Soft Comput
DOI 10.1007/s00500-011-0745-9

ORIGINAL PAPER

Modeling dynamics of a real-coded CHC algorithm in terms of dynamical probability distributions

Jesús Marín · Daniel Molina · Francisco Herrera

Abstract Some theoretical models have been proposed in the literature to predict dynamics of real-coded evolutionary algorithms. These models are often applied to study very simplified algorithms, simple real-coded functions or sometimes these make difficult to obtain quantitative measures related to algorithm performance. This paper, trying to reduce these simplifications to obtain a more useful model, proposes a model that describes the behavior of a slightly simplified version of the popular real-coded CHC in multi-peaked landscape functions. Our approach is based on predicting the shape of the search pattern by modeling the dynamics of clusters, which are formed by individuals of the population. This is performed in terms of dynamical probability distributions as a basis to estimate its averaged behavior. Within reasonable time, numerical experiments show that is possible to achieve accurate quantitative predictions in functions of up to 5D about performance measures such as average fitness, the best fitness reached or number of fitness function evaluations.

Keywords CHC algorithm · Probability distribution · Real-coded optimization · Algorithm dynamics

1 Introduction

The original CHC algorithm (Cross-generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation) was proposed by Eshelman (1990) and extended to deal with real-coded chromosomes (Eshelman and Schaffer 1992). Its basis is to combine a selection strategy with a very high selective pressure and several mechanisms inducing a strong diversity (crossover and incest prevention).

CHC algorithm is an Evolutionary Algorithm (EA) that has reported very good results (e.g. Cano et al. 2003; Luzón et al. 2004; Cordón et al. 2006; Huang and Wang 2007; Nebro et al. 2007), and it also is used as a representative EA in empirical studies of an algorithmÕs performance (Cano et al. 2005; Whitley et al. 2006; Herrera et al. 2010). In particular, a specific version for real-coded optimization, real CHC, was proposed by its authors using the crossover operator BLX-α (Eshelman et al. 1993), and it has been successfully used in real-life applications (e.g. Delgado et al. 2006; Kesur 2009; Santamaría et al. 2009). However, there are no theoretical studies to describe CHC behavior or to predict its performance when applied to optimize real-coded functions. A theoretical model could help us to mathematically understand how the algorithm works on a specific function.

Many authors consider the EA and the fitness function as a system obeying a specific dynamics. Essentially, the optimization process of an EA can be described by a difference equation, $p.e.\ u(t+1) = sel(u(t)) \circ rep(u(t))$, where $u(t)$ is the population at generation t, $sel(\cdot)$ is a survival

J. Marín (&)
Department of Automatic Control (ESAII),
Universitat Politècnica de Catalunya, EUETIB,
Urgell 187, 08036 Barcelona, Spain
e-mail: jesus.marin-sanchez@upc.edu

D. Molina
Department of Computer Science and Engineering,
University of Cádiz, 1103 Cádiz, Spain
e-mail: daniel.molina@uca.es

F. Herrera
Department of Computer Science and Artificial Intelligence,
University of Granada, 18071 Granada, Spain
e-mail: herrera@decsai.ugr.es

selection operator and replacement is a reproduction method as recombination and/or mutation. Under this approach, the goal of a mathematical model (Eiben and Rudolph 1999; Beyer et al. 2002) is to predict the algorithm dynamics or alternatively, certain aspects of its behavior.

A global understanding of the behavior of EAs can be achieved using a detailed description of all population states and applying MarkovÕs chain to analyze its dynamics (e.g. Holland 1992; Stephens and Waelbroeck 1999; Vose 1998). However, this treatment makes it difficult to obtain quantitative measures related to the optimization performance. Instead, some models are based on approaches to modeling nonlinear systems where the average behavior is described by some macroscopic variables such as expected average population fitness and the expected best fitness. Therefore, as far from equilibrium systems, dynamics of EA operators can be described in terms of fitness distributions (Chellapilla and Fogel 1999) that tend to a stationary state, often under infinite population assumptions (Qi and Palmieri 1994a, b). Although the majority of theoretical works are mainly focused on combinatorial problems (e.g. Bornholdt 1998; van Nimwegen et al. 1999; Prügel-Bennett and Rogers 2001; Schmitt 2004), some studies include simple real-coded EAs or some particular operator (e.g. Nomura 1997; Poli et al. 2007; Witt 2008).

Unfortunately, in the majority of these theoretical works on real optimization, to keep them computationally tractable, there are often considered or very simple real-coded functions such as the sphere model, the ridge functions (Beyer et al. 2002), landscapes with one or two peaks (Wright et al. 2002) or functions of up 2D (Poli et al. 2007). Even more, they usually tackle very simplified algorithms that only carry several characteristics of EA applied to real-world problems (Beyer and Schwefel 2002). Hence, it is necessary to bring the mathematical models to real models of evolutionary algorithms in order to reduce as much as possible oversimplifications.

This paper, trying to reduce the simplification, proposes a model of a slightly simplified version of the popular real-coded CHC. Also, we seek to extend the focus on harder real-coded functions such as landscape functions of more than two peaks for several dimensions.

Like other theoretical models do, the proposed model describes the dynamics of the algorithm by using probability distributions on a search space dynamically discretized. Although slightly simplified, our model considers important components of an EA such as selection and crossover operators, to capture the characteristics of real CHC. We believe of interest to analyze whether such a model is computationally tractable to achieve accurate quantitative predictions about some performance measures such as average fitness, the best fitness reached or number of fitness function evaluations.

Our motivation is based in observations about algorithmÕs behavior. In stationary stage, we observe that individuals in population tend to be distributed as clusters around the peaks of fitness function. By an analysis of possible cases, we describe dynamics of these clusters when real operators are applied. This is performed in terms of dynamical probability distributions as a basis to estimate its averaged behavior under infinite population assumption. These probability distributions are implemented by matrices which dynamically discretize search space. Several computational experiments compare the accuracy of the obtained predictions with the obtained by real-coded CHC algorithm. These results show the ability of the model to capture the full dynamics of the shape of the search pattern, even when the function landscape has an irregular appearance. We also show the feasibility of using small matrices to make good predictions in functions of up to 5D.

The paper is structured as follows. Firstly, in Sect. 2, we describe a real-coded CHC algorithm. In Sect. 3, we present a general approach to describe its dynamics by using probability distributions. In Sect. 4, we propose a model for elitist replacement and selection. In Sect. 5, we detail how crossover operator and incest prevention mechanism are modeled. In Sect. 6, we propose a formulation for estimating average performance measures. In Sect. 7, we discuss some aspects about model implementation. In Sect. 8, we describe our cases of study. In Sect. 9, we present numerical results that show the feasibility of our approach. Finally, Sect. 10 presents the concluding remarks on the present study. Additionally, Appendix includes a formulation for hypersphere volumes used in this work.

2 Real-coded CHC algorithm

The CHC algorithm works with a population of individuals or set of solutions. The population size (denoted by N) is a constant, usually small. This population (the parent) is updated at each time step or generation by applying the following operators:

Crossover operator. At each generation, a new offspring is produced by choosing random pairs of individuals from the parent population and recombining them. CHC produces new individuals applying a real-parameter crossover operator as BLX-α (Eshelman and Schaffer 1992; Sánchez et al. 2008) or PBX-α (Lozano et al. 2004).

Incest prevention mechanism. Recombination of mating pairs is only performed if the pair of individuals is not very similar, i.e. if the distance between them is higher than a parameter d, called inhibitory distance of crossover or incest threshold.

Elitist replacement and selection. The next parent population is created according to an elitist criterion, based on selecting the best individuals in the old parent population and their offspring.

The selection elitist criterion makes the population becomes more homogeneous. To delay this situation, CHC applies the incest prevention mechanism. But, along time, the population becomes homogeneous and no new offspring are generated. As a consequence, the incest threshold has to be progressively decreased. Incest prevention mechanism can be adaptive if the incest threshold is automatically reduced.

Often, individuals are encoded using binary reßected Gray coding, the distance measure used is the number of bits (mating threshold or Hamming distance) that differ these pair of binary-coded individuals. Sometimes, Euclidean distance is considered, and the adaptive mechanism makes parameter d automatically decreased from an initial value d_{ini} by multiplying a constant (close to 0.9).

When the incest threshold gets to 0, it is assumed that the population has converged and the algorithm is stalled. A mechanism of restart is then used to generate new diversity in the population. When restarting, the best individual remain unchanged, and the rest are randomly generated (cataclysm). In this paper, cataclysm is not considered.

2.1 BLX-a crossover operator

BLX-a crossover operator (Eshelman and Schaffer 1993) has been successfully used in benchmark studies (Cervone et al. 2000; Zhao et al. 2007) and in real-life applications such as traffic signals (Kesur 2009), image processing (Cordón et al. 2006; Santamaría et al. 2009) or financial predictions (Delgado et al. 2006). In the following, we briefly describe the BLX-a crossover operator.

Let x and y be two real-coded individuals to be crossed, defined as $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$, where n is the function dimensionality. This operator generates an individual $z = (z_1, \ldots, z_n)$ where each z_i is randomly (uniformly) generated within the interval

$$[\min(x_i, y_i) - \alpha I, \max(x_i, y_i) + \alpha I]$$

where $I = |y_i - x_i|$. This equation can be reformulated in terms of random variables $\gamma_i \in [0, 1]$ as follows:

$$z_i = x_i + (y_i - x_i) + \gamma_i \cdot (2\alpha - 1), \quad \forall i = 1, \ldots, n \qquad (1)$$

Instead of generating different values for each γ_i, we simplify BLX-a using the same value γ for all dimensions. This simplified version called sBLX-a can be defined as a linear combination of vectors, such as

$$z = x + (y - x) \cdot \gamma + \alpha \cdot (2\gamma - 1) \qquad (2)$$

3 Theoretical model for CHC behavior

Initially, the algorithm starts with a uniform distribution of the individual over the search space $\Omega \subset \mathbb{R}^n$, where n is the dimensionality of function or also called scale problem. When we assume the function to maximize is a landscape and the location of each peak is known, we observe the next behavior of the algorithm:

1. In stationary stage, individuals in population tend to be distributed around the peaks of fitness function, that we consider as clusters.
2. Crossover is mainly applied to individuals of different clusters. Due to prevention of incest mechanism, individuals belonging to the same cluster cannot be crossed between them.
3. Due to its adaptive mechanism, parameter d is automatically reduced. This reduction increases the average fitness of population. As a consequence, the size and number of clusters is reduced.

Our approach to model CHC are based in these observations. In essence, the proposed model makes estimations about the behavior of these clusters and their contributions to the current generation of individuals. As a result, we estimate measures on the average performance each generationt such as expected average population fitness (denoted by $f_{best}(t)$), the best fitness $\bar{F}(t)$ or number of offspring $\bar{N}_{offs}(t)$.

Although the search space $\Omega \subset \mathbb{R}^n$ is continuous, for practical purpose is discretized (Poli et al. 2007) and denoted as Ω_k. This discretized search space is implemented by using hypermatrices of size Ω_k, $k = s^n$, where s is the number of intervals of discretization at each dimension. This notation is used below for analyzing the computational complexity of each operation performed according to the size of hypermatrices.

Hence, the proposed model works with probability distributions (in statistics, probability mass functions or PMF) instead of probability density functions. These dynamical probability distributions denoted by $\mu^{(t)}(x)$ identifies, at each iteration t; the percentage of population individuals located at $x \in \Omega_k$; or rather, within a particular interval of discretization of Ω_k. A PMF or probability distribution can be understand as the composition of an infinite-sized population limit.

When we compare the results obtained by PMF with obtained by an actual finite-sized population, we observe differences. Because population size is a discrete value, probability distributions used by models cannot reßect the effect of genetic drift that suffer populational algorithms. However, these differences are reduced when the population

size is increased. Thus, infinite population size is a reasonable approximation to the finite population case usually made by many theoretical models (Bartlett 1998) that make it easier to deal analytically.

PMF is implemented by using hypermatrices that discretize this search space. Although discretization introduces errors in approximations, we observe experimentally that is possible to achieve accurate quantitative prediction using hypermatrices of relatively small size. Moreover, discretization is often introduced in CHC algorithm to measure distances such as Hamming distance.

In the beginning, all individuals of population are uniformly distributed over Ω : We denote this PMF by $u^{(1)}_{par}(x)$ distribution. At each generation t, the parent PMF $u^{(t)}_{par}(x)$ is recurrently updated by modeling the algorithm's operator. Therefore, it satisfies:

$$\forall t \in \{1, G\} : \left(\sum_{x \in \Omega} u^{(t)}_{par}(x) \right) = 1 \qquad (3)$$

where G is the number of generations or run length.

To model crossover operator, we estimate the offspring PMF denoted by u_{offs} using intermediate probability distributions according to each cluster k_i as well as to these individuals do not belong to any cluster (denoted by $\overline{K}$). Each pair of individuals generates an offspring when they are crossed. This means that the number of offspring is at most half the size of the population. Formally,

$$\forall t \in \{1, G\} : \left(\sum_{x \in \Omega} u^{(t)}_{offs}(x) \right) \leq 0.5 \qquad (4)$$

To simplify the notation, in the following, we will use the terms $u_{...}$ or $u_{...}(x)$ instead of $u^{(t)}_{...}(x)$.

Selection and replacement require calculating the distribution of fitness values of the population individuals by using the hypermatrix u_{par}. To perform this calculation, we define $F(x)$ as the distribution of fitness values over the discretized search space Ω : This hypermatrix of fitness values is used to determine which are the regions within search space with lower fitness, where decreases the presence of individuals into the population. In this way, selection does reduce its probability in the distribution u_{par}.

Shortly, the procedure of model is summarized in Fig. 1 where the steps for modeling crossover, replacement and performance are detailed in the following sections.

```
Uniform initialization φ^(1)_par(x) = 1/#φ, ∀x ∈ Ω*
      where #φ is the size of hypermatrix φ
For each generation t = 1 to G
     φ^(t)_offs := Crossing operator model(φ^(t)_par)
          For each cluster i = 1 to n_c
               Crosses the cluster k_i with the others
                    individuals whose offspring
                    fills φ_⟨offs,k_i⟩
          Crosses individuals outside the clusters
               whose offspring fills φ_⟨offs,K̄⟩(x)
          Fills φ^(t)_offs by adding φ_⟨offs,k_i⟩ and φ_⟨offs,K̄⟩(x)
               according to their contributions to
               overall population
     φ^(t+1)_par := Model for elitist replacement(φ^(t)_par, φ^(t)_offs)
     [N^(t)_offs, F̄^(t), f^(t)_best] := Performance model(F*, φ^(t)_offs, φ^(t+1)_par)
```

Fig. 1 Procedure for modeling CHC algorithm

4 Modeling elitist replacement and selection

At each generation, a new parent population incorporates the offspring generated:

$$u^{(t+1)}_{par}(x) = u^{(t)}_{offs}(x) + u^{(t)}_{par}(x) \qquad (5)$$

In order to fulfill Eq.3 and thereby hold the population size constant, we need to initialize some positions of $u_{par}(x)$ with low fitness to model the selection of individuals for the next generation.

For this purpose, this probability distribution is sorted according to its fitness, and the best are selected by determining a cutting fitness f_k: The procedure to determinate this value requires the hypermatrix $F(x)$. Each fitness value is paired with its probability as follows:

$$FP = \{\langle f_i, p_i \rangle | f_i = F(x) \text{ and } p_i = u^{(t)}_{par}(x), \forall x \in \Omega\} \qquad (6)$$

and the FP is sorted by fitness satisfying that:

$$\forall i, j \in \{1, \#FP : f_i \leq f_j \Leftrightarrow i \leq j\} \qquad (7)$$

where $\langle f_i, p_i \rangle = FP[i]$, $\langle f_j, p_j \rangle = FP[j]$ and $\#FP = s^n$ is the size of FP.

From the ordered set FP, we search the position k that meets that the cumulated sum of probabilities of previous positions is closer to 1, i.e. the cumulated percentage of individuals completes the entire population. Formally, k satisfies

$$k = \max\left\{ j \left| \sum_{i=1}^{j} p_i \leq 1 \right. \right\} \qquad (8)$$

where $\langle f_i, p_i \rangle = FP[i]$:

Finally, we use f_k to initialized to 0 these positions of $u_{par}(x)$ with lower fitness values:

$$u^{(t+1)}_{par}(x) = \begin{cases} u^{(t)}_{par}(x), & \text{if } F(x) \geq f_k \\ 0, & \text{otherwise} \end{cases} \qquad (9)$$

where $\langle f_k, p_k \rangle = FP[k]$:

Because of sorting process, the computational complexity of this operation is $O(n \log(s^n))$ where s^n is the size of n-dimensional hypermatrices.

5 Modeling crossover operator and incest prevention mechanism

We assume the fitness function is a landscape with k_c peaks within one local optimum. Each peak of the function is treated independently as follows. For each peak we consider a cluster k_i of radius $d/2$, where d is the inhibitory distance of crossover. A cluster k_i is defined by a hypersphere centered at the position of the local optimum x_i

$$k_i = \{x \in \Omega \mid \|x - x_i\| \leq d/2\} \quad (10)$$

where $\|\cdot\|$ is the euclidean distance. In the practice, the size of clusters can be smaller whether individuals around the optima have fitness lower than the worst one of the population.

From parent PMF, we define probability distribution of a cluster k_i as:

$$u_{h_{par,k_i}}(x) = \begin{cases} u_{par}(x), & \text{if } x \in k_i \\ 0, & \text{otherwise} \end{cases} \quad (11)$$

Each cluster is crossed with all locations of parent PMF that do not belong to this cluster. As a result of this mating process of probability distributions, a PMF of offspring denoted by $u_{h_{offs;k_i}}$ is generated for each cluster.

In a first approximation, a PMF of all offspring is calculated by adding the offspring of each cluster, weighted by the probability of each cluster. This weight is the probability of selecting an individual inside this cluster as the first individual for the crossing, and it is defined as:

$$p_{k_i} = \sum_{x \in k_i} u_{h_{par,k_i}}(x) \quad (12)$$

and probability distribution of all offspring is estimated adding the contribution of each cluster as follows:

$$u_{offs}(x) = \sum_{i=1}^{n_c} (p_{k_i} u_{h_{offs;k_i}}(x)) \quad (13)$$

We observe this simplified model of coarse-grained can be refined by filling down some shortcomings. Each case is discussed below and provides a correction factor to the contribution in the offspring of each peak. By considering these cases, the probability distribution for the offspring becomes:

$$u_{offs}(x) = c_1 c_2 c_3 \sum_{i=1}^{n_c} (p_{k_i} u_{h_{offs;k_i}}(x)) + u_{h_{offs;\bar{K}_i}}(x) \quad (14)$$

where c_1, c_2 and c_3 are defined later in Eqs. 20, 21 and 22, respectively; and $u_{h_{offs;\bar{K}_i}}$ is the offspring of crossing individuals outside any cluster, explained in the following section.

5.1 Crossing each cluster with the rest

$u_{h_{offs;k_i}}$ is the offspring produced by crossing a cluster k_i with parent locations outside of the cluster. At first, the $u_{h_{offs;k_i}}$ matrix is initialized to 0. The idea is that the proportion of individuals in each x outside the cluster (denoted by $u_{par}(x)$) is linearly recombined with all positions x_a inside a cluster k_i: Each one of these crossings evenly redistributes the proportion of $u_{par}(x)$ throughout all intermediate locations between pair of recombined parents, according to the a parameter of sBLX-a operator. Each crossing is performed with a probability w_a which is proportional to $u_{par}(x_a)$. As a result of this process, the percentage of individuals in $u_{h_{offs;k_i}}$ increases for intermediate locations between each pair of combined parents.

This procedure is accurate but computationally expensive because it must cover all locations of a hypercone of base hyperspherical. And is relatively huge, specially in first stages of the optimization process, the crossing with the center of the cluster or with its periphery implies great differences in redistributions. This restriction makes that the process cannot be simplified by using a single point as representative of the whole cluster.

Instead, we propose a hybrid solution that solves this problem, achieving a less accurate approximation, but computationally cheaper. Thereby, we combine the current location x with all locations x_a from the periphery of the cluster in direction toward the center of cluster k_i. Formally, we define

$$x_a = a(x - x_i) + x_i, \quad \forall a \in [0, d] \quad (15)$$

where $d = d/2\|x - x_i\|$ is the relative size of the radius from the distance of both positions, and $d/2$ is the radius of the cluster.

Due to each cross occurs with a given probability, x is combined with individuals located at x_a with probability w_a defined as:

$$w_a = u_{par}(x_a) \Big/ \sum_{c=0}^{d} u_{par}(x_c) \quad (16)$$

Crossing process performed by BLX for x and x_a involves a uniform redistribution of $u_{par}(x)$ for intermediate locations x_b: As search space Ω is discretized, these locations are adjacent positions of hypermatrix x_{par}

generated by step Δb: By assuming sBLXa with $a = 0$; b can be fixed within interval $[a; 1]$: Therefore, these redistributions are performed along a normalized distance $1 - a$: Formally, the proportion of individuals $u_{hoffs;k_i}(x_b)$ increases as follows:

$$\Delta u_{hoffs;k_i}(x_b) = u_{par}(x)\, w_a \frac{1}{1-a}\Delta a; \quad \forall b \in [a; 1] \qquad (17)$$

where Δa is the interval for the discretization of a also used to traveling adjacent positions of hypermatrix u_{par}: Figure 2 shows schematically a graphical representation for the notation used.

One way for optimizing this process is to consider k_a in terms of x_b: Specifically, for each location x_b we consider all points x_a that combined with x can generate individuals in x_b: Under this approach, Eq. 17 can be reformulated so that x_b increases its probability depending on x as follows:

$$\Delta u_{hoffs;k_i}(x_b) = u_{par}(x) \int_{a=0}^{\min(b,d)} \frac{w_a}{1-a}\, da; \quad \forall b \in [0; 1] \qquad (18)$$

where $\min(b; d)$ is used to avoid x_a is outside cluster: in this section, x can only be crossed with positions x_a within the cluster.

This reformulation of Eq. 18 in terms of continuous space provides an interpretation for the interval of the discretization Δa; as appeared in Eq. 17. But to work on discretized spaces, both a and b use the same interval of discretization such that $\Delta a = \Delta b$ in order to traveling the same adjacent positions of hypermatrix u_{par}:

As the search space is discretized by using hypermatrices of size $k\Omega$, $k = s^n$; this size is used to define computational complexity of the whole process. In particular, the Eq. 18 is computed according to locations between x and the center of the cluster x_i; the number of locations is asymptotically limited by $O(\sqrt[n]{s})$. This process is applied to each $x \in \Omega$ to calculate $u_{hoffs;k_i}$: Thereby, $u_{hoffs;k_i}$ is computed in $O(\sqrt[n]{s^{n+1}})$.

To graphically illustrate the whole process, Fig. 3 shows the result of crossing a cluster with the remaining clusters for a simple example. In this figure, we observe an attraction effect toward the cluster used in the cross. This gives the algorithm the ability to explore positions between clusters. Attraction effect also suggests a competition between clusters controlled by the percentage of population that brings.

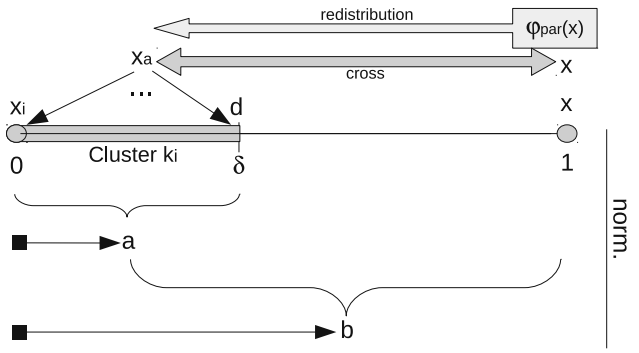

Fig. 2 Schema of notation used in the crossing of a cluster k_i with an external point x by assuming $a = 0$: Linear redistribution of $u_{hpar,k_i}(x)$ crossing x with the locations located between the center of the cluster and on its periphery

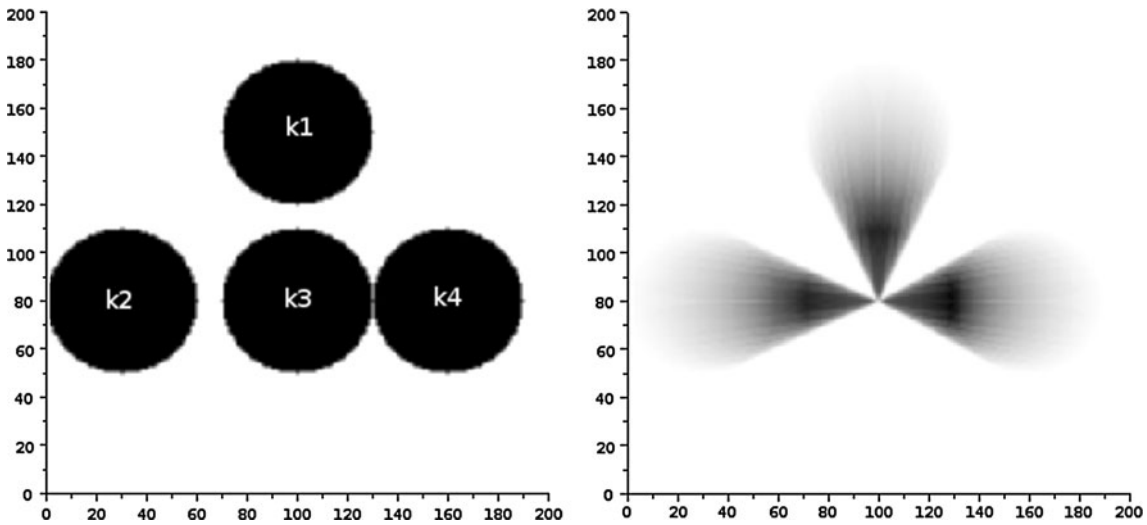

Fig. 3 Example of crossing a peak with the others in a search space 2D. Left A parent MPF $u_{par}(x)$ where all positions within the cluster are equally likely. Right probability distribution $u_{hoffs;k_3i}$ obtained by crossing the central cluster k_3 with all external points. Darker regions represent higher probability

5.2 Overlapping clusters

When there are overlapping clusters, these individuals located in overlap regions are considered more than once. We define p_K as the probability of an individual is in any cluster as:

$$p_K = \sum_{x \in K} u_{hpar,k_i}(x) \quad (19)$$

where $K = [_{i=1}^{n_c} k_i$ contains the points belonging to any cluster. Under this overlapping situation p_K may be different to $\sum p_{k_i}$: Although the importance of this gap is reduced as d decreases, we define a correction factor c_1 to Eq. 14 to remove the proportion of individuals which should not have been considered:

$$c_1 = \left(p_K \Big/ \sum_{i=1}^{n_c} p_{k_i} \right)^2 \quad (20)$$

where the proportion is squared because the overlap region has been considered both within and outside the cluster in the choice of mating pairs. c_1 is computationally determined according to number of clusters $O(n_c s^n)$.

5.3 Parents chosen from the same cluster

In previous sections, we fix one order in the choice of the pair for mating (p.e. a cluster crossed with rest). This fact requires halving the contribution of each cluster. Specifically, p_{k_i} is used to choose an individual inside a cluster assuming it is the first individual of to cross-pair. As this can be the second, this probability should be half. Moreover, it also happens that both could belong to the same cluster. For these reasons, the weight for each PMF in Eq. 13 requires be adjusted by: $0.5(p_{k_i} - p_{k_i}^2)$: Due to this improvement must be introduced after removing the effect of the overlap, we introduce other correction factor c_2 used in Eq. 14 instead of modifying the weight:

$$c_2 = 0.5 \frac{p_K}{n_c} \quad (21)$$

where n_c is the number of clusters and p_K/n_c is the average probability of choosing an individual of a cluster. c_2 is computationally determined in $O(n_c s^n)$.

5.4 Nearby clusters that interfere

In previous sections, we have already considered overlap regions between clusters, but the model allows also to cross points of different clusters which are closer than d. This situation is produced when the distance between centers of clusters is lower than $d/2$. In order to remove these crossings in a way computationally efficient, we use some kind of rough heuristic. For this, we consider a situation

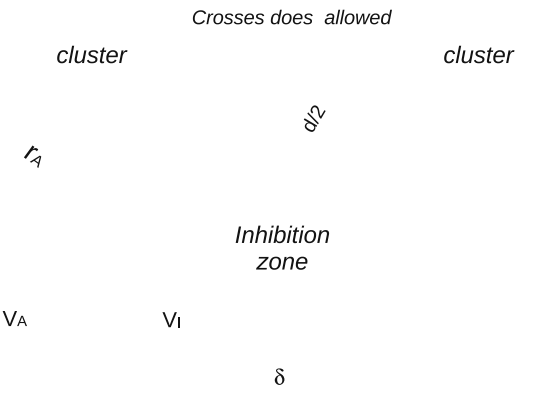

Fig. 4 Schema of intersection of average cluster of radius r_A with a hypersphere of diameter d: Here, hyperspheres are represented as circles. Intersection area represents crossings do not allowed

where two average clusters are separated by an distance δ. As shows Fig. 4, the idea is to estimate the intersection of one of these average clusters of radius r_A with a hypersphere of diameter d; whose center is located midway between the two clusters. This hypersphere represents an interference region where crossings do not allowed.

On one hand, we approximate δ by the median of distances between pairs of cluster centers. On the other hand, we estimate the average cluster by using the average volume V_A of all clusters: $V_A = \sum_{i=1}^{n_c} V_{k_i}/n_c$ where n_c is the number of clusters, and V_{k_i} is the volume estimated for each cluster k_i: V_{k_i} can be estimated considering only points belonging to cluster k_i having a probability higher than 0, i.e. that meets $x \in k_i$ and $u_{hpar,k_i}(x) > 0$: Assuming this volume is of a hypersphere, we calculate its radius using the formula for volume of a hypersphere $r_A = (V_A/C_n)^{1/n}$ (see Appendix for details. Therefore, we consider situations where a cluster can have an effective radius lower than $d/2$ because some locations have fitness values lower than the worst of population.

As defined in Appendix, the intersection between the average cluster of radius r_a and a hypersphere of diameter d located at a distance $\delta/2$ is $V_I(r_A; d/2; d/2)$. By assuming the probability of each point belonging to average cluster is uniform, we estimate the probability inside the volume of intersection as follows:

$$p_I = \frac{\sum_i p_{k_i}}{n_c} \frac{V_I}{V_A}$$

where p_{k_i} is defined in Eq. 12. Hence, we define another factor (see Eq. 14) that removing crossings between pairs of points:

$$c_3 = 1 - n_c p_I^2 \quad (22)$$

where $V_C(r_A)$ is the average probability of a cluster. c_3 is computationally determined according the number of clusters in $O(n_c s^n)$.

5.5 Crossings outside the clusters

In previous sections, we have only considered crossings in which clusters are involved. But in early stages, we expect many crossings are performed between individuals located outside the clusters. In order to estimate this impact, we may consider these points of the parent PMF which are not in any cluster:

$$u_{hpar,\overline{K}i}(x) = u_{par}(x) - \max_{k_i}\{u_{hpar,k_i}(x)\} \quad (23)$$

In order to avoid crossings between neighbor individuals, we divide this space outside the clusters in other virtual clusters shaped hypersphere of diameter d. But for a better estimation of its number, it is more convenient to consider hypercubes. Therefore, the number of virtual clusters is given by $n_{vc} = V_{\overline{K}}/d^n$ where $V_{\overline{K}}$ is the volume estimated considering points x that meets $u_{hpar,\overline{K}i}(x) > 0$.

At this point, a fine-grained formulation involves to tessellate this space and re-apply all above cases for each virtual cluster, but computational overhead does not worth this level of detail. Instead, we assume offspring follows a parental-like distribution u_{par} and we estimate the extent to which contributes to overall offspring:

$$u_{hoffs,\overline{K}i}(x) = c_4 u_{par}(x) \sum_{x' \in \Omega} u_{hpar,\overline{K}i}(x') \quad (24)$$

where the parental distribution is normalized to match the size of $u_{hpar,\overline{K}i}$; and c_4 is the proportion of individuals of the offspring generated by crossing.

This factor c_4 is then estimated following a similar reasoning to the previous cases. We define $p_{vc} = 1/n_{vc}$ as the probability of choosing a virtual cluster. This virtual cluster is crossed with the other clusters—the latter with probability $(n_{vc} - 1)/n_{vc}$. The same operation is repeated for each cluster. By this process, the offspring generated has the following proportion:

$$c_4 = 0.5\, n_{vc}(p_{vc} - p_{vc}^2) = \frac{n_{vc} - 1}{n_{vc}} \quad (25)$$

In terms of computational complexity, $u_{hoffs,\overline{K}i}$ is computed in $O(n_c s^n)$.

6 Average performance model

By the proposed model, some measures on the average performance can be estimated at each generation t. These are the measures used in our study: the number of offspring $(N_{offs}^{(t)})$, mean of the population fitness $(\overline{f}^{(t)})$ and fitness value of the best solution found $(f_{best}^{(t)})$. These measures are estimated on the basis of both u_{offs} and u_{par} PMFs.

u_{offs} is calculated by crossover model as it is described in Sect. 5. In terms of computational complexity, this hypermatrix is computed in $O(n_c \bar{n} s^{n+1})$ where n_c is the number of clusters and s^n is the hypermatrix size. According to the complexity of selection model described in Sect. 4, u_{par} is computed in $O(n \log(s) + n s^n)$.

N_{offs} measure is specially interesting to evaluate the computational effort because it is an estimator about the

Table 1 Peak features: height, width and location for each function of F_1 to F_5

Function F_1			Functions F_2 to F_5					
h_i	w_i	Peak of F_1	h_i	w_i	Peak of F_2	Peak of F_3	Peak of F_4	Peak of F_5
50	50	25 25	40	80	35 85	35 85 35	35 85 35 85	35 85 35 85 50
30	50	66 66	−55	50	75 75	75 75 75	75 75 75 75	75 75 75 75 30
95	50	30 80	75	65	25 30	25 30 30	25 30 30 25	25 30 30 25 87
70	50	75 15	99	135	45 45	45 45 45	45 45 45 45	45 45 45 45 34
100	10	45 45	85	80	80 55	80 55 55	80 55 55 80	80 55 55 80 67
"	"	"	95	40	65 55	65 55 65	65 55 65 55	65 55 65 55 72
"	"	"	−85	90	25 65	25 65 25	25 65 25 65	25 65 25 65 12
"	"	"	65	60	85 15	85 15 85	85 15 85 15	85 15 85 15 34
"	"	"	92	60	90 90	90 90 90	90 90 90 90	90 90 90 90 24
"	"	"	−35	75	70 10	70 10 10	70 10 10 70	70 10 10 70 78
"	"	"	−10	25	45 45	45 5 45	45 45 45 45	45 45 45 45 69
"	"	"	30	70	30 30	30 30 30	30 30 30 30	30 30 30 30 76
"	"	"	−45	250	15 15	15 15 15	15 15 15 15	15 15 15 15 45
"	"	"	85	70	60 35	60 35 60	60 35 60 35	60 35 60 35 12
"	"	"	−50	200	0 0	0 0 0	0 0 0 0	0 0 0 0 0
"	"	"	85	1,000	50 50	50 50 50	50 50 50 50	50 50 50 50 20

required number of Þtness function evaluations. Because CHC model implementation u_{offs} is the percent of population for each location, the sum of all locations is an estimator for the relative size of the offspring from parents. Therefore, the number of offspring N_{offs} can be calculated as

$$N_{offs}^{(t)} = \left\lfloor N \sum_{x \in \Omega} u_{offs}^{(t)}(x) \right\rfloor \quad (26)$$

where N is the population size. In terms of computational complexity, $N_{offs}^{(t)}$ is computed in $O(s^n)$.

The expected value for the population Þtness $\bar{F}$ is the sum of the probability of each location multiplied by its Þtness:

$$\bar{F}^{(t)} = \sum_{x \in \Omega} (u_{par}^{(t)}(x) F(x)) \quad (27)$$

where F is a Þtness-value distribution. According to both F and u_{par} hypermatrix sizes $\bar{F}^{(t)}$ is computed in $O(s^n)$.

Fitness value of the best solution found f_{best} can be estimated by a method similar to that previously used to determine cutting Þtness for selection operator. From the ordered set FP deÞned in Eq. we search the location k that meets that the cumulated sum of probabilities of previous locations is closer to $1/N$; being N the population size. Accordingly, it is expected at least one individual with Þtness greater or equal than f_k. Hence,

$$f_{best}^{(t)} = f_k \quad (28)$$

where $\langle f_k, p_k \rangle = FP[k]$ and

$$k = \min\left\{ j \mid \sum_{i=1}^{j} p_i \geq \frac{1}{N} \text{ and } j \in [1, \#FP] \right\} \quad (29)$$

where $\langle f_i, p_i \rangle = FP[i]$ and $\#FP$ the number of pair of FP. Because of sorting process, the computational complexity for this measure is $O(n \log(s^n) + n s^n)$.

Therefore, the computational complexity of whole model is $O(n_c \sqrt{n} s^{n/2+1} + \log(s^n) + n s^n)$ according to dimensionality of function (n), number of peaks (n_c) and quality of discretization of search space (s).

The model works with hypermatrices which discretize the search space to both PMF and Þtness function discretization. Due to hypermatrix size scales up with the dimensionality, it is need to discuss several implementation details which make feasible to apply the model in higher scale functions. Computational time and memory requirement are two key concepts in feasibility of the model.

The number of intervals of discretization (denoted s) is the same for each variable of function and each cell of

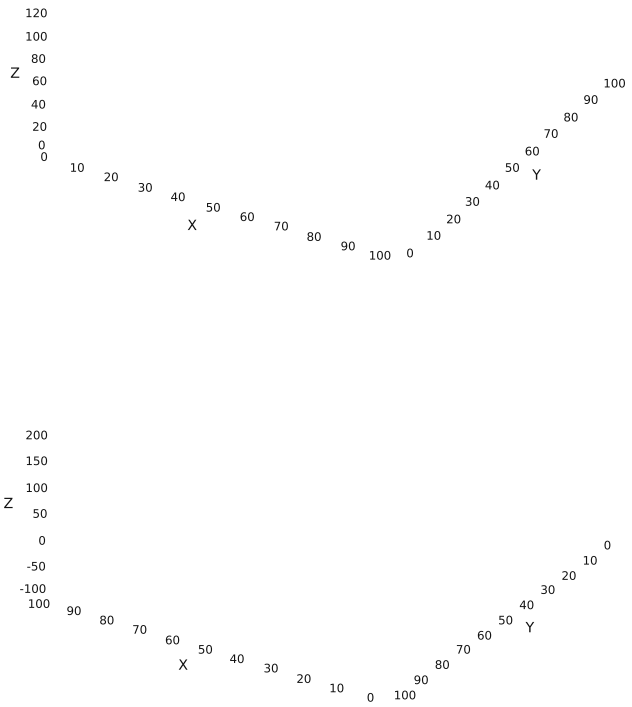

Fig. 5 Graphical representation for functions F_3 (up) and F_2 (down)

Table 2 Averaged time measures (s) required by the model (t_{model}) and one algorithm run (t_{run}) averaging 100 runs for the set of test functions

F	t_{run}	$\#F$	t_{model} by $\#u$:						
			15^n	18^n	19^n	20^n	25^n	50^n	100^n
F_1	0.187	10^6	Đ	Đ	Đ	0.252	0.297	1.06	5.59
F_2	0.176	10^6	Đ	Đ	Đ	0.404	0.555	1.99	13.6
F_3	0.177	49,836,032	Đ	Đ	Đ	4.40	8.96	116	Đ
F_4	0.174	49,787,136	Đ	Đ	83.9	103	314	Đ	Đ
F_5	0.174	45,435,424	331	857	1,168	Đ	Đ	Đ	Đ

Model results are shown according to hypermatrices sizes

the matrix store a ßoat coded in simple precision. In particular, the PMF such as parent (u_{par}) or offspring (u_{offs}) are implemented by matrices whose size is denoted by $\#u = s^n$:

An initial hypermatrix of Þtness values, denoted by F_0, is performed by uniform sampling of the search space. This matrix size is denoted by $\#F$ has been limited to $5 \cdot 10^7$ items. But this matrix F_0 is a raw matrix that is cropped and resized dynamically to the size of the probability distribution hypermatrices, according to the needs of the model as is shown below. This reduced matrix denoted by F is used by the model, as was seen. Speciþc values used in our experiments for $\#u$ and $\#F$ are summarized in Table 2.

We observe in early stages of the search that it is not needed to work with a Þne-course discretization, unlike what happens in later stages. At the same time, these regions without presence of individuals unnecessarily extend the process. Whether these hypermatrices are clipped and dynamic rescaled, we reduce considerably their sizes. The time overhead for this process is well invested because it reduces the time consumed by operating unproÞtable matrix positions.

The process of rescaling is done by following these steps:

- For each dimension, it is estimated ranges of the hypermatrix containing non-empty positions. This allows you to calculate a new effective domain for the function.
- We calculate the percentage of relative volume of new domain of the function with regard to the old. The process of rescaling is done if this percentage is less than a given constant. In our case, 60% achieves better results in the quality-time trade-off.
- We construct a new matrix u_{par} that it discretizes the new domain from old-matrix information. For each position of the new matrix, we calculate the corresponding position in the former.
- The same process is used to construct a new matrix F of same size and domain that u_{par} but using the data from F_0:

This model has been implemented using Scilab 5.3., and some low-level critical routines have been built in language C using intersci as interfacing tool. These external routines are mainly cluster crossover and rescaling routines.

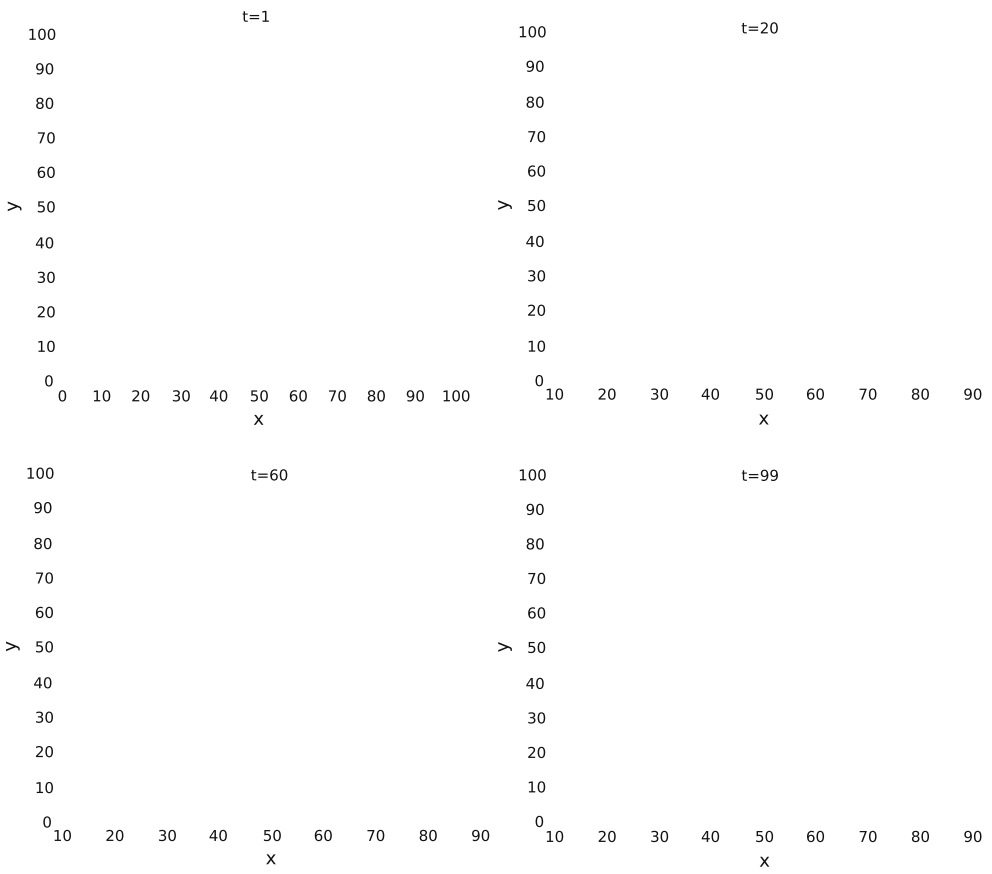

Fig. 6 Prediction for population probability distribution u_{par} and real locations of population individuals by one run of algorithm using F_1: Each picture shows the generations $t = 1; 20; 60; 99$: Darker regions represent higher probability. Individual locations are marked with stars

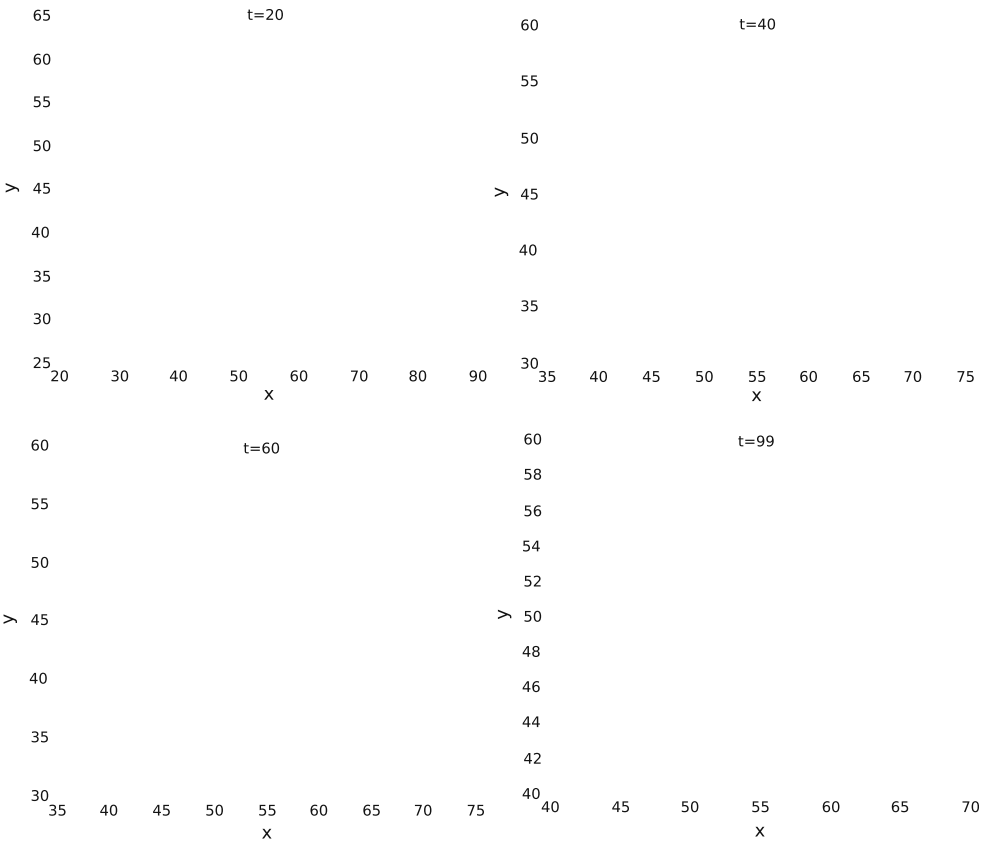

Fig. 7 Prediction for population probability distribution $u_{par}^{(t)}$ and real locations of population individuals by one run of algorithm using F_2: Each picture shows the generations $t = 20, 40, 60, 99$. Darker regions represent higher probability. Individual locations are marked with stars

Table 3 Error measures achieved by the model and standard deviation for $\overline{F}^{(t)}$ and $f_{best}^{(t)}$, averaged for all generations

Func.	Measure	Values			
F_1	#u	25^n	50^n	100^n	$\bar{r}$
	$\Delta\overline{F}$	0.09575	−0.02224	−0.03923	0.08494
	Δf_{best}	0.01460	−0.01453	−0.01011	0.06897
F_2	#u	25^n	50^n	100^n	$\bar{r}$
	$\Delta\overline{F}$	0.00098	−0.00413	−0.00113	0.01721
	Δf_{best}	−0.00232	−0.00121	−0.00077	0.00527
F_3	#u	20^n	25^n	50^n	$\bar{r}$
	$\Delta\overline{F}$	0.02275	0.01725	0.01209	0.03612
	Δf_{best}	−0.00062	0.00186	0.00364	0.02695
F_4	#u	19^n	20^n	25^n	$\bar{r}$
	$\Delta\overline{F}$	0.02924	0.03961	0.05195	0.04388
	Δf_{best}	0.00425	0.01533	0.02064	0.04310
F_5	#u	15^n	18^n	19^n	$\bar{r}$
	$\Delta\overline{F}$	0.01868	0.02179	0.00653	0.03398
	Δf_{best}	−0.03411	−0.00737	−0.02292	0.05876

Values are shown according to hypermatrices sizes

Some other efficiency improvements for its implementation have been introduced as well: (a) When it is detected peaks of cluster without individuals, with probability distribution equal to 0, these are not considered. $p(h_c)$ is calculated using a boolean hypermatrix that is used as mask to select the region of interest from a parent PMF. (c) Equations 16 and 18 are computed as a sum by traveling of the corresponding discrete locations of hypermatrices.

Table 4 Area of curves for $\overline{F}$; f_{best}; N_{offs} : real measures (mean and standard deviation using 300 runs) and model predictions are shown for each function according to hypermatrix sizes

Funct.		#u	$\overline{F}$	f_{best}	N_{offs}
F_1	Mean		0.5488	0.9135	0.2270
	r		0.0743	0.0510	0.0445
	Model	25	0.6374	0.9115	0.1122
	Model	50	0.5194	0.9062	0.1520
	Model	100	0.5024	0.9032	0.1716
F_2	Mean		0.9050	0.9937	0.0661
	r		0.0126	0.0036	0.0138
	Model	25	0.9064	0.9919	0.0492
	Model	50	0.9013	0.9927	0.0498
	Model	100	0.9043	0.9929	0.0574
F_3	Mean		0.7766	0.9542	0.0530
	r		0.0298	0.0193	0.0094
	Model	20	0.7960	0.9530	0.0616
	Model	25	0.7905	0.9549	0.0564
	Model	50	0.7854	0.9569	0.0570
F_4	Mean		0.7169	0.9109	0.0500
	r		0.0378	0.0287	0.0051
	Model	19	0.7454	0.9120	0.0802
	Model	20	0.7558	0.9294	0.0724
	Model	25	0.7682	0.9320	0.0686
F_5	Mean		0.6355	0.8192	0.0520
	r		0.0294	0.0415	0.0054
	Model	15	0.6541	0.7921	0.1368
	Model	18	0.8006	0.8469	0.0914
	Model	19	0.6419	0.7945	0.1206

All values are normalized

Table 5 p values estimated by using the Wilcoxon test to compare real measures and predictions for the area of curves $\overline{F}$; f_{best} and N_{offs}

Funct.	#u	$\overline{F}$	f_{best}	N_{offs}
F_1	25	0.1768	0.7776	0.0875
	50	0.5818	0.6787	0.1261
	100	0.4155	0.6379	0.1939
F_2	25	0.9527	0.4072	0.1764
	50	0.6319	0.5063	0.1884
	100	0.8636	0.5450	0.4645
F_3	20	0.3941	0.7227	0.2309
	25	0.5815	0.8127	0.4102
	50	0.8290	0.9468	0.3782
F_4	19	0.2997	0.8642	0.0870
	20	0.1550	0.3806	0.0910
	25	0.1088	0.3201	0.0910
F_5	15	0.4305	0.3685	0.0851
	18	0.0852	0.4370	0.0872
	19	0.8852	0.3987	0.0872

Results are shown for each function and hypermatrix size

By a reformulation, partial results can be accumulated to avoid redundant calculations. (d) Some intermediate PMF as defined in Eq. 11 are not calculated. Instead, a conditional sentence is inserted in the code.

3 Cases of study

We seek out the model behavior with different ruggedness and scale of the problem. For this purpose, we define the

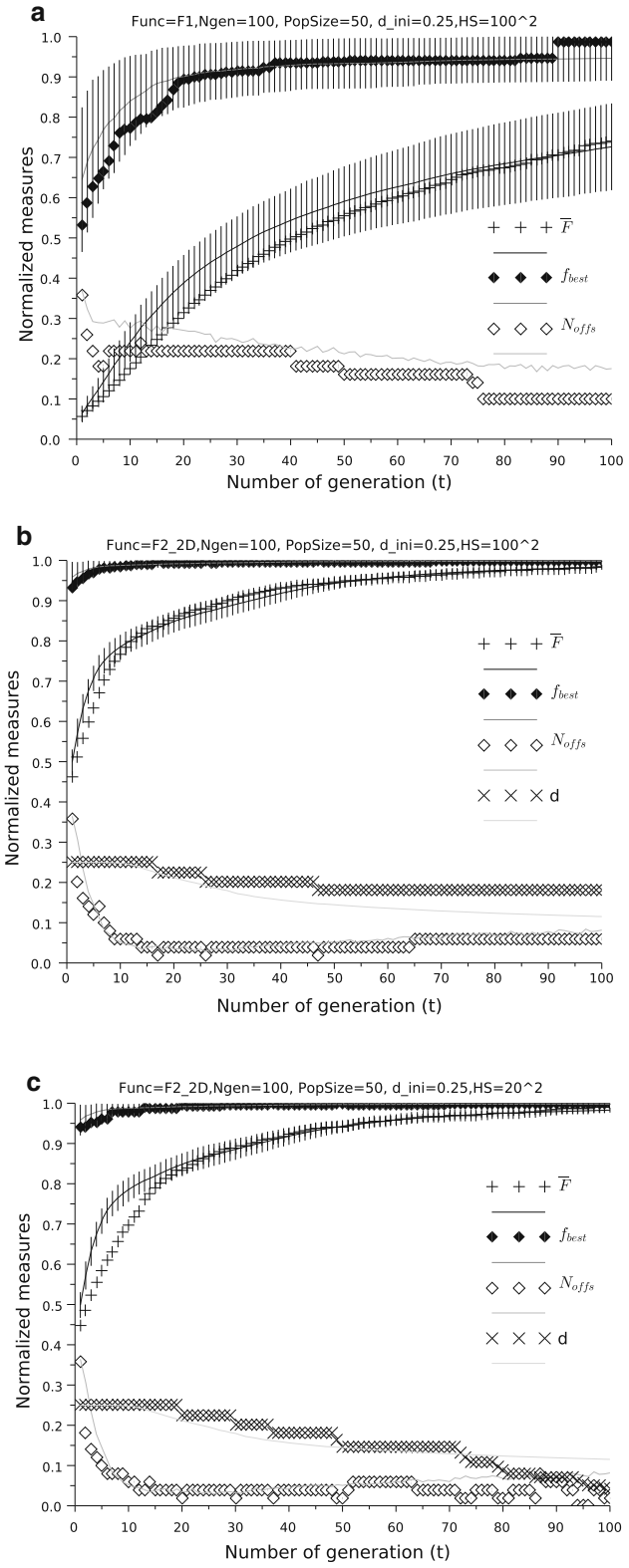

Fig. 8 Model predictions and real values at each generation for the functions 2D: F_1 (up), F_2 using $\#u = 100^2$ (middle), and F_2 using $\#u = 20^2$ (down). Model measures are represented by points, and the curves represent real measures

set of functions used to compare model predictions with measures taken from algorithm runs. A useful approach (the so-called fitness landscape) is to consider the function as a linear combination of a set of unimodal functions (peaks).

These functions (Mañ and Sole 1999) called $F_1 - F_5$ are defined as follows:

$$F_n(x) = \sum_{i=1}^{m} h_i e^{-\sum_{k=1}^{n} (x_k - P_{ik})^2 / w_i} \quad (30)$$

where m is the number of peaks, $x_k \in [0, 100]^n$; n is the dimensionality, P_{ik} is the location of local optima, h_i is the height of each peak, and w_i is its width.

A first example of function denoted by F_1 is a bidimensional function with five peaks. These peaks have a regular shape and are clearly distributed within search space. The function F_2 is also bidimensional but it exhibits more rugged landscape whose peaks have a more complex shape. This function has been added input variables to build functions of 3, 4 and 5 dimensions named as F_3, F_4 and F_5, respectively. Figure 5 plots the appearance of the functions F_1 and F_2, and Table 1 shows these values for P_{ik}, w_i and h_i which define the features of each function.

9 Numerical results

In the performed experiments, we use as population size $N = 50$, number of generations $G = 100$, sBLX-a with $a = 0$, the incest threshold d is an Euclidean distance whose initial value is $d_{ini} = 0.25$, and the adaptive mechanism does reduce the parameter d by multiplying by 0.9.

The hypermatrix sizes used by the model are shown in Table 2. Specifically, third column of this table summarizes hypermatrix sizes with fitness-value distribution (denoted by #F). These values are calculated as $5 \cdot 10^7 / c^n$. Hence, we limit its size to $5 \cdot 10^7$ cells. Table 2 also shows the sizes of probability distribution hypermatrices (denoted by #u) which are used in our experiments to assess their influence on model performance.

Time measures of Table 2 have been obtained using a PC with a CPU at 2.0 GHz and 500 MB of RAM memory. As a result, this table illustrates the proposed model is computationally tractable. As usual in simulation of theoretical models, time required by the model (t_{model}) is higher compared to a single run of algorithm (t_{run}), especially when using larger matrices and functions of higher dimensionality. Unlike what happens with the model, the algorithm performance is worse by increasing population size. Also, many runs of the algorithm are required to

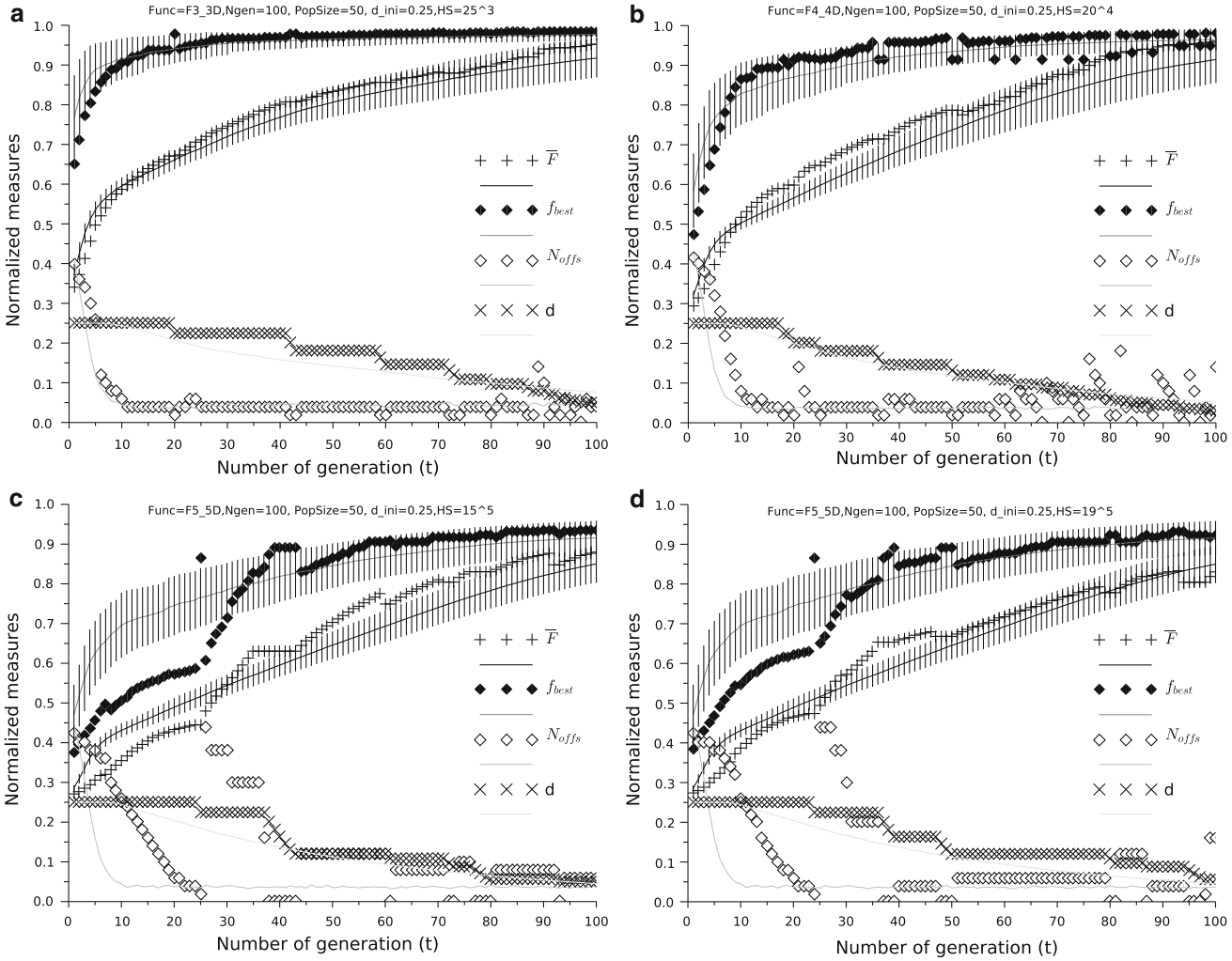

Fig. 9 Model predictions and real values at each generation for the functions 3D to 5D: F_3 and F_4 (up), F_5 using $\#u = 19^5$ and $\#u = 15^5$ (down). Model measures are represented by points, and the curves represent real measures

statistically estimate its average behavior and to reduce the effect of genetic drift in results.

In order to illustrate the accuracy of predictions in distribution of locations of population individuals, we show in Figs. 6 and 7 the expected probability mass obtained by the model and real individual locations achieved by one algorithm run. Four stages of optimization process are represented in these Þgures for test functions F_1 (see Fig. 6) and F_2 (see Fig. 7).

Figure 6 shows that the proposed model is able to make predictions about which clusters are active in each time step, locations of these clusters and their sizes. Moreover, Fig. 7 illustrates ability of the model to capture the dynamics of the shape of the clusters, although the peaks of function do not provide such a regular appearance on the function landscape as shown by the function F_1.

Several experiments have been performed to compare quantitatively the model predictions with average measures taken from runs of the CHC algorithm. In particular, we compare the number of offspring $\bar{N}_{offs}(t)$, mean of the population Þtness $\bar{F}(t)$ and Þtness value of the best solution found $f_{best}(t)$. These experiments are mainly focused on the inßuence of dimensionality and hypermatrix sizes on the quality of model predictions. Some results achieved from these experiments are summarized in Table 3 and plotted in Figs. 8 and 9. Ñdata are collected in Tables 6, 7, 8, 9, 10, 11 and 12 included in Appendix. In these Þgures, the predictions given by the model are represented by points and the curves represent real measures by averaging 300 algorithm runs.

Figure 8 shows results for functions F_1 and F_2: The pictures of this Þgure show an accurate quantitative Þt of model predictions compared to real measures taken from algorithm runs. Despite the more rugged appearance of the function F_2, the adjustment between real and model

Table 6 Model predictions and average simulation data at each generation for the function F_1

t	Model predictions				Averaged measures using real CHC				Differences	
	$\overline{F}$	f_{best}	N_{offs}	d	$\overline{F}$	f_{best}	N_{offs}	d	$\Delta\overline{F}$	Δf_{best}
1	0.0554	0.5327	0.36	0.25	0.0612 0.0198	0.6368 0.1869	0.36	0.25	0.0058	0.1041
5	0.1066	0.6644	0.18	0.25	0.1410 0.0369	0.7578 0.1516	0.29	0.25	0.0344	0.0935
10	0.1761	0.7719	0.22	0.25	0.2414 0.0539	0.8304 0.1221	0.28	0.25	0.0652	0.0584
15	0.2472	0.8138	0.22	0.25	0.3248 0.0645	0.8713 0.0978	0.28	0.25	0.0775	0.0575
20	0.3134	0.8942	0.22	0.25	0.3930 0.0705	0.8949 0.0784	0.27	0.25	0.0796	0.0007
25	0.3690	0.9072	0.22	0.25	0.4458 0.0733	0.9082 0.0705	0.26	0.25	0.0768	0.0010
30	0.4174	0.9138	0.22	0.25	0.4893 0.0759	0.9164 0.0665	0.25	0.25	0.0719	0.0026
35	0.4605	0.9221	0.22	0.25	0.5257 0.0784	0.9229 0.0616	0.24	0.25	0.0653	0.0009
40	0.4980	0.9340	0.22	0.25	0.5563 0.0818	0.9280 0.0587	0.24	0.25	0.0583	0.0060
45	0.5278	0.9356	0.18	0.25	0.5818 0.0846	0.9334 0.0566	0.23	0.25	0.0541	0.0021
50	0.5546	0.9382	0.16	0.25	0.6052 0.0880	0.9370 0.0557	0.22	0.25	0.0506	0.0012
55	0.5792	0.9402	0.16	0.25	0.6260 0.0899	0.9396 0.0547	0.21	0.25	0.0467	0.0006
60	0.6016	0.9402	0.16	0.25	0.6448 0.0917	0.9433 0.0520	0.21	0.25	0.0433	0.0031
65	0.6223	0.9402	0.16	0.25	0.6621 0.0939	0.9459 0.0503	0.20	0.25	0.0398	0.0057
70	0.6445	0.9402	0.16	0.25	0.6779 0.0956	0.9474 0.0499	0.19	0.25	0.0334	0.0072
75	0.6651	0.9402	0.14	0.25	0.6927 0.0970	0.9499 0.0494	0.19	0.25	0.0275	0.0097
80	0.6752	0.9402	0.10	0.25	0.7060 0.0983	0.9514 0.0493	0.18	0.25	0.0308	0.0111
85	0.6932	0.9466	0.10	0.25	0.7186 0.0992	0.9527 0.0490	0.18	0.25	0.0254	0.0060
90	0.7064	0.9872	0.10	0.25	0.7295 0.1001	0.9537 0.0490	0.18	0.25	0.0231	0.0335
95	0.7283	0.9872	0.10	0.25	0.7402 0.1009	0.9549 0.0485	0.18	0.25	0.0119	0.0323
96	0.7324	0.9872	0.10	0.25	0.7423 0.1010	0.9550 0.0483	0.18	0.25	0.0099	0.0321
97	0.7345	0.9872	0.10	0.25	0.7444 0.1009	0.9554 0.0480	0.18	0.25	0.0100	0.0318
98	0.7361	0.9872	0.10	0.25	0.7463 0.1010	0.9556 0.0479	0.18	0.25	0.0102	0.0315
99	0.7383	0.9872	0.10	0.25	0.7483 0.1012	0.9559 0.0476	0.18	0.25	0.0100	0.0312
100	0.7400	0.9872	0.10	0.25	0.7501 0.1010	0.9561 0.0477	0.17	0.25	0.0101	0.0311

$G = 100$; $N = 50$; $d_{ini} = 0.25$; $HS = 100^2$: All values are normalized

measures is also quite accurate. Specifically, the mean of population fitness and the best fitness values are in the worst case within the range that provides the standard deviation.

Figure 8b and c show the results by using hypermatrices of sizes $\#u = 100^2$ and $\#u = 20^2$ for function F_2: A lower value for $\#u$ introduces errors in estimating the fitness average, whose influence is more pronounced in the early stages of optimization.

Our model works with real-coded measures to estimate the averaged number of offspring. Considering only one algorithm run, this measure is a discrete value that shows oscillations and it reaches zero more often, but softened when averaged. Therefore, to make predictions about d is more complicated in final stages, according to the results shown in Fig. 8.

Results for functions F_3 – F_5 are shown Fig. 9. As shown in Fig. 8, these results also show a quantitative fit of model predictions, although we observe a slight worsening with increasing dimensionality of the function. The reason is that the matrix size $\#F$ is a constant but the number of discretization intervals decreases with increasing dimensionality. In Fig. 9c and d, we compare the results for function F_5 achieved by using hypermatrices of sizes $\#u = 15^5$ and $\#u = 19^5$; respectively. As in the case of F_2; a lower value for $\#u$ introduces an error in estimating the fitness average and the best fitness, but in this function its influence is seen in all stages.

9.1 Statistical analysis

Table 3 summarizes average numerical results for $\overline{F}^{dif}$ and f_{best}^{dif}: Specifically, this table shows prediction errors calculated as differences between average real measures and model predictions. These differences are averaged for all generations t and denoted by $\Delta\overline{F}$ and Δf_{best}: Each column in the table shows prediction errors by using different hypermatrix size $\#u$: The last column shows standard

Table 7 Model predictions and average simulation data at each generation for the function F_2

t	Model predictions				Averaged measures using real CHC				Differences	
	$\overline{F}$	f_{best}	N_{offs}	d	$\overline{F}$	f_{best}	N_{offs}	d	$\Delta\overline{F}$	Δf_{best}
1	0.4618	0.9329	0.36	0.25	0.4959 0.0329	0.9510 0.0429	0.36	0.25	0.0341	0.0181
5	0.6341	0.9697	0.12	0.25	0.7047 0.0300	0.9805 0.0175	0.14	0.25	0.0706	0.0108
10	0.7672	0.9846	0.06	0.25	0.7833 0.0261	0.9872 0.0116	0.05	0.25	0.0161	0.0025
15	0.8293	0.9902	0.04	0.25	0.8194 0.0248	0.9898 0.0093	0.04	0.23	0.0099	0.0005
20	0.8564	0.9924	0.04	0.23	0.8457 0.0230	0.9914 0.0079	0.03	0.22	0.0107	0.0010
25	0.8811	0.9939	0.04	0.23	0.8660 0.0228	0.9928 0.0063	0.03	0.20	0.0151	0.0011
30	0.8990	0.9946	0.04	0.20	0.8834 0.0222	0.9937 0.0055	0.03	0.18	0.0155	0.0009
35	0.9187	0.9955	0.04	0.20	0.9003 0.0219	0.9944 0.0050	0.04	0.16	0.0183	0.0011
40	0.9311	0.9959	0.04	0.20	0.9165 0.0205	0.9954 0.0039	0.04	0.16	0.0146	0.0005
45	0.9399	0.9962	0.04	0.20	0.9306 0.0183	0.9960 0.0033	0.05	0.15	0.0092	0.0002
50	0.9457	0.9962	0.04	0.18	0.9418 0.0155	0.9964 0.0030	0.05	0.14	0.0039	0.0001
55	0.9517	0.9965	0.04	0.18	0.9506 0.0135	0.9967 0.0027	0.06	0.14	0.0011	0.0002
60	0.9562	0.9966	0.04	0.18	0.9575 0.0118	0.9969 0.0024	0.06	0.14	0.0013	0.0003
65	0.9602	0.9969	0.06	0.18	0.9629 0.0106	0.9971 0.0023	0.06	0.13	0.0027	0.0002
70	0.9649	0.9972	0.06	0.18	0.9672 0.0097	0.9972 0.0021	0.07	0.13	0.0023	0.0001
75	0.9694	0.9973	0.06	0.18	0.9708 0.0092	0.9974 0.0019	0.07	0.13	0.0014	0.0000
80	0.9728	0.9975	0.06	0.18	0.9737 0.0086	0.9974 0.0019	0.07	0.12	0.0009	0.0000
85	0.9757	0.9977	0.06	0.18	0.9761 0.0081	0.9975 0.0019	0.07	0.12	0.0004	0.0002
90	0.9784	0.9978	0.06	0.18	0.9782 0.0077	0.9976 0.0018	0.07	0.12	0.0002	0.0002
95	0.9809	0.9978	0.06	0.18	0.9800 0.0072	0.9976 0.0017	0.07	0.12	0.0009	0.0002
96	0.9811	0.9979	0.06	0.18	0.9803 0.0072	0.9976 0.0017	0.07	0.12	0.0008	0.0002
97	0.9813	0.9979	0.06	0.18	0.9807 0.0071	0.9977 0.0017	0.07	0.12	0.0006	0.0002
98	0.9823	0.9980	0.06	0.18	0.9809 0.0070	0.9977 0.0017	0.07	0.12	0.0014	0.0004
99	0.9827	0.9981	0.06	0.18	0.9812 0.0070	0.9977 0.0017	0.07	0.12	0.0014	0.0005
100	0.9830	0.9981	0.06	0.18	0.9815 0.0069	0.9977 0.0017	0.08	0.12	0.0015	0.0004

$G = 100$; $N = 50$; $d_{ini} = 0.25$; $HS = 100^2$: All values are normalized

deviations obtained in 300 algorithm runs also averaged for all generations t. In accordance with previous results, this table shows numerically that errors obtained by using these matrix size are within the range of the standard deviations.

To statistically validate the significance of these results, we consider the curves depicted by each measure, ($\overline{F}^{obt}$, f_{best}^{obt}, and N_{offs}^{obt}). In particular, these curves are quantified as the area below them for each algorithm run and model prediction related. Table 4 summarizes areas below curve for model predictions according to hypermatrix size ($\overline{F}^{pred}$, f_{best}^{pred}, N_{offs}^{pred}) in comparison to the curve areas by averaging 300 runs of real-coded CHC algorithm.

We use a non-parametric test to validate if value distributions of these areas show significant differences to areas predicted by the model. The p values obtained by applying the Wilcoxon test in each case are collected in Table 5. In all cases p values are above 0.05. That means the non-parametric test does not detect a significant evidence for a difference between actual run measures and model predictions at the $\alpha = 0.05$ significance level. Therefore, statistical analysis does not prove significant differences between actual behavior of real-coded CHC algorithm and predictions performed by our model.

10 Conclusions

This paper proposes a theoretical model to describe the behavior of the real-coded CHC algorithm in multi-peaked landscape functions for several dimensions. The proposed model works with dynamical probability distributions, which were implemented by hypermatrices on a search space dynamically discretized. Several experiments have shown the ability of the model to capture the full dynamics of the shape of the individual clusters, even when the function landscape has an irregular appearance.

We have shown the feasibility of using small matrices to make good predictions in functions up to 5D, within a time computationally tractable, where errors in prediction can be

Table 8 Model predictions and average simulation data at each generation for the function F_2

t	Model predictions				Averaged measures using real CHC				Differences	
	$\overline{F}$	f_{best}	N_{offs}	d	$\overline{F}$	f_{best}	N_{offs}	d	$\Delta\overline{F}$	Δf_{best}
1	0.4478	0.9396	0.36	0.25	0.4959 0.0329	0.9510 0.0429	0.36	0.25	0.0481	0.0114
5	0.5839	0.9612	0.10	0.25	0.7047 0.0300	0.9805 0.0175	0.14	0.25	0.1208	0.0193
10	0.6977	0.9793	0.06	0.25	0.7833 0.0261	0.9872 0.0116	0.05	0.25	0.0856	0.0079
15	0.7884	0.9870	0.04	0.25	0.8194 0.0248	0.9898 0.0093	0.04	0.23	0.0310	0.0028
20	0.8322	0.9864	0.02	0.23	0.8457 0.0230	0.9914 0.0079	0.03	0.22	0.0136	0.0050
25	0.8695	0.9922	0.04	0.23	0.8660 0.0228	0.9928 0.0063	0.03	0.20	0.0035	0.0006
30	0.8888	0.9870	0.02	0.20	0.8834 0.0222	0.9937 0.0055	0.03	0.18	0.0054	0.0067
35	0.9082	0.9959	0.04	0.20	0.9003 0.0219	0.9944 0.0050	0.04	0.16	0.0079	0.0015
40	0.9228	0.9969	0.04	0.18	0.9165 0.0205	0.9954 0.0039	0.04	0.16	0.0062	0.0015
45	0.9363	0.9969	0.04	0.18	0.9306 0.0183	0.9960 0.0033	0.05	0.15	0.0057	0.0009
50	0.9419	0.9949	0.02	0.15	0.9418 0.0155	0.9964 0.0030	0.05	0.14	0.0000	0.0015
55	0.9523	0.9970	0.06	0.15	0.9506 0.0135	0.9967 0.0027	0.06	0.14	0.0017	0.0003
60	0.9591	0.9970	0.06	0.15	0.9575 0.0118	0.9969 0.0024	0.06	0.14	0.0016	0.0001
65	0.9653	0.9968	0.04	0.15	0.9629 0.0106	0.9971 0.0023	0.06	0.13	0.0024	0.0003
70	0.9688	0.9977	0.04	0.15	0.9672 0.0097	0.9972 0.0021	0.07	0.13	0.0016	0.0005
75	0.9702	0.9977	0.04	0.11	0.9708 0.0092	0.9974 0.0019	0.07	0.13	0.0006	0.0004
80	0.9742	0.9966	0.02	0.09	0.9737 0.0086	0.9974 0.0019	0.07	0.12	0.0005	0.0008
85	0.9790	0.9977	0.04	0.08	0.9761 0.0081	0.9975 0.0019	0.07	0.12	0.0029	0.0002
90	0.9851	0.9982	0.06	0.07	0.9782 0.0077	0.9976 0.0018	0.07	0.12	0.0069	0.0007
95	0.9877	0.9966	0.00	0.06	0.9800 0.0072	0.9976 0.0017	0.07	0.12	0.0077	0.0010
96	0.9877	0.9966	0.00	0.05	0.9803 0.0072	0.9976 0.0017	0.07	0.12	0.0074	0.0010
97	0.9889	0.9984	0.04	0.05	0.9807 0.0071	0.9977 0.0017	0.07	0.12	0.0083	0.0007
98	0.9894	0.9984	0.04	0.05	0.9809 0.0070	0.9977 0.0017	0.07	0.12	0.0085	0.0007
99	0.9894	0.9966	0.02	0.05	0.9812 0.0070	0.9977 0.0017	0.07	0.12	0.0082	0.0011
100	0.9894	0.9966	0.02	0.04	0.9815 0.0069	0.9977 0.0017	0.08	0.12	0.0079	0.0011

$G = 100$, $N = 50$, $d_{ini} = 0.25$, $HS = 20^2$. All values are normalized

controlled by adjusting discretization and population size. Our results make possible accurate quantitative predictions about some performance measures such as average fitness, the best fitness reached or number of fitness function evaluations. We formulate a model to make quantitative predictions about the actual behavior of the real-coded CHC algorithm which do not show statistically significant differences.

Further work can be focused in two lines: convergence studies, and to extend the model to a wider set of functions and algorithms. The former concerns the ability of the model to estimate probability of finding individuals in clusters. These predictions could be used to study optima population sizes and as well to predict local optima falling. For instance, the latter would include to making predictions in highly ruggedness landscapes. Our approach makes it possible because the operation of cluster crossing is easily parallelizable.

Acknowledgments This work is supported by the Ministerio de Ciencia e Innovación (Spain) under grant TEC2008-02754/TEC and TIN2008-05854.

Appendix: A formulation for hypersphere volumes

The volume of a hypersphere V_H of radius $r = d/2$ is given by $V_H(r) = C_n r^n$ where n is the dimension, $C_n = \pi^{n/2}/\Gamma(1 + n/2)$ is the unit sphere, and $\Gamma(\cdot)$ is the Gamma function.

The volume for a hypersphere cap V_C of radius r and height h is given by

$$V_C(r, h) = C_n r^n \left(\frac{1}{2} - \frac{r - h}{r} \frac{\Gamma\left(1 + \frac{n}{2}\right)}{\sqrt{\pi}\,\Gamma\left(\frac{n+1}{2}\right)} {}_2F_1(a, b; c; z) \right) \quad (31)$$

where $a = \frac{1}{2}$, $b = \frac{1-n}{2}$, $c = \frac{3}{2}$, $z = \left(\frac{r-h}{r}\right)^2$. ${}_2F_1$ is a hypergeometric function defined by the next series:

$${}_2F_1(a, b; c; z) = \sum_{i=0}^{1} \frac{(a)_n (b)_n z^n}{(c)_n n!} \quad (32)$$

when it meets z that in this case it is $z > 0$. Therefore, $(x)_n$ is given as follows:

Table 9 Model predictions and average simulation data at each generation for the function F_5

t	Model predictions				Averaged measures using real CHC					Differences	
	$\overline{F}$	f_{best}	N_{offs}	d	$\overline{F}$	f_{best}		N_{offs}	d	$\Delta\overline{F}$	Δf_{best}
1	0.3392	0.6520	0.40	0.25	0.3733	0.0205	0.7621	0.1091	0.40	0.25	0.0341	0.1101
5	0.4957	0.8331	0.26	0.25	0.5327	0.0222	0.8890	0.0632	0.13	0.25	0.0370	0.0559
10	0.5879	0.9065	0.06	0.25	0.5947	0.0214	0.9173	0.0486	0.05	0.24	0.0069	0.0108
15	0.6391	0.9378	0.04	0.25	0.6303	0.0233	0.9312	0.0402	0.04	0.23	0.0088	0.0066
20	0.6699	0.9638	0.02	0.23	0.6629	0.0261	0.9415	0.0343	0.04	0.21	0.0070	0.0222
25	0.7128	0.9545	0.04	0.23	0.6928	0.0284	0.9488	0.0307	0.04	0.19	0.0200	0.0057
30	0.7475	0.9653	0.04	0.23	0.7213	0.0308	0.9543	0.0273	0.04	0.18	0.0263	0.0110
35	0.7741	0.9664	0.04	0.23	0.7472	0.0329	0.9586	0.0245	0.04	0.17	0.0269	0.0078
40	0.8019	0.9692	0.04	0.23	0.7705	0.0347	0.9616	0.0229	0.04	0.16	0.0315	0.0076
45	0.8144	0.9711	0.04	0.18	0.7917	0.0364	0.9642	0.0215	0.04	0.15	0.0227	0.0070
50	0.8334	0.9717	0.04	0.18	0.8103	0.0381	0.9664	0.0206	0.04	0.14	0.0231	0.0053
55	0.8479	0.9767	0.04	0.18	0.8266	0.0398	0.9678	0.0203	0.04	0.13	0.0213	0.0088
60	0.8554	0.9774	0.02	0.15	0.8411	0.0416	0.9695	0.0194	0.04	0.12	0.0142	0.0079
65	0.8695	0.9782	0.04	0.15	0.8546	0.0429	0.9706	0.0191	0.04	0.12	0.0150	0.0075
70	0.8817	0.9798	0.04	0.15	0.8669	0.0441	0.9721	0.0178	0.04	0.11	0.0149	0.0077
75	0.8839	0.9793	0.04	0.11	0.8782	0.0453	0.9731	0.0173	0.04	0.10	0.0058	0.0062
80	0.8975	0.9845	0.02	0.10	0.8884	0.0460	0.9742	0.0171	0.04	0.10	0.0091	0.0103
85	0.9167	0.9836	0.04	0.10	0.8979	0.0468	0.9753	0.0166	0.04	0.09	0.0188	0.0083
90	0.9393	0.9845	0.10	0.08	0.9066	0.0468	0.9761	0.0163	0.04	0.08	0.0326	0.0083
95	0.9433	0.9853	0.02	0.06	0.9148	0.0464	0.9769	0.0157	0.04	0.08	0.0284	0.0083
96	0.9457	0.9842	0.04	0.06	0.9163	0.0464	0.9772	0.0155	0.04	0.08	0.0294	0.0070
97	0.9457	0.9853	0.00	0.05	0.9177	0.0463	0.9775	0.0155	0.04	0.08	0.0280	0.0078
98	0.9489	0.9844	0.06	0.05	0.9193	0.0462	0.9776	0.0155	0.04	0.07	0.0296	0.0068
99	0.9515	0.9849	0.04	0.05	0.9207	0.0461	0.9777	0.0154	0.04	0.07	0.0307	0.0072
100	0.9536	0.9850	0.04	0.05	0.9222	0.0459	0.9778	0.0154	0.04	0.07	0.0314	0.0072

$G = 100$; $N = 50$; $d_{ini} = 0.25$; $HS = 25^3$: All values are normalized

Table 10 Model predictions and average simulation data at each generation for the function F_6

t	Model predictions				Averaged measures using real CHC					Differences	
	$\overline{F}$	f_{best}	N_{offs}	d	$\overline{F}$	f_{best}		N_{offs}	d	$\Delta\overline{F}$	Δf_{best}
1	0.2950	0.4736	0.42	0.25	0.3214	0.0112	0.5871	0.0963	0.43	0.25	0.0264	0.1135
5	0.3981	0.6894	0.32	0.25	0.4499	0.0167	0.7675	0.0905	0.13	0.25	0.0517	0.0781
10	0.5186	0.8642	0.08	0.25	0.5034	0.0171	0.8291	0.0778	0.04	0.24	0.0152	0.0350
15	0.5753	0.8933	0.04	0.25	0.5357	0.0198	0.8548	0.0687	0.04	0.22	0.0396	0.0385
20	0.5980	0.9571	0.02	0.20	0.5660	0.0218	0.8719	0.0606	0.04	0.20	0.0320	0.0852
25	0.6479	0.9571	0.02	0.18	0.5983	0.0263	0.8863	0.0556	0.04	0.18	0.0496	0.0708
30	0.6879	0.9324	0.04	0.18	0.6283	0.0304	0.9021	0.0468	0.03	0.16	0.0596	0.0303
35	0.7151	0.9585	0.04	0.18	0.6571	0.0346	0.9130	0.0422	0.04	0.15	0.0580	0.0454
40	0.7397	0.9585	0.04	0.15	0.6853	0.0389	0.9225	0.0385	0.04	0.14	0.0544	0.0360
45	0.7687	0.9620	0.04	0.15	0.7123	0.0426	0.9293	0.0353	0.03	0.13	0.0565	0.0328
50	0.7860	0.9571	0.02	0.13	0.7377	0.0472	0.9348	0.0333	0.04	0.12	0.0483	0.0223
55	0.7934	0.9589	0.04	0.12	0.7636	0.0521	0.9409	0.0324	0.04	0.11	0.0298	0.0179
60	0.8175	0.9620	0.06	0.11	0.7875	0.0564	0.9466	0.0294	0.04	0.10	0.0300	0.0155
65	0.8480	0.9682	0.06	0.10	0.8099	0.0599	0.9508	0.0278	0.04	0.09	0.0381	0.0174

Table 10 continued

t	Model predictions				Averaged measures using real CHC					Differences	
	$\overline{F}$	f_{best}	N_{offs}	d	$\overline{F}$	f_{best}		N_{offs}	d	$\Delta\overline{F}$	Δf_{best}
70	0.8779	0.9714	0.06	0.09	0.82980.0619	0.9537	0.0278	0.04	0.09	0.0481	0.0177
75	0.8881	0.9780	0.02	0.07	0.84870.0636	0.9561	0.0279	0.04	0.08	0.0394	0.0219
80	0.9229	0.9780	0.02	0.06	0.86640.0632	0.9587	0.0273	0.04	0.07	0.0564	0.0193
85	0.9372	0.9765	0.04	0.05	0.88100.0630	0.9604	0.0269	0.04	0.07	0.0563	0.0161
90	0.9504	0.9774	0.10	0.05	0.89380.0618	0.9618	0.0269	0.03	0.06	0.0565	0.0156
95	0.9534	0.9774	0.12	0.04	0.90520.0607	0.9636	0.0266	0.03	0.06	0.0482	0.0138
96	0.9564	0.9774	0.08	0.04	0.90720.0602	0.9639	0.0267	0.03	0.06	0.0493	0.0134
97	0.9564	0.9780	0.00	0.03	0.90900.0601	0.9641	0.0266	0.03	0.05	0.0475	0.0139
98	0.9583	0.9798	0.04	0.03	0.91110.0596	0.9645	0.0265	0.04	0.05	0.0471	0.0152
99	0.9583	0.9780	0.02	0.03	0.91320.0590	0.9649	0.0265	0.04	0.05	0.0451	0.0131
100	0.9617	0.9798	0.14	0.03	0.91520.0586	0.9651	0.0265	0.04	0.05	0.0465	0.0147

$G = 100; N = 50; d_{ini} = 0.25; HS = 20^4$: All values are normalized

Table 11 Model predictions and average simulation data at each generation for the function F_6

t	Model predictions				Averaged measures using real CHC					Differences	
	$\overline{F}$	f_{best}	N_{offs}	d	$\overline{F}$	f_{best}		N_{offs}	d	$\Delta\overline{F}$	Δf_{best}
1	0.2714	0.3759	0.42	0.25	0.28600.0076	0.4701	0.0757	0.45	0.25	0.0146	0.0941
5	0.3059	0.4555	0.38	0.25	0.38170.0177	0.6249	0.0807	0.16	0.25	0.0757	0.1694
10	0.3563	0.5056	0.26	0.25	0.43160.0176	0.6974	0.0820	0.04	0.25	0.0752	0.1918
15	0.4050	0.5507	0.16	0.25	0.46030.0198	0.7204	0.0738	0.04	0.22	0.0552	0.1698
20	0.4328	0.5736	0.06	0.25	0.48950.0233	0.7465	0.0754	0.04	0.20	0.0567	0.1729
25	0.4435	0.8662	0.02	0.23	0.51660.0249	0.7669	0.0736	0.03	0.18	0.0731	0.0992
30	0.5455	0.7159	0.38	0.23	0.54480.0269	0.7852	0.0678	0.04	0.17	0.0007	0.0693
35	0.6306	0.8273	0.30	0.23	0.57010.0285	0.8014	0.0661	0.04	0.15	0.0605	0.0259
40	0.6318	0.8662	0.00	0.16	0.59600.0292	0.8192	0.0644	0.04	0.14	0.0358	0.0470
45	0.6554	0.8338	0.12	0.12	0.62080.0317	0.8326	0.0595	0.03	0.12	0.0346	0.0012
50	0.7031	0.8666	0.12	0.12	0.64610.0351	0.8463	0.0557	0.03	0.11	0.0570	0.0203
55	0.7445	0.8905	0.12	0.12	0.67040.0377	0.8594	0.0525	0.03	0.10	0.0741	0.0311
60	0.7485	0.9049	0.12	0.12	0.69440.0406	0.8684	0.0512	0.04	0.09	0.0540	0.0365
65	0.7809	0.9044	0.08	0.11	0.71750.0420	0.8772	0.0497	0.04	0.09	0.0634	0.0273
70	0.8099	0.9161	0.08	0.11	0.74130.0434	0.8855	0.0474	0.04	0.08	0.0686	0.0307
75	0.8235	0.9161	0.10	0.09	0.76300.0445	0.8940	0.0459	0.04	0.07	0.0605	0.0221
80	0.8292	0.9352	0.00	0.06	0.78430.0456	0.9005	0.0440	0.04	0.06	0.0448	0.0347
85	0.8570	0.9329	0.08	0.06	0.80420.0464	0.9062	0.0424	0.04	0.06	0.0528	0.0267
90	0.8746	0.9336	0.08	0.06	0.82190.0466	0.9103	0.0429	0.04	0.05	0.0528	0.0233
95	0.8607	0.9329	0.06	0.05	0.83720.0467	0.9134	0.0427	0.03	0.05	0.0235	0.0195
96	0.8661	0.9329	0.06	0.05	0.84030.0467	0.9140	0.0425	0.04	0.05	0.0258	0.0189
97	0.8710	0.9336	0.06	0.05	0.84310.0466	0.9148	0.0422	0.04	0.05	0.0279	0.0188
98	0.8746	0.9336	0.06	0.05	0.84580.0465	0.9152	0.0422	0.03	0.05	0.0288	0.0184
99	0.8773	0.9336	0.06	0.05	0.84860.0467	0.9161	0.0422	0.04	0.05	0.0287	0.0175
100	0.8798	0.9336	0.06	0.05	0.85100.0470	0.9165	0.0425	0.04	0.04	0.0288	0.0171

$G = 100; N = 50; d_{ini} = 0.25; HS = 15^5$: All values are normalized

Table 12 Model predictions and average simulation data at each generation for the function F_6. All values are normalized

t	Model predictions				Averaged measures using real CHC				Differences	
	$\overline{F}$	f_{best}	N_{offs}	d	$\overline{F}$	f_{best}		N_{offs}	d	$\Delta\overline{F}$	Δf_{best}
1	0.2731	0.3844	0.42	0.25	0.2861	0.0076	0.4701	0.0757	0.45	0.25	0.0130	0.0857
5	0.3106	0.4693	0.38	0.25	0.3817	0.0177	0.6249	0.0807	0.16	0.25	0.0711	0.1556
10	0.3858	0.5474	0.26	0.25	0.4316	0.0176	0.6974	0.0820	0.04	0.25	0.0458	0.1500
15	0.4385	0.5981	0.14	0.25	0.4603	0.0198	0.7204	0.0738	0.04	0.22	0.0218	0.1224
20	0.4655	0.6206	0.06	0.25	0.4895	0.0233	0.7465	0.0754	0.04	0.20	0.0240	0.1258
25	0.4945	0.6516	0.44	0.23	0.5166	0.0249	0.7669	0.0736	0.03	0.18	0.0221	0.1154
30	0.5723	0.7724	0.30	0.23	0.5448	0.0269	0.7852	0.0678	0.04	0.17	0.0275	0.0128
35	0.6290	0.8055	0.20	0.23	0.5701	0.0285	0.8014	0.0661	0.04	0.15	0.0589	0.0041
40	0.6593	0.8440	0.04	0.16	0.5960	0.0292	0.8192	0.0644	0.04	0.14	0.0633	0.0248
45	0.6766	0.8620	0.04	0.16	0.6208	0.0317	0.8326	0.0595	0.03	0.12	0.0558	0.0294
50	0.6686	0.8991	0.00	0.12	0.6461	0.0351	0.8463	0.0557	0.03	0.11	0.0225	0.0528
55	0.6950	0.8641	0.06	0.12	0.6704	0.0377	0.8594	0.0525	0.03	0.10	0.0246	0.0047
60	0.7180	0.8775	0.06	0.12	0.6944	0.0406	0.8684	0.0512	0.04	0.09	0.0236	0.0091
65	0.7404	0.8905	0.06	0.12	0.7175	0.0420	0.8772	0.0497	0.04	0.09	0.0229	0.0134
70	0.7615	0.9044	0.06	0.12	0.7413	0.0434	0.8855	0.0474	0.04	0.08	0.0202	0.0189
75	0.7821	0.9049	0.06	0.12	0.7630	0.0445	0.8940	0.0459	0.04	0.07	0.0192	0.0109
80	0.7774	0.9071	0.00	0.11	0.7843	0.0456	0.9005	0.0440	0.04	0.06	0.0070	0.0066
85	0.8134	0.9054	0.12	0.10	0.8042	0.0464	0.9062	0.0424	0.04	0.06	0.0093	0.0009
90	0.8270	0.9233	0.04	0.09	0.8219	0.0466	0.9103	0.0429	0.04	0.05	0.0051	0.0129
95	0.8029	0.9308	0.00	0.08	0.8372	0.0467	0.9134	0.0427	0.03	0.05	0.0343	0.0173
96	0.8029	0.9308	0.00	0.07	0.8403	0.0467	0.9140	0.0425	0.04	0.05	0.0374	0.0168
97	0.8029	0.9308	0.00	0.06	0.8431	0.0466	0.9148	0.0422	0.04	0.05	0.0401	0.0160
98	0.8029	0.9308	0.02	0.06	0.8458	0.0465	0.9152	0.0422	0.03	0.05	0.0429	0.0156
99	0.8191	0.9181	0.16	0.06	0.8486	0.0467	0.9161	0.0422	0.04	0.05	0.0295	0.0019
100	0.8302	0.9233	0.16	0.06	0.8510	0.0470	0.9165	0.0425	0.04	0.04	0.0209	0.0068

$G = 100$; $N = 50$; $d_{ini} = 0.25$; $HS = 19^5$: All values are normalized

$$\sigma(x)_n = \prod_{i=0}^{n-1} \sigma(x+i) \quad (33)$$

The volume of intersection of two hyperspheres can be found by adding the volume of two hyperspherical caps. Let $r1$ and $r2$ be the radius of both hyperspheres that intersect, and d the distance between their centers. Then,

$$V_I(r1, r2, d) = V_C(r1, h1) + V_C(r2, h2) \quad (34)$$

where the heights of the caps $h1$ and $h2$ are

$$h1 = \frac{(r1 + r2 - d)(r2 - r1 + d)}{2d} \quad (35)$$

$$h2 = \frac{(r1 + r2 - d)(r1 - r2 + d)}{2d} \quad (36)$$

References

Barnett L (1998) Collapsing the state space by applying markov analysis to evolutionary systems. In: Workshop presentation at the sixth international conference on artificial life, UCLA

Beyer H, Schwefel H (2002) Evolution strategies—a comprehensive introduction. Nat Comput 1(1):3–52

Beyer HG, Schwefel HP, Wegener I (2002) How to analyse evolutionary algorithms. Theor Comput Sci 287(1):101–130

Bornholdt S (1998) Genetic algorithm dynamics on a rugged landscape. Phys Rev E 57(4):3853–3860

Cano JR, Herrera F, Lozano M (2003) Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study. IEEE Trans Evol Comput 7(6):561–575

Cano JR, Herrera F, Lozano M (2005) Strategies for scaling up evolutionary instance reduction algorithms for data mining. In: Ghosh A, Jain L (eds) Evolutionary computation in data mining. Studies in fuzziness and soft computing, vol 163. Springer, Berlin, pp 21–39

Cervone G, Kaufman KK, Michalski RS (2000) Experimental validations of the learnable evolution model. In: Proceedings of the 2000 congress on evolutionary computation, pp 1064–1071

Chellapilla K, Fogel DB (1999) Fitness distributions in evolutionary computation: motivation and examples in the continuous domain. Biosystems 54(1-2):15–29

Cordón O, Damas S, Santamaría J (2006) Feature-based image registration by means of the chc evolutionary algorithm. Image Vis Comput 24(5):525–533

Delgado M, Pegalajar M, Pegalajar M (2006) Evolutionary training for dynamical recurrent neural networks: an application in

Þnantial time series prediction. Mathware Soft Comput 13:89Ð110
Eiben AE, Rudolph G (1999) Theory of evolutionary algorithms: a birdÕs eye view. Theor Comput Sci 229(1):3Ð9
Eshelman L (1990) The chc adaptive search algorithm. In: Rawlins G (ed) Foundations of genetic algorithms. Morgan Kaufmann, San Francisco, pp 265Ð283
Eshelman L, Caruana A, Schaffer J (1993) Real-coded genetic algorithms and interval-schemata. Found Genetic Algorithms 2:187Ð202
Eshelman LJ, Schaffer JD (1992) Real-coded genetic algorithms and interval-schemata. In: Foundation of genetic algorithms, pp 187Ð202
Eshelman LJ, Schaffer JD (1993) Real-coded genetic algorithms in genetic algorithms by preventing incest. Foundation of genetic algorithms, vol 2, pp 187Ð202
Herrera F, Lozano M, Molina D (2010) Components and parameters of DE, real-coded CHC, and G-CMA-ES. Tech. rep., SCI2S, University of Granada, Spain.http://sci2s.ugr.es/eamhco/descriptions.pdf
Holland J (1992) Adaptation in natural and artiÞcial systems: an introductory analysis with applications to biology, control, and artiÞcial intelligence. 1st edn. MIT Press, Cambridge
Huang W, Wang X (2007) An improved CHC algorithm for damage diagnosis of offshore platforms. J Ocean Univ China 6:85Ð89 (english edition)
Kesur KB (2009) Advances in genetic algorithm optimization of trafÞc signals. J Transport Eng 135(4):160Ð173
Lozano M, Herrera F, Krasnogor N, Molina D (2004) Real-coded memetic algorithms with crossover hill-climbing. Evol Comput 12(3):273Ð302
Luzón M, Barreiro E, Yeguas E, Joan-Arinyo R (2004) GA and CHC two evolutionary algorithms to solve the root identiÞcation problem in geometric constraint solving. In: Bubak M, van Albada GD, Sloot PMA, Dongarra JJ (eds) Computational scienceÑICCS 2004. Lecture notes in computer science, vol 3039. Springer, Berlin, pp 139Ð146
Marón J, Sokol RV (1999) Macroevolutionary algorithms: a new optimization method on Þtness landscapes. IEEE Trans Evol Comput 3(4):272Ð286
Nebro AJ, Alba E, Molina G, Chicano F, Luna F, Durillo JJ (2007) Optimal antenna placement using a new multi-objective chc algorithm. In: GECCO Õ07: proceedings of the 9th annual conference on genetic and evolutionary computation. ACM, New York, NY, USA, pp 876Ð883
Nomura T (1997) An analysis on crossovers for real number chromosomes in an inÞnite population size. In: IJCAIÕ97 proceedings of the Þfteenth international joint conference on artiÞcial intelligence. Morgan Kaufmann, San Francisco, CA, USA, pp 936Ð941
Poli R, Langdon W, Clerc M, Stephens C (2007) Continuous optimisation theory made easy? Finite-element models of evolutionary strategies, genetic algorithms and particle swarm optimizers. In: Stephens C, Toussaint M, Whitley D, Stadler P (eds) Foundations of genetic algorithms. Lecture notes in computer science, vol 4436. Springer, Berlin, pp 165Ð193
Prügel-Bennett A, Rogers A (2001) Modelling GA dynamics. Nat Comput. Springer
Qi X, Palmieri F (1994) Theoretical analysis of evolutionary algorithms with an inÞnite population size in continuous space. Part I: basic properties of selection and mutation. IEEE Trans Neural Netw 5(1):102Ð119
Qi X, Palmieri F (1994) Theoretical analysis of evolutionary algorithms with an inÞnite population size in continuous space. Part II: analysis of the diversiÞcation role of crossover. IEEE Trans Neural Netw 5(1):120Ð129
Sánchez AM, Lozano M, García-Martínez C, Molina D, Herrera F (2008) Real-parameter crossover operators with multiple descendents: an experimental study. Int J Intell Syst 23:246Ð268
Santamaría J, Cordón O, Damas S, García-Torres JM, Quirin A (2009) Performance evaluation of memetic approaches in 3d reconstruction of forensic objects. Soft Comput 13:883Ð904. http://portal.acm.org/citation.cfm?id=1530043.1530048
Schmitt LM (2004) Theory of genetic algorithms II: models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary Þtness function under scaling. Theor Comput Sci 310(1Ð3):181Ð231
Stephens C, Waelbroeck H (1998) Schemata evolution and building blocks. Evol Comput 7:109Ð124
van Nimwegen E, CrutchÞeld JP, Mitchell M (1999) Statistical dynamics of the Royal Road genetic algorithm. Theor Comput Sci 229(1Ð2):41Ð102
Vose MD (1998) The simple genetic algorithm: foundations and theory. MIT Press, Cambridge
Whitley D, Lunacek M, Sokolov A (2006) Comparing the niches of CMA-ES, CHC and pattern search using diverse benchmarks. In: Runarsson T, Beyer HG, Burke E, Merelo-Guervs J, Whitley L, Yao X (eds) Parallel problem solving from natureÑPPSN IX. Lecture notes in computer science, vol 4193. Springer, Berlin, pp 988Ð997
Witt C (2008) Population size versus runtime of a simple evolutionary algorithm. Theor Comput Sci 403(1):104Ð120
Wright AH, Rowe JE, Neil JR (2002) Analysis of the simple genetic algorithm on the single-peak and double-peak landscapes. In: Proceedings of the congress on evolutionary computation (CEC) 2002. IEEE Press, pp 214Ð219
Zhao X, Gao XS, Hu ZC (2007) Evolutionary programming based on non-uniform mutation. Appl Math Comput 192(1):1Ð11