I}}$
1	7.27	5.97	7.36	5.12	8.68	4.99	9.87	4.87
2	9.92	5.89	9.64	5.60	7.11	5.67	7.20	4.94
3	7.08	5.27	12.38	7.52	7.50	5.34	6.82	3.81
4	8.86	5.89	7.36	5.58	7.56	5.24	8.02	3.33
5	10.38	5.83	12.39	5.68	10.44	5.61	9.41	5.91
6	9.86	6.72	7.94	5.92	9.56	5.54	6.55	4.02
7	11.82	5.65	8.43	5.46	8.35	4.69	7.23	4.99
8	9.95	5.41	8.87	5.83	9.59	5.71	7.75	4.73
9	12.38	5.34	8.75	5.05	8.33	5.82	8.99	5.02
10	10.68	5.30	10.58	6.01	7.73	6.54	6.98	5.33
Diff. summary	$\overline{\Delta x_{2^{60}}}$ 9.82	$\sigma_{\Delta x_{2^{60}}}$ 5.73	$\overline{\Delta x_{2^{70}}}$ 9.37	$\sigma_{\Delta x_{2^{70}}}$ 5.78	$\overline{\Delta x_{2^{80}}}$ 8.49	$\sigma_{\Delta x_{2^{80}}}$ 5.52	$\overline{\Delta x_{2^{100}}}$ 7.88	$\sigma_{\Delta x_{2^{100}}}$ 4.70

Table 12: Validation of Gamma model in predicting the run-length for CHC.

Δp_T	Problem size							
CHC	2^{60}		2^{70}		2^{80}		2^{100}	
Instance	$\overline{\Delta p_{2^{60},I}}$	$\sigma_{\Delta p_{2^{60},I}}$	$\overline{\Delta p_{2^{70},I}}$	$\sigma_{\Delta p_{2^{70},I}}$	$\overline{\Delta p_{2^{80},I}}$	$\sigma_{\Delta p_{2^{80},I}}$	$\overline{\Delta p_{2^{100},I}}$	$\sigma_{\Delta p_{2^{100},I}}$
1	8.76	6.04	8.13	6.35	6.77	4.38	10.56	4.68
2	8.70	5.88	10.00	5.36	5.07	5.10	9.29	5.52
3	8.52	6.38	11.56	5.85	6.18	5.47	9.80	4.78
4	7.56	5.28	9.73	6.48	5.57	4.66	6.23	4.98
5	10.47	5.44	10.63	5.39	9.08	5.18	10.16	5.94
6	8.83	5.52	8.70	5.91	6.62	4.11	6.78	5.03
7	11.00	5.91	11.63	6.03	5.17	4.03	5.03	4.55
8	7.81	4.80	9.07	5.71	7.23	5.07	4.99	4.42
9	10.96	5.41	8.91	5.54	6.23	5.10	7.03	5.32
10	8.68	5.16	10.14	5.93	7.17	5.57	6.33	4.61
Diff. summary	$\overline{\Delta p_{2^{60}}}$ 9.13	$\sigma_{\Delta p_{2^{60}}}$ 5.58	$\overline{\Delta p_{2^{70}}}$ 9.85	$\sigma_{\Delta p_{2^{70}}}$ 5.85	$\overline{\Delta p_{2^{80}}}$ 6.51	$\sigma_{\Delta p_{2^{80}}}$ 4.87	$\overline{\Delta p_{2^{100}}}$ 7.62	$\sigma_{\Delta p_{2^{100}}}$ 4.98

Table 13: Validation of Gamma model in predicting the solution quality for CHC.

problem size does not suppose increasing the difference between the models and the empirical results. Therefore, the Gamma model can provide a suitable sketch for the behaviour of CHC and PBIL when they are used to solve the Root Identification Problem.

7 Conclusions and Future works

In the present work, the behaviour of evolutionary algorithms when they are used to solve the Root Identification Problem has been studied. The Root Identification Problem has been defined in terms of a combinatorial optimization problem and CHC and PBIL algorithms have been used in a statistically optimal way to solve it.

First, the expressions to obtain a parameter setting for each algorithm given

Δx_T	Problem size							
PBIL	2^{60}		2^{70}		2^{80}		2^{100}	
Instance	$\overline{\Delta x_{2^{60},I}}$	$\sigma_{\Delta x_{2^{60},I}}$	$\overline{\Delta x_{2^{70},I}}$	$\sigma_{\Delta x_{2^{70},I}}$	$\overline{\Delta x_{2^{80},I}}$	$\sigma_{\Delta x_{2^{80},I}}$	$\overline{\Delta x_{2^{100},I}}$	$\sigma_{\Delta x_{2^{100},I}}$
1	6.67	5.50	8.32	4.99	7.83	4.71	7.86	4.03
2	11.75	4.41	7.88	5.02	11.50	3.83	8.23	4.98
3	7.50	4.88	6.98	5.09	9.45	6.07	9.26	4.45
4	9.30	5.44	7.33	4.81	10.27	4.34	8.11	4.32
5	6.42	5.22	7.99	5.33	9.00	5.63	8.88	4.11
6	7.91	6.30	8.02	4.42	12.23	5.83	7.56	3.99
7	8.50	6.06	8.33	4.36	6.42	4.33	8.12	5.03
8	7.71	5.38	9.09	5.76	10.00	5.18	9.01	5.11
9	5.90	4.59	8.17	5.41	8.93	5.38	8.56	4.87
10	10.56	6.02	8.33	4.09	7.00	3.79	7.07	4.56
Diff. sumary	$\overline{\Delta x_{2^{60}}}$ 8.22	$\sigma_{\Delta x_{2^{60}}}$ 5.38	$\overline{\Delta x_{2^{70}}}$ 8.04	$\sigma_{\Delta x_{2^{70}}}$ 4.93	$\overline{\Delta x_{2^{80}}}$ 9.26	$\sigma_{\Delta x_{2^{80}}}$ 4.91	$\overline{\Delta x_{2^{100}}}$ 8.27	$\sigma_{\Delta x_{2^{100}}}$ 4.55

Table 14: Validation of Gamma model in predicting the run-length for PBIL.

Δp_T	**Problem size**							
PBIL	2^{60}		2^{70}		2^{80}		2^{100}	
Instance	$\overline{\Delta p_{2^{60},I}}$	$\sigma_{\Delta p_{2^{60},I}}$	$\overline{\Delta p_{2^{70},I}}$	$\sigma_{\Delta p_{2^{70},I}}$	$\overline{\Delta p_{2^{80},I}}$	$\sigma_{\Delta p_{2^{80},I}}$	$\overline{\Delta p_{2^{100},I}}$	$\sigma_{\Delta p_{2^{100},I}}$
1	9.60	5.48	9.33	5.33	8.26	5.25	8.55	5.01
2	10.45	4.96	8.07	4.87	9.40	5.47	8.33	5.23
3	8.48	5.55	8.89	4.92	8.08	4.80	8.31	5.41
4	8.65	5.45	8.42	5.23	8.47	5.33	7.81	5.12
5	8.41	6.34	8.13	5.11	6.60	3.90	7.69	4.65
6	7.18	4.78	9.23	4.78	9.25	4.82	7.42	4.71
7	11.10	5.41	8.16	4.64	8.46	7.08	8.23	4.82
8	7.54	5.72	7.98	4.81	8.57	4.25	7.36	4.76
9	9.38	5.68	6.05	4.51	6.39	4.94	6.42	4.51
10	9.10	5.69	7.01	5.07	7.41	5.43	6.81	4.43
Diff. summary	$\overline{\Delta p_{2^{60}}}$ 8.99	$\sigma_{\Delta p_{2^{60}}}$ 5.51	$\overline{\Delta p_{2^{70}}}$ 8.13	$\sigma_{\Delta p_{2^{70}}}$ 4.93	$\overline{\Delta p_{2^{80}}}$ 8.09	$\sigma_{\Delta p_{2^{80}}}$ 5.13	$\overline{\Delta p_{2^{100}}}$ 7.69	$\sigma_{\Delta p_{2^{100}}}$ 4.87

Table 15: Validation of Gamma model in predicting the solution quality for PBIL.

Difference between Gamma model and algorithm empirical run								
	Run-length: Δx_T							
Algorithm	$\overline{\Delta x_{2^{60}}}$	$\sigma_{\Delta x_{2^{60}}}$	$\overline{\Delta x_{2^{70}}}$	$\sigma_{\Delta x_{2^{70}}}$	$\overline{\Delta x_{2^{80}}}$	$\sigma_{\Delta x_{2^{80}}}$	$\overline{\Delta x_{2^{100}}}$	$\sigma_{\Delta x_{2^{100}}}$
CHC	9.82	5.73	9.37	5.78	8.49	5.52	7.88	4.70
PBIL	8.22	5.38	8.04	4.93	9.26	4.91	8.27	4.55
	Solution quality: Δp_T							
Algorithm	$\overline{\Delta p_{2^{60}}}$	$\sigma_{\Delta p_{2^{60}}}$	$\overline{\Delta p_{2^{70}}}$	$\sigma_{\Delta p_{2^{70}}}$	$\overline{\Delta p_{2^{80}}}$	$\sigma_{\Delta p_{2^{80}}}$	$\overline{\Delta p_{2^{100}}}$	$\sigma_{\Delta p_{2^{100}}}$
CHC	9.13	5.58	9.85	5.85	6.51	4.87	7.62	4.98
PBIL	8.99	5.51	8.13	4.93	8.09	5.13	7.69	4.87

Table 16: Validation summary for CHC and PBIL.

a problem size have been defined by the simple linear regression method. The sample has been provided by a previous study where ANOVA was used to find the best parameter values.

Second, a platform-independent performance model (Run-Length Distribution) for evolutionary algorithms has been defined. Such a model can be fit to theoretical statistical distributions in order to analyze, improve and compare in a better way the algorithms. CHC and PBIL behaviours have been analyzed in a benchmark constituted by problem sizes up to 2^{50}. Among a set of distributions, the Gamma model has been considered to have the best goodness-of-fit to the empirical Run-Length Distributions.

The Gamma model has been proposed as a candidate to model CHC and PBIL performance. A set of expressions have been defined to estimate a Gamma model for each problem size and algorithm. The reliability of the predicted Gamma models have been demonstrated by validating CHC and PBIL algorithms with problem sizes with high computational requirements (search spaces of up to 2^{100} solutions).

Future works are mainly oriented to test if the behaviour of other evolutionary algorithms and metaheuristics can be modelled by the Gamma model to solve the Root Identification Problem. This would allow us to determine a general peformance model for our problem.

8 Acknowledgements

This research has been supported by FEDER and by CICYT of the Spanish Ministerio de Educación y Ciencia under the project TIN2007-67474-C03-01.

References

[1] E. H. L. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization.* Princeton University Press, Princeton, NJ, 2003.

[2] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover, New York, ninth dover printing, tenth gpo printing edition, 1964.

[3] R. K. Ahuja and J. B. Orlin. Use of representative operation counts in computational testing of algorithms. *INFORMS Journal on Computing*, 8(3):318–330, 1996.

[4] J. Aldrich. R. A. Fisher and the making of maximum likelihood 1912-1922. *Statistical Science*, 12(3):162–176, August 1997.

[5] M. M. Ali. Durbin-watson and generalized durbin-watson tests for autocorrelations and randomness. *Journal of Business and Economic Statistics*, 5(2):195–203, 1987.

[6] 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.

[7] 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.

[8] M. Birattari, L. Paquete, T. Stützle, and K. Varrentrapp. Classification of metaheuristics and design of experiments for the analysis of components. Technical Report AIDA-01-05, Darmstadt University of Technology, November 2001.

[9] L. Booker, D. Fogel, D. Whitley, and P. Angeline. Recombination. In T. Bäck, D. 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.

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

[11] G. C. Canavos. *Applied probability and statistical methods.* Addison-Wesley, 1984.

[12] I. M. Chakravarti, R. G. Laha, and J. Roy. *Handbook of Methods of Applied Statistics: Techniques of Computation, Descriptive Methods and Statistical Inference*, volume I. John Wiley and Sons, New York, 1967.

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

[14] W. J. Cody. An overview of software development for special functions. *Lecture Notes in Mathematics*, 506, 1976.

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

[16] H. Cundy and A. Rollet. *Mathematical Models.* Oxford University Press, 1961. Second edition.

[17] J. L. Devore. *Probability and Statistics for Engineering and the Sciences.* Duxburg and Brooks Cole, Pacific Grove, CA, 6th edition, 2004.

[18] A. R. DiDonato and J. Alfred H Morris. Computation of the incomplete gamma function ratios and their inverse. *ACM Trans. Math. Softw.*, 12(4):377–393, 1986.

[19] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, 2004.

[20] N. R. Draper and H. Smith. *Applied Regression Analysis.* John Wiley and Sons, Inc., New York, NY, 3rd edition, 1998.

[21] A. Eiben and Z. 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.

[22] L. J. Eshelman. The CHC adaptative search algorithm: How to safe search when engaging in nontraditional genetic recombination. *Foundations of Genetic Algorithms I*, pages 265–283, 1991.

[23] C. Essert-Villard, P. Schreck, and J.-F. Dufourd. Skecth-based pruning of a solution space within a formal geometric constraint solver. *Artificial Intelligence*, 124:139–159, 2000.

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

[25] F. Glover and M. Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England, 1993. Blackwell Scientific Publishing.

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

[27] J. Groβ. Linear regression. *Lecture Notes in Statistics*, 175(12), 2003.

[28] T. Hill and P. Lewicki. *Statistics: Methods and Applications.* Statsoft, Inc., Tulsa, OK, 2006.

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

[30] J. H. Holland. *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, 1975.

[31] 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.

[32] R. Joan-Arinyo, M. V. Luzón, and A. Soto-Riera. Constructive geometric constraint solving: a new application of genetic algorithms. In *Parallel Problem Solving from Nature-PPSN VII*, volume 1, pages 759–768, 2002.

[33] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Parameter tuning for PBIL algorithm in geometric constraint solving systems. In *World Congress in Computer Science, Computer Engineering and Applied Computing. International Conference on Genetics and Evolutionary Methods*, pages 69–75, 2008.

[34] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Ajuste, optimización y representación del rendimiento de PBIL y CHC aplicados al problema de la selección de la solución deseada. Technical Report LSI-07-41-R, Departament de Llenguatges i sistemes informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2007.

[35] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Comparación basada en distribuciones de longitud de tiempo de ejecución para PBIL y CHC aplicados al problema de la selección de la solución deseada. Technical Report LSI-07-37-R, Departament de Llenguatges i sistemes informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2007.

[36] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Parameter tuning for PBIL and CHC algorithms to solve the root identification problem in geometric constraint solving. Technical Report LSI-07-03-R, Departament de Llenguatges i sistemes informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2007.

[37] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Predicción del rendimiento de CHC y PBIL aplicados al problema de la selección de la solución deseada. Technical Report LSI-07-42-R, Departament de Llenguatges i sistemes informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2007.

[38] R. Joan-Arinyo, M. V. Luzón, and E. Yeguas. Estimación y validación de distribuciones de longitud de ejecución para chc y pbil aplicados al problema de la selección de la solución deseada. Technical Report LSI-08-27-R, Departament de Llenguatges i sistemes informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2008.

[39] 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.

[40] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[41] K.-R. Koch. *Parameter Estimation and Hypothesis Testing in Linear Models.* Springer Verlag, Berlin; Heidelberg; New York, 1999.

[42] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation.* Springer, 2002.

[43] F. G. Lobo, C. F. Lima, and Z. Michalewicz, editors. *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence.* Springer, 2007.

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

[45] M. V. Luzón. *Resolución de Restricciones Geométricas. Selección de la Solución Deseada.* PhD thesis, Dpto. de Informática, Universidade de Vigo, Ourense, Spain, September 2001.

[46] 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.

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

[48] N. Mladenović and P. Hansen. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.

[49] I. H. Osman and G. Laporte. Metaheuristics: A bibliography. *Annals of Operations Research*, 63(5):511–623, October 1996.

[50] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, New York, NY, 3rd edition, 2007.

[51] H. Ramalhino, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. K. eds., editors, *Handbook of Metaheuristics*, pages 321–353, 2002.

[52] E. A. Robinson. *Least Squares Regression Analysis in Terms of Linear Algebra.* Goose Pond Press, Houston, TX, 1981.

[53] U. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5:96–101, 1994.

[54] B. Sareni and L. Krähenbühl. Fitness sharing and niching methods revisited. *IEEE Transaction on Evolutionary Computation*, 2(3):97–106, 1998.

[55] T. Schmelzer and L. N. Trefethen. Computing the gamma function using contour integrals and rational approximations. *SIAM Journal on Numerical Analysis*, 45(2):558–571, 2007.

[56] S. S. Shapiro and C. W. Brain. A review of distributional testing procedures and development of a censored sample distributional test. In C. Taillie, G. P. Patil, and B. A. Baldessari, editors, *Statistical distributions in scientific work*, volume 5, pages 1–24, Dordrecht, Holland, 1981. D. Reidel Publishing Company.

[57] G. W. Snedecor and W. G. Cochran. *Statistical Methods.* Iowa State University Press, 8th edition, 1989.

[58] M. A. Stephens. EDF statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association*, 69:730–737, 1974.

[59] J. W. Tukey. *Exploratory Data Analysis.* Addison-Wesley, Reading, MA, 1977.

[60] S. Vila. *Contribution to Geometric Constraint Solving in Cooperative Engineering.* PhD thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, September 2003.

[61] S. Weisberg. *Applied Linear Regression.* John Wiley and Sons, Inc., New York, NY, 3rd edition, March 2005.

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

[63] E. Yeguas. Root identification problem in geometric constraint solving. September 2008. `http://www.uco.es/~in1yeboe/benchmark.html`.

[64] E. Yeguas, M. Luzón, E. Barreiro, and R. Joan-Arinyo. Estudio e implementación de metaheurísticas para solucionar el problema de la selección de la solución deseada. Technical Report LSI-08-25-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 2008. Written in Spanish.

[65] E. Yeguas, M. Luzón, E. Barreiro, and R. Joan-Arinyo. Experimentos y análisis de resultados tras la aplicación de metaheurísticas al problema de la selección de la solución deseada. Technical Report LSI-08-26-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 2008. Written in Spanish.