A Probabilistic Cooperative-Competitive Hierarchical Model for Global Optimization

Abstract— **Stochastic searching methods have been widely applied to areas such as global optimization and combinatorial optimization problems in a vast number of disciplines. Existing methods intended to solve these problems by navigating their search on the surface of the rugged landscapes. This is considered insufficient since the property of the landscape at different resolutions can be very different. It is reasonable to adopt appropriate searching strategies at different resolutions. In this paper, we propose a new probabilistic searching model for global optimization. The main contributions of the model are 1) to provide a basis for resolution control and smoothing of search space and 2) to introduce memory into stochastic search. Structurally, the search space is divided into finite number of n-dimensional partitions. The benefits of this organization are twofold. First, the rugged problem landscape can be smoothed, as the hierarchy allows different levels of resolution. The difficulty due to the ruggedness can be decreased. Second, it provides a basis to implement algorithms which dynamically change the 'view' of the landscape on the way of searching. In the presence of feedback, past searching experience obtained can be used to provide guidance to the current search.**

We present results on the algorithm performance in handling numerical function optimization (both high-dimension and deceptive). The empirical results showed that our new model is comparable to that of the other advanced methods in terms of solution quality and computation.

I. Introduction

A. Motivation

Global optimization approaches under the category of stochastic methods such as simulated annealing (SA) [12], [11] and evolutionary algorithms (EAs) [7], [6], [5], [2] and those under the category of heuristic search methods such as multistart greedy descent strategies (MGDs) [20], [17], [8] share several common characteristics:

1 *Landscape Without Different Resolutions*–For SA and MGDs, these algorithms use a single search space at the highest resolution during the search. This type of algorithm is inefficient and inflexible since the search space at the highest resolution is typically huge to be search practically.

2 *Search Without Memorization*–Typically these algorithms do not use information accumulated from the past during the search. Often, these past information may help to guide the search in a more focused and efficient manner.

Motivated by their shortcomings, we present an alternative approach for global optimization.

Suppose we have a balanced binary hierarchy (Fig. 1) of l levels, if we need to make a decision on the branch to traverse next, we will have to make l number of such decisions. Since a branch of the hierarchy leads to a unique non-overlapping sub-hierarchy below it, after making a decision on the branch to go, in principle we just need to consider the corresponding sub-hierarchy in the next decision. It is clear that the size of the hierarchy we are facing is diminishing with the decision made towards the bottom.

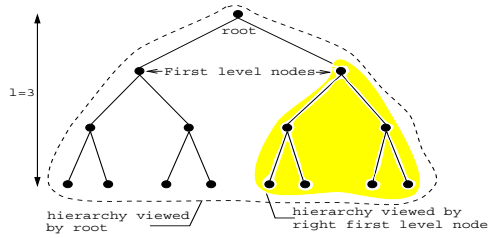


Fig. 1. Search space reduction (First view)

Viewing the hierarchy in another way, if we cut the hierarchy into 2 halves longitudinally at node level $\lfloor l/2 \rfloor$ as shown in Fig. 2, the number of leaf nodes faced by all sub-hierarchies at the upper half are reduced to $2^{\lfloor l/2 \rfloor}$. Those in the lower half are, however, kept unchanged as mentioned before. In general, if we cut the hierarchy successively at each node level in a top-down manner, total number of 'leaf nodes' faced are $2l$. It can be seen that the apparent size of the hierarchy can be reduced drastically.

The formation of such a hierarchy basically defined $l+1$ number of *resolution levels* of the solution landscape. Node level upper in the hierarchy represents a coarse landscape revealing the general macroscopic view, while node level lower in the hierarchy represents a fine landscape revealing the detail. This resolution hierarchy allows an algorithm designed to concentrate on the searching at the lower resolution, which is easier, locating the promising area first and to drive into the precise optimum later at the higher resolution when the search is converging.

B. Paper Organization

Section II describes the essential idea of the probabilistic cooperative-competitive hierarchical model. Section **??** is the formulation of the model. Section **??** shows the experimental results carried to illustrate the behavior and performance of the model. algorithms.

II. Search Space Partitioning

A typical function optimization can be expressed as follows. Given a n-dimensional continuous real-value function

$$F : X \longrightarrow \mathbb{R} \qquad \text{where} \quad X \subseteq \mathbb{R}^n \qquad \text{and} \quad x^l \leq X \leq x^u \tag{1}$$

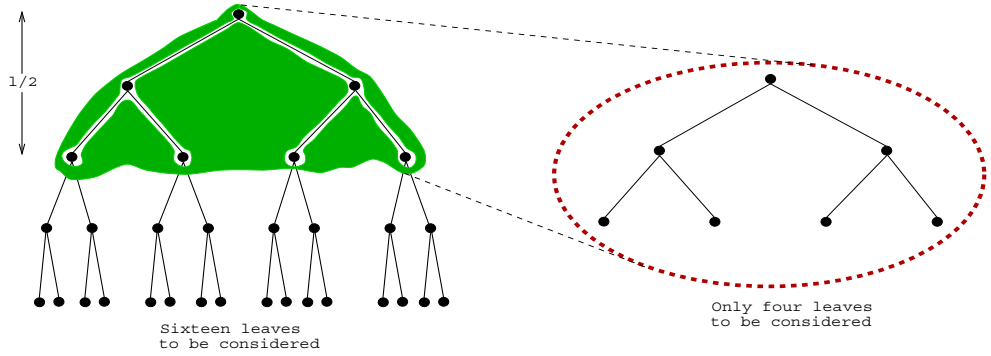


Fig. 2. Search space reduction (Second view)

to optimize, we need to find $x^* \in X$ such that $F(x^*)$ is maximized (or minimized).

To solve the above optimization problem, one may quantize the search space into equal partitions. The size of these partitions is related to the solution precision, i.e., the smaller the partition size, the finer the precision.

Definition 1 (Basic partitioning:) Given a search space of n-dimension, it contains V^n number of equal partitions, where V equals 2^l; each of them is of n-dimension.

By creating this sample space with V^n partitions, the optimization problem can then be modeled as a searching and approximation problem with V^n number of choices in n-dimensional space. Without loss of generality, we consider the one-dimensional case first.

Definition 2 (Binary number labeling scheme:) Denoting S as the set of all binary strings of length l in the form of $b_{l-1}\, b_{l-2} \cdots b_0$, where $b_i \in \{0, 1\}$, we can label the partitions of the sample space of the one-dimensional function by assigning consecutive binary strings from 0 to $V - 1$ to consecutive partitions as illustrated in Fig. 3.

For instance, if l equals to 3, the partitions are represented sequentially as 000, 001, 010, . . ., 110, and 111 in an increasing x direction. Based on this labeling scheme, we noticed that the one-dimensional search space is not only divided into V partitions, but also a hierarchy of partitions with each bit demarcating the partition inherited from the immediate more-significant bit into two halves (see Fig. 4). (The digits at the upper part of Fig. 3 show the partition hierarchy formed by the different significant bits). The top layer (the most significant bit) consists of two bit-values which represent the right and the left half of the whole sample space. The second layer consists of four bit-values representing the four partitions divided from the two in the previous layer. Partitioning in this way allows us to treat each partition as a sequence of bits so that finding an optimal partition can be done by optimizing each bit.

For functions of n-dimension, we apply the same labeling scheme to each of the variables in x, and hence n number of such separate binary hierarchies. The optimization problem would then become n simultaneous series of l selections.

To locate the optimal solution, we adopt the probabilistic search. In this search, we give scores to the states of each bit b_i. Since we are considering a binary system, two scores

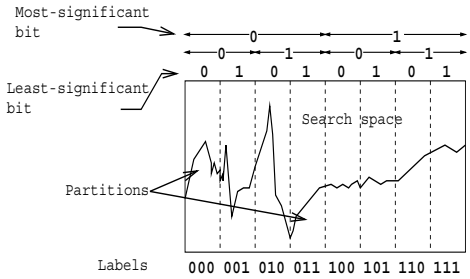


Fig. 3. Labeling of partitions: Partitioning

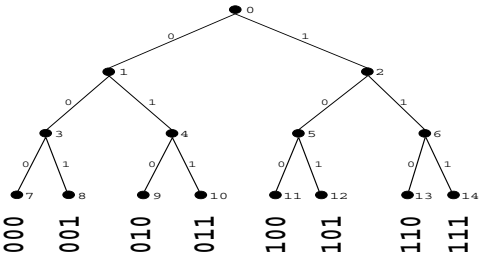


Fig. 4. Labeling of partitions: (b) Formation of hierarchy

are assigned, one to each state, indicating how well the states perform in that bit position in the past.

We now model our problem as follows.

The original problem is to find $x^* \in X$ where $X \subseteq \mathbb{R}^n$ such that

$$\forall x \in X \bullet \begin{cases} F(x^*) \geq F(x) & \text{if Maximization,} \\ F(x^*) \leq F(x) & \text{if Minimization.} \end{cases} \tag{2}$$

After the transformation, it becomes finding an optimal vector of binary strings $s^* \in S^n$ to where x^* belongs probabilistically:

$$\begin{aligned} &\max\ Prob(\text{select } s^*) \\ &= \max \prod_{m=0}^{n-1} Prob(\text{select } s^*_m) \\ &= \max \prod_{m=0}^{n-1} \prod_{i=l-1}^{0} Prob(\text{select } b^*_{m,i}) \end{aligned} \tag{3}$$

where $s^*_m \in S$ is the m-th component in vector s^* and $b^*_{m,i}$ is the i-th significant bit of binary string s^*_m.

It can then be re-formulated as finding $b^*_{m,i}$ such that for $0 \leq m < n$ and $0 \leq i < l$,

$$b^*_{m,i} = \arg\max_k \{\ a_{m,k} : k = 2(l\text{–}1\text{–}i) + b_{m,i}\ \}. \tag{4}$$

III. The model

To solve the problem formulated in the last section, we present in this section an iterative algorithm based on an information processing cycle characterized by a population of homogeneous searching agents and a searching environment. We will describe the model progressively from: (1) the basic pBHS (*probabilistic binary hierarchical search*) to (2) pBHS with cooperation (*pcBHS*) and finally (3) pcBHS with competition (*pccBHS*).

A. Local searching agents

Each agent is designed to generate in each time step n number of binary strings through n sequences of *bit-value selection* probabilistically. We treat the set of scores $a_{m,k}(t) \in [0.0, 1.0]$ at time t stated in Eq. (4) as our global information accumulated up to time t. For each function variable x_m, we define a vector

$$A_m(t) = [\ a_{m,0}(t)\ \ a_{m,1}(t)\ \ a_{m,2}(t)\ \ \cdots\ \ a_{m,2l-1}(t)\ ] \tag{5}$$

composed of $2l$ number of $a_{m,k}(t)$ (two consecutive $a_{m,k}(t)$ for one bit in binary string of length l). For an n-dimensional problem, the whole set of scores would be

$$A(t) = [\ A_0(t)\ \ A_1(t)\ \ \cdots\ \ A_{n-1}(t)\ ]^T. \tag{6}$$

In order to make the selection possible, a correspondence is drawn between $A_m(t)$ and our binary string s_m. Every non-overlapping pair of two consecutive $a_{m,k}(t)$ is used to represent a single bit. For instance, elements $a_{m,0}(t)$ and $a_{m,1}(t)$ correspond to the most-significant bit b_{l-1}, $a_{m,2}(t)$ and $a_{m,3}(t)$ correspond to the second most-significant bit b_{l-2} and so on. For each such pair of elements, we dedicate the former one as the score for $b_i = 0$ and the later one as the score for $b_i = 1$. For instance, $a_{m,0}(t)$ is the score of 0 in bit b_{l-1} and $a_{m,1}$ is the score of 1 in bit b_{l-1}. Fig. 5 shows the correspondence of a binary string and $A_m(t)$. In fact, it is not necessarily that $A_m(t)$ and the correspondence be defined as above. Different applications may have different definitions.

Definition 3 (Bit-value selection probability:) The probability of selecting a bit-value at the i-th bit $b_{m,i}$ of the m-th string s_m is defined as follows:

$$Prob\ (b_{m,i} = \kappa)\ =\ \begin{cases} a_{m,k}(t), & \kappa = 0, \\ 1 - a_{m,k}(t), & \kappa = 1. \end{cases} \tag{7}$$

As shown in Eq. (7), the selection of bit-value depends in a straightforward way on the respective global information complying with Eq. (3) and Eq. (4). The larger the $a_{m,k}$ value, the higher the chance the corresponding bit-value is selected.

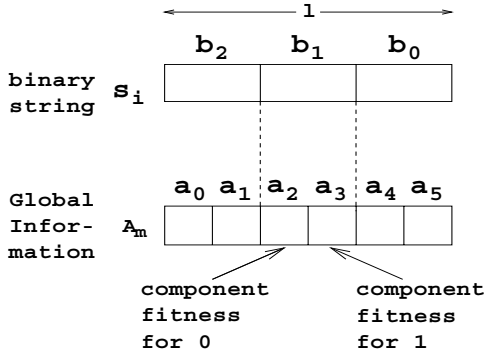


Fig. 5. Correspondence of binary string and the retained component fitness list

After generating the binary strings $s_m, 0 \leq m < n$ for all function variables x_m, we have an n-dimensional partition picked out. Since the ultimate goal is to optimize the original function stated in Eq. (2), we need a function value x from the partition for evaluation. The function value x for the partition is chosen according to:

$$x_m = \frac{s_m}{V}(x^u_m - x^l_m) + x^u_m, \tag{8}$$

i.e., the minimum x in the region s_m. Now we have a means of evaluating the partition by evaluating the representative instead.

B. Global environment

Given a reliable global information A^*, the searching agents described in the above section may be able to find s^* with probability approaching 1.0 fulfilling Eq. (3), i.e., $Prob$(select s^*) ≈ 1. The question is how to make A^* reliable? We approach this question as follows.

Every binary string generated will be evaluated to give a function value $F(x)$, which is the raw fitness of the binary string vector. Assuming that the good performance of a binary string vector is contributed by the underlying components of each constituting binary strings, we assign the raw fitness of the binary string vector to the constituting components. The previously defined correspondence between A_m and a binary string basically treats each bit as a single constituting component. Then, two vectors of length l for the raw fitness values of both states gained by a binary string are defined. We denote $u_m = [\ u_{m,0}\ u_{m,1}\ \ldots\ u_{m,l-1}\ ]$ as a vector indicating the raw fitness of the bits with bit-values equal to 0 for s_m and $w_m = [\ w_{m,0}\ w_{m,1}\ \ldots\ w_{m,l-1}\ ]$ as the vector indicating the raw fitness of bits with bit-values equal to 1. Fitness assignment to the states of each component is as follows:

For the m-th binary string s_m of the solution x, and $0 \leq i < l$,

$$\begin{cases} u_{m,i} = F(x)\ \text{ and }\ w_{m,i} = 0 & \text{if } b_{m,l-1-i} = 0, \\ u_{m,i} = 0\ \text{ and }\ w_{m,i} = F(x) & \text{if } b_{m,l-1-i} = 1. \end{cases} \tag{9}$$

It is obvious that a single sample is not reliable enough in terms of getting the global view. Hence, we distribute a population of searching agents trying different partitions

simultaneously. Their raw fitness values are added together forming another quantity called *component fitness*. The more partitions are tried, the more reliable the component fitness values are.

In order to maintain sufficient convergence power when the raw fitness of the samples are getting closer on flat landscape and at the sharp peak, the raw fitness $F(x)$ used in Eq. (9) must be scaled:

$$f_j = \frac{F(x_j) - F^{min}}{F^{max} - F^{min}} \tag{10}$$

where $F^{max} = \max_{0 \le i < N} F(x_i)$, and $F^{min} = \min_{0 \le i < N} F(x_i)$.

Definition 4 (Component fitness:) For a population of size N, we have two sets of N raw fitness vectors u_m and w_m. Summation of all the same components of the N vectors of the respective sets gives the component fitness for the respective states. Denoting $u_{j,m,i}$ and $w_{j,m,i}$ as the raw fitness values for states 0 and 1 in the $(l{-}1{-}i)$-th bit gained from evaluating the j-th binary string in the population for variable x_m respectively, the component fitness values for both states of the $(l-1-i)$-th bit $b_{m,j,(l-1-i)}$ resulted from the population are:

$$U_{m,i} = \frac{\sum_{j=0}^{N-1} u_{m,j,i}}{|\Omega_{m,i,0}|}, \qquad W_{m,i} = \frac{\sum_{j=0}^{N-1} w_{m,j,i}}{|\Omega_{m,i,1}|}. \tag{11}$$

where $\Omega_{m,i,\kappa} = \{j \in \{0,1,\ldots,N-1\} : b_{m,j,l-1-i} = \kappa\}$ and $\kappa = \{0,1\}$.

While U_m and W_m are vectors with l number of vector components:

$$U_m = [\, U_{m,0}\ U_{m,1}\ \cdots\ U_{m,l-1}], \tag{12}$$

$$W_m = [\, W_{m,0}\ W_{m,1}\ \cdots\ W_{m,l-1}]. \tag{13}$$

Vectors U_m and W_m are normalized such that $U_{m,i} + W_{m,i} = 1$, $0 \le i < l$.

Putting U_m and W_m together, we obtain a vector of combined component fitness with the same structure as A_m:

$$H_m = [\, U_{m,0}\ W_{m,0}\ U_{m,1}\ W_{m,1} \cdots U_{m,l-1}\ W_{m,l-1}\ ]. \tag{14}$$

Using this current component fitness values to make decision, the searching agents should be able to produce better binary strings, as they now could rely on an immediate past searching experience. Continuously using the newly produced component fitness means forgetting the past searching experience except the immediate one. Instead of forgetting completely the past, we retain all the past information. The past component fitness values for the m-th function variable are retained as follows:

Definition 5 (Accumulation of past searching experience:) Denote $h_{m,k}(t-1)$ as the k-th component of H_m at time $t-1$, $0 \le i < l$,

$$\begin{aligned} a_{m,k}(t) \;=\; & \beta_{m,i}(t-1) \cdot a_{m,k}(t-1) \;+ \\ & (1-\beta_{m,i}(t-1)) \cdot h_{m,k}(t-1) \end{aligned} \tag{15}$$

where $k = 2(l-1-i)$ for state 0, and $k = 2(l-1-i)+1$ for state 1.

In practical application, we keep every antagonistic pair inside $A_m(t)$ normalized: $a_{m,k}(t) + a_{m,k+1}(t) = 1$.

The newly introduced quantity $\beta_{m,i}(t)$ is called *remembrance*. It determines the fraction of the past collected information $a_{m,k}(t{-}1)$ to be retained in the generation t. It is defined in such a way that different bits can have different remembrance values. There are two reasons why different bits should have different remembrances:

1 Intuitively, the more significant bits controlling larger common partitions should have more reliable information collected than the less significant bits controlling smaller shattered partitions given same number of samples tried. Losing more past information to accommodate for the new one at the more significant bits to increase the speed of convergence becomes plausible. Hence, the more significant the bit, the smaller the remembrance should be.

2 The hierarchical structure has the advantage on search space reduction . Briefly speaking, reduction occurs at a level of the hierarchy when sufficient information is collected in all of the upper levels. For instance, if the most-significant bit $b_{m,l-1}$ collected enough information, either $a_{m,0}(t)$ or $a_{m,1}(t)$ will have very high value. Say if $a_{m,1}(t)$ has a higher value, it is highly probable that the right partition contains the global optimum. Searching should then be concentrated on that region. In other words, the size of the search space is reduced by half, suggesting a smaller remembrance value be used to speed up the convergence.

Therefore, we devised an *adaptive remembrance scheme* to speedup the convergence.

Definition 6 (Adaptive remembrance scheme:) Let τ denote a convergence threshold. Any score $a_{m,k}(t)$ that is greater than τ as said to be converged. Let β denote the minimum allowed remembrance. Suppose the r-th bit $b_{m,r}$ of binary string s_m for function variable x_m is the first bit encountered starting from the most significant side that satisfies the following:

$$|\, 0.5 - a_{m,2(l-1-r)}(t)\,| \;>\; \tau \;\vee\; |\, 0.5 - a_{m,2(l-r)}(t)\,| \;<\; \tau. \tag{16}$$

Then the remembrance value used in each bit of s_m is set according to:

$$\beta_{m,i}(t) = \begin{cases} \beta(t) & l > i \ge r, \\ \dfrac{r - i + \beta(t)}{r - i + 1} & r > i \ge 0. \end{cases} \tag{17}$$

This scheme, basically, keeps the remembrance for the converged bits (b_{l-1} to b_{r+1}) constant at β, while interpolates the rest from β to $(r+\beta)/(r+1)$. Fig. 6 shows the remembrance settings at difference stages of convergence.

C. pcBHS: pBHS with cooperation

High-dimensionality poses a great challenge to all optimization algorithms, in particular searching algorithms, because of the exponential scale-up of the size of the search

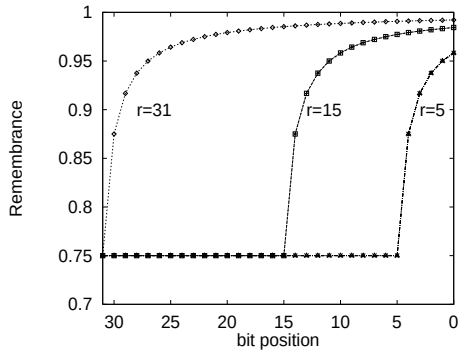


Fig. 6. Remembrance for different bits under different stages of convergence

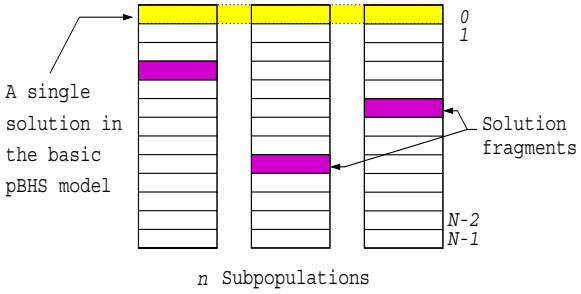


Fig. 7. Decoupling

space. To handle the high-dimensional problem, we introduce *pcBHS* (Probabilistic Cooperative Binary Hierarchical Search) incorporating *cooperation*. Besides the cooperation among searching agents in the basic model described before, the model incorporates the cooperation among dimensions of the problem [15].

In the basic pBHS model, a population is defined as a group of sample points consisting of samples from all subspaces. In the following, sample point is referred to as a *complete solution* while a sample from a sub-space is referred to as a *solution fragment*. For instance, a complete solution for a three-dimensional function $F(x)$ is a vector $[x_0, x_1, x_2]$ which consists of three solution fragments: x_0, x_1 and x_2.

In pcBHS, decoupling is taken such that each sub-space exists as its own and a single population in the pBHS model becomes n subpopulations here. The size of each subpopulation is still kept at N in order to maintain the same varieties of solution fragments as in pBHS. The situation is illustrated in Fig. 7. The shaded region enclosed by two dotted lines indicates a single complete solution in the basic pBHS model. In pcBHS, all solution fragments in a subpopulation are not tied with any solution fragment in other subpopulations. What we have are n sets of N solution fragments.

Before describing how to combine solution fragments, the issue on fitness measurement should be addressed first. The ordinary fitness measurement as used in pBHS becomes inappropriate in the cooperative model. Raw fitness is meaningful only when a single complete solution exists. After decoupling, fragments representing the problem sub-spaces are created. Their fitness values are undefined. *Cooperative fitness* as defined in [15] is employed to evaluate the solution fragments. Given n arbitrary solution fragments $\{x_0, x_1, \cdots, x_{n-1}\}$ from each subpopulation, each of their raw cooperative fitness equals $F(x)$ where $x = [x_0 \ x_1 \ \cdots \ x_{n-1}]$. Suppose that the same set of solution fragments are given with x_{n-1} replaced by x'_{n-1}, their raw cooperative fitness become $F(x')$ where $x' = [x_0 \ x_1 \ \cdots \ x'_{n-1}]$.

The cooperative pcBHS model differs from the basic pBHS model in three aspects - fitness evaluation, fitness scaling, and elitism:

1 *Fitness measurement and fitness scaling*
As discussed in the previous sections, raw fitness is replaced by cooperative fitness owing to the decoupling of solution fragments. Suppose that there is a *global elite* $x^e = [\, x_0^e \ x_1^e \ \cdots \ x_{n-1}^e \,]$, the cooperative fitness of each solution fragment $x_{m,j}$ in each subpopulation m is defined as $cF(x_{m,j}, x^e)$. Function cF is simply the objective function F applied to a complete solution formed by replacing the m-th element in x^e by $x_{m,j}$. Algorithm 2 shows how it is implemented. Under this scheme, there are $n \times N$ number of complete solutions centered around x^e formed. These raw cooperative fitness cF of each solution fragments are scaled within their subpopulations only. Denoting cf as the *scaled cooperative fitness*, the cf of the j-th individual in the m subpopulation is:

$$cf(x_{m,j}, x^e) = \frac{cF(x_{m,j}, x^e) - cF_m^{min}}{cF_m^{max} - cF_m^{min}} \tag{18}$$

$cF_m^{max} = \max\{\ F(x^e),\ \max_{0 \le j \le N-1} cF(x_{m,j}, x^e)\}$ and $cF_m^{min} = \min\{\ F(x^e),\ \min_{0 \le j \le N-1} cF(x_{m,j}, x^e)\}$. In Eq. 9 and , it is f_j that is fed back into the system. We now use the scaled cooperative fitness cf. Given a binary string s_m of the m-th dimension, and $0 \le i < l$, the component fitness for the $(l-1-i)$-th bit is determined as follows:

$$\begin{cases} u_{m,i} = cf(x_{m,j}, x^e) \ \text{ and} & \\ w_{m,i} = 0 & \text{if } b_{m,l-1-i} = 0, \\ u_{m,i} = 0 \ \text{ and} & \\ w_{m,i} = cf(x_{m,j}, x^e) & \text{if } b_{m,l-1-i} = 1. \end{cases} \tag{19}$$

2 *Elitism*
Under this model, the elitist strategy used in the basic pBHS model have to be modified. Since each subpopulation is individually responsible for a single unique dimension, elitism is applied separately to each subpopulation (see Algorithm 2) in each generation producing a set of new local elites $\{x_0'^e, x_1'^e, \cdots, x_{n-1}'^e\}$. The new global elite x''^e is selected from the either one of the following:

- **No-change**
  The existing global elite x^e with raw fitness $F(x^e)$.
- **Local**
  Any one of the local elite $x_m'^e$ in cooperation with the existing global elite x^e.

- **Random**
  The best of n complete solutions formed be cooperating randomly picked solution fragment x_m^r with the global elite x^e. The cooperative fitness of a complete solution formed by cooperating a randomly selected solution fragment in the m-th subpopulation with the global elite is denoted as $cf(x_m^r, x^e)$.
- **Multiple**
  A complete solution formed by combining the existing global elite x^e and those $x_m'^e$ whose cf is greater than $F(x^e)$:

$$\forall m, 0 \leq m < n \ s.t. \ cf(x_m'^e, x^e) \geq F(x^e) \qquad (20)$$

  Suppose that the set of local elites satisfies this criterion is $\psi = \{x_1'^e, x_3'^e\}$, the complete solution would be $\{x_0^e, x_1'^e, x_2^e, x_3'^e, \cdots, x_{n-1}^e\}$ and its cooperative fitness is denoted as $cf(\psi, x^e)$.

The global elite is replaced by any one of them with the maximum cooperative fitness:

$$F(x''^e) \geq \max \{F(x^e),\ cf(x_m'^e, x^e),\ cf(x_m'^e, x^r),\ cf(\psi, x^e)\} \qquad (21)$$

The last two choices (*random* and *multiple*) are used to lower the greediness of the simple *no-change* plus *local* scheme of CCGA-1 as illustrated in [15]. Although the replacement scheme is still a winner-take-all strategy, the *random* scheme may introduce new solution fragments which may lead to a new and possibly optimal path, while the *multiple* scheme allows multiple-subspace movement in one single step.

D. pccBHS: The cooperative-competitive model

The two basic design goals of the model described so far are: (1) distribution of a population of individuals who search cooperatively for a single global optimum, and (2) assumption of no (or minimal) *aprior* information about the problem to be solved. However, this design would make both algorithms easily be deceived, because (1) when the landscape of a problem has multiple number of similar basin of attractions, and (2) the landscape of a problem to be solved provides misleading information. In this section, we extend both models to cater for the problem by introducing *redundancy* and *competition*. The enhanced model, *pccBHS (Probabilistic cooperative-competitive binary hierarchical search)* shares similarities with the existing sharing mechanisms. The strength of the model over sharing mechanisms is the avoidance of *niche radius*. Niche radius inherently limits the niching mechanisms to be applied to problems that requires niches to be located at different resolution levels simultaneously. The drawback of the pccBHS model, is that the number of niches to be occupied is bound by a prescribed number.

The main structural characteristic of the pccBHS model is the division of a whole population into a number of subpopulation groups (*subgroups*) to provide redundancy. They are allowed to gather their own set of global information. The one with the highest fitness is considered as the global solution. In the course of searching, these subgroups are allowed to compete with each other for exclusive occupancy of territories. The aim of the competition is to force them to search different areas by separating them in the n-dimensional space. The competition is achieved by generating a repulsive force when two subgroups come together in the n-dimensional space. The closer the two subgroups, the greater the repulsive force. Once they are separated, the force disappears. Effectively, pccBHS re-models the function landscape in such a way that the deceptive attractor is made hidden by another subgroup as shown in Fig. 8. What each subgroup faces is a unimodal or non-deceptive landscape.

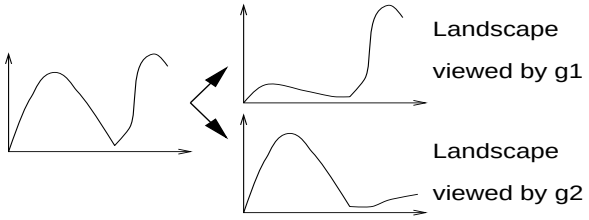


Fig. 8. Re-modeling of function landscape

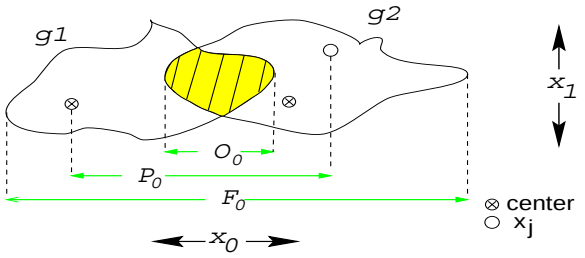


Fig. 9. Overlapping of two subgroups

Suppose there are G number of subgroups g_r, $0 \leq r < G$. Each of these subgroups is the same as a single population in pcBHS model. The only difference is the size of each subgroup which is equal to $\lfloor N/G \rfloor$. The cooperative pcBHS is a special case of the pccBHS model that $G = 1$.

For the sake of clarity, the model is presented using two subgroups only. Given two subgroups $g1$ and $g2$, we first check if all of their dimensions are overlapped, since two subgroups are said to be overlapped only when they are overlapped in *all* dimensions. Two metrics that are required to calculate the repulsive force are (i) *degree of overlapping* and (ii) *proximity*.

For each dimension m, we measure the distance F_m which is the maximum distance of all pairs of binary strings from the two subgroups in consideration as shown in Fig. 9. Denote $g1_m^{min}$ and $g1_m^{max}$ as the minimum and the maximum of the m-th dimension solution fragments of $g1$ respectively, $g2_m^{min}$ and $g2_m^{max}$ as the minimum and the maximum of m-th dimension solution fragments of $g2_m$ respectively. The distance F_m is defined as,

$$F_m = \max\{g1_m^{max}, g2_m^{max}\} - \min\{g1_m^{min}, g2_m^{min}\}. \qquad (22)$$

Minimum possible value of F_m is 0 when all of the m-th dimension solution fragments in $g1$ and $g2$ are identical, i.e.,

both $g1$ and $g2$ are merged together in the m-th dimension. Maximum possible value of F_m is $x_m^u - x_m^l$, i.e., the full range of the m-th dimension of x. We also measure the distance O_m of the region where they overlap (the shaded region in Fig. 9). Overlapping distance O_m equals 0 when $g1_m^{max} < g2_m^{min}$ or $g2_m^{max} < g1_m^{min}$.

Definition 7 (Degree of overlapping) Degree of overlapping $D_m(g1, g2)$ between the same dimension m of $g1$ and $g2$ is defined as:

$$D_m(g1, g2) \;=\; \frac{O_m}{F_m} \tag{23}$$

There are three distinct situations:

1 **Disjoint**
$g1$ and $g2$ are totally separated.

$$D_m(g1, g2) \;=\; 0 \quad \text{when} \quad O_m \;=\; 0 \tag{24}$$

2 **Enclosure**
$g1$ is totally enclosed by $g2$ or vice versa.

$$D_m(g1, g2) \quad = \quad 1$$

when

$$\left\{ \begin{array}{l} g1_m^{min} < g2_m^{min} \wedge g1_m^{max} > g2_m^{max}, \quad \text{or} \\ g2_m^{min} < g1_m^{min} \wedge g2_m^{max} > g1_m^{max} \end{array} \right. \tag{25}$$

3 **Overlapping**
$g1$ and $g2$ are overlapping.

$$0 < D_m(g1_m, g2_m) < 1$$

when

$$\left\{ \begin{array}{l} g1_m^{min} < g2_m^{min} < g1_m^{max} < g2_m^{max}, \quad \text{or} \\ g2_m^{min} < g1_m^{min} < g2_m^{max} < g1_m^{max} \end{array} \right. \tag{26}$$

The quantity D_m serves two purposes: (1) decides whether the repulsion exists and (2) determines the level of force required if repulsion exists. However, it does not reflect the fact that individuals farther away from the overlapping subgroup should receive less repulsive force. Thus, a *proximity value* is then introduced.

Definition 8 (Proximity) For the m-th dimension, proximity value $P_m(g1, x_{m,j})$ is defined over the j-th individual $x_{m,j}$ of $g2$ and its overlapping neighbor subgroup $g1$ as the normalized distance between $x_{m,j}$ and the center of $g1$ (see Fig. 9):

$$P_m(g1, x_{m,j}) \;=\; \frac{|x_{m,j} - x_m^e|}{F_m}, \tag{27}$$

where x_m^e is the m-th solution fragment of the elite in $g1$ and $P_m(g, x_{m,j}) \in [0, 1]$. Being the driving force within a subgroup, the subgroup elite is considered as the center.

Definition 9 (Repulsive force) Repulsive force $R_m \in [0, 1]$ experienced by the binary string $x_{m,j}$ of $g2$ due to the overlapping with $g1$ is defined as:

$$R_m(g1, x_{m,j}) \;=\; D_m(g1, g2) \times (1 - P_m(g1, x_{m,j})). \tag{28}$$

Finally, another quantity *interaction fitness* $I_m(x_{m,j}, x^e)$ is defined to indicate how well an individual performs in the competition:

Definition 10 (Interaction fitness) For the mth dimension,

$$I_m(x_{m,j}, x^e) \;=\; \frac{cf(x_{m,j}, x^e)}{R_m(g1, x_{m,j})}, \tag{29}$$

where $cf(x_{m,j}, x^e)$ is the cooperative fitness of $x_{m,j}$ (see Eq. 19).
Instead of feeding back cf into the system, I_m should be used:

$$\left\{ \begin{array}{lcll} u_{m,i} & = & I_m(x_{m,j}, x^e) \;\text{ and} & \\ w_{m,i} & = & 0 & \text{if } b_{m,l-1-i} = 0, \\ u_{m,i} & = & 0 \;\text{ and} & \\ w_{m,i} & = & I_m(x_{m,j}, x^e) & \text{if } b_{m,l-1-i} = 1. \end{array} \right. \tag{30}$$

IV. Experiments

A. Basic

In this experiment, we tried four problems listed in Table I. They are commonly used in testing global optimization algorithms. We tried:

for R2, GP2 and H3
$N = \{10, 50, 90\}$
$\beta = \{0.80, 0.85, 0.90, 0.95\}$
$\mu = 1$, $\tau = 0.4$
$l = 16$
for S1
$N = \{10, 20, 30\}$
$\beta = \{0.93, 0.94, 0.95, 0.96\}$
$\mu = 1$, $\tau = 0.4$
$l = 16$

Tables II and III show the percentage of trials and average iterations required to get the global optima of the respective functions. We obtained 100% success rate (reaching the prescribed f^+) on S1 and GP2 functions under majority of our experimental conditions. Comparing the amount of computations required, our algorithm is in general less expensive than those listed in Table IV-A. While for R2 and H3 functions, the low success rates for some test conditions can be explained by their rugged landscapes and the raise in dimensionality.

B. High-dimensionality

In this experiment, several problems with problem size up to 100 dimensions are tried. These problems are commonly used in testing global optimization algorithms. Each family of problems possesses characteristics quite different from each other. A brief summary of the test problems used are listed in Table IV.

The results listed in Table V shows that the performance of our algorithm is comparable with the existing advanced techniques such as Breeder GA (BGA) [14] and Evolutionary Algorithm with Soft genetic operators (EASY) [21], [22]. Both variants of GAs are said to be highly effective

TABLE I

BENCHMARK TEST FOR PBHS: TEST PROBLEMS

Problems	n	f^*	x^*	#eval†
S1	1	14.5926520	0.6858609	♭
R2	2	2.0000	[0 0]	♭
GP2	2	-3.00001	[0 -1]	492 [19]
H3	3	3.86	[0.4047 0.8828 0.8732]	1,014 [3]

S1 - Shekel, R2 - Rastrigin, GP2 - Goldstein-price, H3 - Hartman3

†: Average number of function evaluations

♯: Number of iterations

♭: See the entry GA in the table below.

Algorithms	Function evaluations			
	S1	R2	GP2	H3
MS	-	1176	4400	2500
CRS	-	-	2500	2400
SA	-	-	563	1459
SAsde	-	-	5439	3416
HGA	-	-	146	191
ARS	-	-	492	-
NP	-	-	936	1014
PE	-	-	200♯	-
GA⋆	1185.9	3676	1644.6	972

♯: Number of iterations

⋆: Experiments carried by ourselves.

See the following table for the experimental conditions.

-: No results reported.

MS - Multistart [17], [20]

CRS - Controlled random search [16]

SA - Simulated annealing [4]

SAsde - SA based on stochastic differential equations [1]

ASA - Adaptive simulated annealing [10]

HGA - Hybrid genetic algorithm [9]

APRS - Adaptive partitioned random search [19]

NP - New Price's algorithm [3]

PE - Perttunen's method [18]

GA - Genetic Algorithm

Experimental conditions for the GA test				
	S1	R2	GP2	H3
Population size	30	50	30	90
Mutation probability P_μ	$1/nl$, $l = 16$			
Crossover probability P_χ	1.0 (Two-point)			
Fitness	Fitness scaling			
Selection	2-Tournament			
Replacement	Proportional			
Success rate	100%			

TABLE II

PERCENTAGE OF TRIALS GETTING THE GLOBAL OPTIMUM FOR S1, R2, GP2, AND H3

β	S1			β	R2		
	10	**20**	**30**		**10**	**50**	**90**
0.96	100	100	100	**0.95**	84	100	100
0.95	99	100	100	**0.90**	69	100	100
0.84	100	100	100	**0.85**	56	97	100
0.93	98	100	100	**0.80**	44	94	98
β	GP2			β	H3		
	10	**50**	**90**		**10**	**50**	**90**
0.95	100	100	100	**0.95**	100	98	100
0.90	100	100	100	**0.90**	95	96	98
0.85	99	100	100	**0.85**	83	92	98
0.80	99	100	100	**0.80**	74	90	95

TABLE III

AVERAGE NUMBER OF FUNCTION EVALUATIONS REQUIRED TO REACH THE GLOBAL OPTIMUM FOR S1, R2, GP2, AND H3

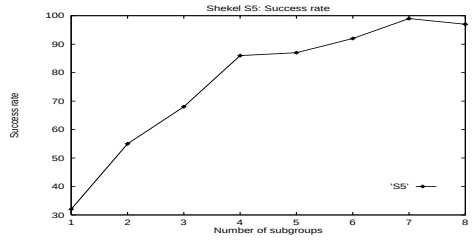
β	S1			β	R2		
	10	**20**	**30**		**10**	**50**	**90**
0.96	1,236	1,908	2,510	**0.95**	1,260	4,565	6,993
0.95	1,001	1,600	2,174	**0.90**	704	2,695	4,023
0.94	915	1,444	2,039	**0.85**	514	1,925	3,006
0.93	825	1,331	1,800	**0.80**	398	1,530	2,412
β	GP2			β	H3		
	10	**50**	**90**		**10**	**50**	**90**
0.95	951	3,100	4,653	**0.95**	709	1,181	1,547
0.90	590	1,890	3,042	**0.90**	434	724	959
0.85	405	2,420	2,331	**0.85**	330	555	766
0.80	329	1,125	1,872	**0.80**	274	458	563

yet as a general model for evolutionary algorithms. However, the performance of the algorithm on the Shekel family is comparatively poorer than those from the existing techniques. We attribute this poor performance as their *golf-hole*-like landscapes. The figure shows that S5 has 5 prominent optima resting on a plateau, all of which has similar basin of attraction. Landscapes of this kind provides no useful information for guidance. The reason why the successful rate drops from S5 to S10 is due to the raise in the number of prominent optima. This problem will be catered in the next test (see section IV-C).

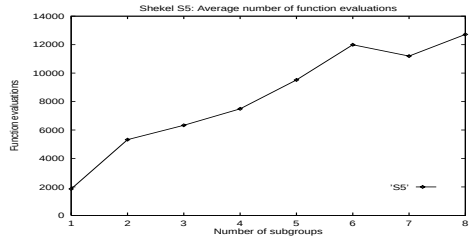
C. Deception

Our algorithm is demonstrated to be able to solve some of the GA-deceptive problems such as Goldberg's bipolar deceptive function (see [13] for details).

Besides handling GA deceptive problems, our algorithm can also handle problems with golf-hole-like landscape such as those in the Shekel family—S5, S7 and S10. The landscapes of these three functions share a commonality: several sub-optima with similar basin of attractors sitting on a large flat plateau. The sizes of these attractors are so similar that no useful information can be obtained about

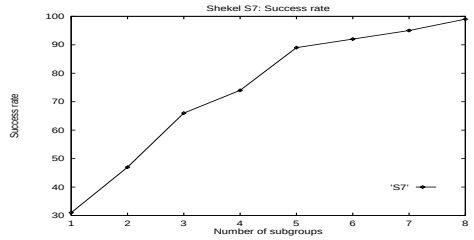


(a)

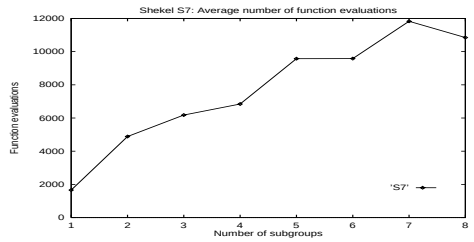


(b)

Fig. 10. Shekel family S5: (a) Graph showing the effect of different number of subgroups on the percentage of runs getting the global optimum. (b) Graph showing the computational expenses on using different number of subgroups.



(a)



(b)

Fig. 11. Shekel family S7: (a) Graph showing the effect of different number of subgroups on the percentage of runs getting the global optimum. (b) Graph showing the computational expenses on using different number of subgroups.

TABLE IV
BENCHMARK: TEST PROBLEMS

Problems♯	n	f^+	#eval	**Ref.**
S5 - Shekel	4	9.9	5,403	⋆
S7 - Shekel	4	9.9	5,386	⋆
S10 - Shekel	4	9.9	5,862	⋆
H3 - Hartman	3	3.86	1,014	⋆
A30 - Ackley	30	0.001	13,997/19,420	†
A100 - Ackley	100	0.001	57,628/53,860	†
R20 - Rastrigin	20	0.9	6,098/3,608	†
R100 - Rastrigin	100	0.9	45,118/25,040	†

♯: see Appendix ?? for the description of the problems
⋆: A clustering technique: New Price's algorithm [3]
†: EA with soft genetic operators/Breeder GA [EASY/BGA] [21]
f^+: Function values at which the algorithms stop.
#eval: Number of function evaluations.

TABLE V
BENCHMARK TEST: RESULTS

Problems	f^+ attained	#eval	%‡	**Conditions***
S5	10.004523	12,476.40	34%	β=0.97 N=50
S7	10.100106	13,168.60	29%	β=0.97 N=50
S10	10.126623	13,909.30	14%	β=0.97 N=50
H3	3.861696	755.80	100%	β=0.90 N=30
A30	-0.00078	18,679.68	100%	β=0.40 N=40
A100	-0.00074	58,216.17	90%	β=0.35 N=40
R20	-0.48987	5,413.22	100%	β=0.45 N=40
R100	-0.54718	45,194.86	100%	β=0.45 N=40

‡: Percentage of runs reaching f^+ stated in Table IV.
∗: 100 independent consecutive runs, $l = 16$.

the location of the global solution.

The performance of our algorithm with $G = 1$ is shown in Table V, while the experimental conditions are listed in Table VI. The purpose of this experiment is to illustrate the usefulness of redundancy with competition in improving the pcBHS model. Moreover, we examined the effect of different number of subgroups on the algorithm performance for these problems. The result is listed in Table VII. It shows clearly that by increasing the number of subgroup, the success rate can be increased.

V. CONCLUSION

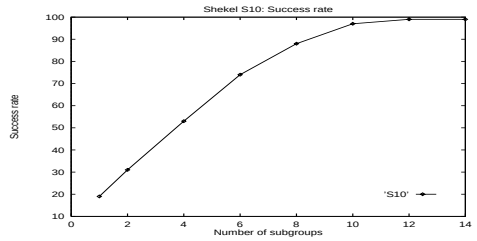
In this paper, a new iterative stochastic searching algorithm called *probabilistic cooperative-competitive binary hierarchical search (pccBHS)* is proposed for global optimization. It is proposed to complement for the insufficiency of existing algorithms by providing resolution controls, smoothing of search space and introducing memory into stochastic search. Two key ideas of nature has been used in the algorithm: cooperation and competition. Dealing with high-dimensional functions, we have adopted a decoupling scheme. This decoupling scheme improves the

TABLE VI

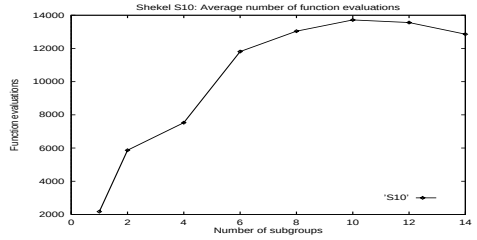
SHEKEL FAMILY: CONDITIONS

Problems	f^+	G	Conditions$^\sharp$
S5	9.9	1,2,3,4,5,6,7,8	$N = 100$
S7	9.9	1,2,3,4,5,6,7,8	$N = 100$
S10	9.9	1,2,4,6,8,10,12,14	$N = 140$

$\sharp$: 100 consecutive independent runs, $\mu = 1$, $\beta = 0.05$



(a)



(b)

Fig. 12. Shekel family S10: (a) Graph showing the effect of different number of subgroups on the percentage of runs getting the global optimum. (b) Graph showing the computational expenses on using different number of subgroups.

algorithm significantly on solving high-dimensional function.

Competition has been introduced to level off the strong exploitation effect. By competition, we mean the use of multiple number of populations to search separately while keeping a repulsive force among them. In this way, each population can search in high speed to exploit the information gathered and at the same time maintain the solution quality by searching diversely. Furthermore, this competition model has one advantage over existing techniques that is it does not need a pre-defined *radius* which inherently limits the algorithm from finding optima in different resolutions simultaneously.

Comparing the performance of ours with the existing algorithms, we have the advantage of versatility and computationally more economical.

TABLE VII

SHEKEL FAMILY: RESULTS

S5			
G	%	f^+	#eval
1	32	10.030556	1,860.6
2	55	10.023210	5,323.7
3	68	10.012363	6,331.0
4	86	10.021969	7,487.6
5	87	10.011920	9,519.9
6	92	10.013254	11,991.6
7	**99**	**10.009583**	**11,196.6**
8	97	10.013743	12,717.8

S7			
G	%	f^+	#eval
1	31	10.171804	1,659.3
2	47	10.113239	4,885.9
3	66	10.121162	6,179.5
4	74	10.113394	6,841.3
5	89	10.145635	9,574.2
6	92	10.117482	9,579.1
7	95	10.098005	11,826.0
8	**99**	**10.100353**	**10,845.1**

S10			
G	%	f^+	#eval
1	19	10.358868	2,170.7
2	31	10.189666	5,868.7
4	53	10.194831	7,526.4
6	74	10.135957	11,811.3
8	88	10.194929	13,040.2
10	97	10.169039	13,717.1
12	**99**	**10.164886**	**13,560.0**
14	99	10.173139	12,858.3

f^+: Average function value reached
#eval: Number of function evaluations
%: Success rate

APPENDIX

I. ALGORITHM*

REFERENCES

[1] F. Aluffi-Pentini, V. Parisi, and F. Zirilli. Global optimization and stochastic differential equations. *Journal of Optimization Theory and Applications*, 47:1–16, 1985.

[2] D. Beasley, D.R. Bull, and R.M. Ralph. An overview of genetic algorithms: Part 1, fundenmental. *University Computing*, 15(2):58–69, 1993.

[3] P. Brachetti, M. De Felice Ciccoli, G. Di Pillo, and S. Lucidi. A new version of the price algorithm for global optimization. *J. Global Optimization*, 10, 1997.

[4] A. Dekkers and E. Aarts. Global optimization and simulated annealing. *Mathematical programming*, 50:367–393, 1981.

[5] D.B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, 1 1994.

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

Algorithm 1 THE INFORMATION PROCESSING CYCLE

Procedure INFORMATIONPROCESSINGCYCLE
  global environment ← Empty
  While *stopping criteria are not met*
  Loop
    For each searching agent **do**
      search result ← Search(global environment)
    End For
    global environment ← Modify(collection of search result, global environment)
  End While
End Procedure

Algorithm 2 PCBHS - COOPERATIVE FITNESS ASSIGNMENT AND *local* ELITES UPDATING. This procedure assigns fitness (cooperative fitness) to all solution fragments. In each subpopulation, if the best fragment is better than the elite fragment x'^e_m, it becomes the new elite fragment. It should be noted that the current elite x^e is not changed in this procedure.

Procedure COOPEVALUATION
  For each subpopulation P_m, $0 \leq m < n$
    /* x'^e_m: Elite fragment of the m-th subpopulation */
    $x'^e_m \leftarrow x^e_m$
    For each solution fragment $x_{m,i}$ in P_m
      $x \leftarrow$ replace the m-th element of x^e by $x_{m,i}$
      $cf(x_{m,i}, x^e) \leftarrow F(x)$
      if $cf(x_{m,i}, x^e) > cf(x'^e_m, x^e)$ **then**
        $x'^e_m \leftarrow x_{m,i}$
      End if
    End for
  End for
End Procedure

[7] J.H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1975.

[8] R. Horst and P.M. Pardalos. *Handbook of Global Optimization*, page 832. Kluwer Academic Publishers, 1995.

[9] M.F. Hussain and K.S. Al-Sultan. A hybrid genetic algorithm for nonconvex function minimization. *Journal of Global Optimization*, 11:313–324, 1997.

[10] L. Ingber. Simulated annealing: Practice versus theory. *Journal of Mathematical and Computer Modeling*, 18(11):22–59, 1993.

[11] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *J. Statistical Physics*, 34(5–6):976–986, 1986.

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

[13] K.S. Leung, Terence Wong, and Irwin King. Probabilistic cooperative-competitive hierarchical modeling as a genetic operator in global optimization. In *Proc. 1998 Intl. Conf. Systems, Man, and Cybernetics (SMC'98)*, 1998.

[14] H. Mühlenbein and D Schlierkamp-Vosen. Predictive models for the breeder genetic algorithm, i. continuous parameter optimization. *Evolutionary Computation*, 1(1):25–49, 1993.

[15] M.A. Potter and K.A. De Jong. A cooperative coevolutionary approach to function optimization. In Y. Davidor, H-P Schwefel, and R. Manner, editors, *Parallel Problem Solving from Nature III*. Springer-Verlag; Berlin, Germany, 1994.

[16] W.L Price. *A controlled random search procedure for global optimization*, pages 71–84. North Holland, Amsterdam, 1978.

[17] A.H.G. Rinnooy Kan and G.T. Timmer. Stochastic methods for global optimization. *American Journal of Mathematical and Management Sciences*, 4(1), 1984.

[18] B.E. Stuckman and E.E. Easom. A comparison of bayesian/sampling global optimization techniques. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(5):1024–1032, 1992.

[19] Z.B. Tang. Adaptive partitioned random search to global optimization. *IEEE Transactions on Automatic Control*, 39(11):2235–2244, 1994.

[20] A. Törn and A. Žilinskas. *Global Optimization.* Lecture Notes in Computer Science. Springer-Varlag Berlin Heidelberg, 1989.

[21] Hans-Michael Voigt. Soft genetic operators in evolutionary algorithms. In W. Banzhaf and F.H. Eeckman, editors, *Evolution and Biocomputation.* Berlin; New York: Springer, 1995.

[22] Hans-Michael Voigt and Anheyer Thomas. Modal mutations in evolutionary algorithms. In *Proceedings of the First (1994) IEEE Conference on Evolutionary Computation (ICEC'94). IEEE World Congress on Computational Intelligence*, pages 88–92. IEEE; New York, NY, USA, 1994.