PROBABILISTIC COOPERATIVE-COMPETITIVE HIERARCHICAL MODELING AS A GENETIC OPERATOR IN GLOBAL OPTIMIZATION

Kwong-Sak Leung Terence Wong Irwin King
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Email:{ksleung,wongyb,king}@cse.cuhk.edu.hk

ABSTRACT

Existing search-based discrete global optimization methods share two characteristics: (1) searching at the highest resolution; and (2) searching without memorizing past searching information. In this paper, we firstly provide a model to cope with both. Structurally, it transforms the optimization problem into a selection problem by organizing the continuous search space into a *binary hierarchy* of partitions. Algorithmically, it is an iterative stochastic cooperative-competitive searching algorithm with memory. It worths mentioning that the competition model eliminates the requirement of the *niche radius* required in the existing niching techniques. The model is applied to (but not limited to) function optimization problems (includes high-dimensional problems) with experimental results which show that our model is promising for global optimization. Secondly, we show how pccBHS can be integrated into genetic algorithms as an operator.

1. INTRODUCTION

MOTIVATION

Global optimization approaches such as simulated annealing [1], evolutionary algorithms [2], [3] and greedy decent/ascent [4] share two characteristics: (1) they search the sample space at the highest resolution; and (2) they search without memorizing past global information. These characteristics could in some circumstances be undesirable. Motivated by these characteristics, we provide our model as a complementary approach.

We approach the optimization problem by organizing the sample space into a binary hierarchy of partitions so as to make the reduction of search space and the control of resolution possible. To deal with the absence of reliable global information, we adopt a stochastic cooperative searching algorithm. A set of searching agents are allowed to explore and collect information about the sample space autonomously. This information is then used to guide the searching agents in future explorations. Concerning about the ability to cope with high-dimensional problems, we combined the cooperative model devised by De Jong [5] and the competition model based on niching methods [6], [7], [8].

SEARCH SPACE REDUCTION AND RESOLUTION CONTROL WITH BINARY HIERARCHY

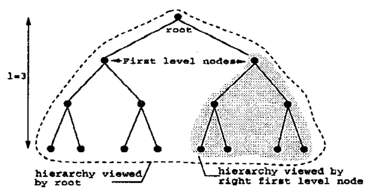

Fig. 1. Balanced binary hierarchy

Given a balanced binary hierarchy (Fig. 1) of l levels[1], there are $l+1$ number of node layers and 2^l number of leaf nodes. To locate a leaf node, we go through l number of branches starting from the root. If we need to make a decision on which 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

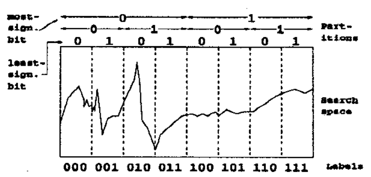

Fig. 2. Labeling of partitions

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.

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

The formation of such a hierarchy basically defines $l+1$ number of resolution levels. The levels upper in the hierarchy represent the sample space in lower resolutions and vice versa. This resolution hierarchy allows our algorithm to concentrate the searching at the lower resolution (general shape of the landscape), which is easier, locating the promising area first and to drive into the precise optimum later at the higher resolution when it is converging.

2. PROBLEM FORMULATION

In this section, we express our problem in terms of unconstrained function optimization. Given a continuous real-value function $F(x)$ to optimize, we need to find x^* such that $F(x^*)$ is the optimum. Depending on the required solution precision, we quantize the search space into V partitions. Imposing a restriction on V that it should be equal to 2^l where $l \in \mathbb{N}$, a binary number labeling scheme is then introduced to label the partitions as shown in Fig. 2. Suppose S denotes the set of binary strings s_i of length l, the partitions are labeled as s_0, s_1, ..., s_{V-1} successively. Based on this labeling scheme, we notice that the search space is not only divided into V partitions, but also a hierarchy of partitions with each bit separating the partition inherited from the immediate more-significant bit into two halves. We can then treat each partition as a sequence of bit-values for optimization. The problem becomes so simple that it accounts for just a series of l selections between 0 and 1.

To locate the optimal solution, we explore the hierarchy in a probabilistic way. To do the probabilistic search, we give the states of each bit $b_i \in \{0,1\}$ scores a_k where $k = 2(l{-}1{-}i){+}b_i$ indicating how well the states perform in that bit position in the past. Using these scores, a reasonable bit-value selection scheme (probabilistic search) becomes possible. We now restate our problem as follows:

[1]We define a 'level' as a layer of branches but not as a layer of nodes.

The original problem is to find x^*, such that

$$F(x^*) \geq F(x), \forall x \in X,\ X \subseteq \mathbb{R} \qquad (1)$$

After transformation, it becomes a problem to find probabilistically an optimal binary string $s^* \in S$ to where x^* belongs:

$$\max\ Prob(\text{select } s^*) = \max \prod_{i=l-1}^{0} Prob(\text{select } b_i^*) \qquad (2)$$

which can be re-formulated as finding b_i^* such that:

$$b_i^* = \arg\max_k \{\, a_{k^*} : k = 2(l{-}1{-}i){+}b_i \,\} \qquad (3)$$

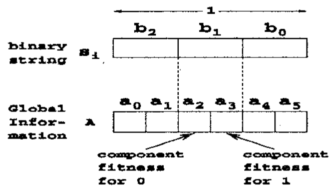

Fig. 3. Correspondence of bit-string and the retained component fitness list

3. INFORMATION PROCESSING CYCLE

To solve the problem formulated in the last section, we present in this section an algorithm based on the information processing cycle characterized by a population of homogeneous searching agents and a searching environment.

SEARCHING AGENTS - LOCAL BEHAVIOR

Each agent is designed to generate in each iteration a binary string through a sequence of bit value selection probabilistically. We treat the set of scores $a_k \in [0.0, 1.0]$ stated in Eq.(3) as our global information. It is defined as a list A having $2l$ number of $a_k \in \mathbb{R}$. In order to make the selection possible, a correspondence is drawn between A and the bit-strings s_i. Every non-overlapping pair of two consecutive a_k is used to represent a single bit position. For each pair of the list elements, we dedicate the former one as the score for 0 and the later one as the score for 1. Fig. 3 shows the correspondence of A and a bit-string.

Specifically, the generation of a binary string starts at the most-significant bit and proceeds towards the least-significant one, carrying the meaning of dividing the search space into half successively following the sample space hierarchy. The probabilities p and q of selecting 0 and 1 respectively at bit b_i given that bits b_{l-1} to b_{i+1} are generated are defined as follows:

$$p = a_k \quad \text{and} \quad q = 1 - p \tag{4}$$

ENVIRONMENT - GLOBAL INFORMATION

Given a reliable global information A^*, the searching agents described in the above section should be able to find s^* with probability approaching 1 fulfilling Eq.(2), i.e., $Prob(\text{select } s^*) \approx 1$. The question is how to make A^* reliable? We approach this problem as follows:

Assuming that the good performance of a binary string is due to its underlying components, we assign the raw fitness of the binary string to each of its constituting components. A population of searching agents of size N is distributed to try different partitions simultaneously. Their raw fitness values are assembled into *component fitness values*. The more partitions are tried, the more reliable the component fitness values are. The assembling is done in the following way: Let h_{k+c} be the component fitness of state $c \in \{0,1\}$ at bit position i in the current population. Then,

$$h_{k+c} = \frac{\sum_{j=0}^{N-1} F(x_j \mid b_i \text{ of } s_j \text{ equals } c)}{n_c} \tag{5}$$

where n_c is the total number of agents satisfying the constraint: b_i of s_j equals c. Every antagonistic pair of component fitness values are normalized in such a way that $h_k + h_{k+1} = 1$. Using these values to make decision, the searching agents should be able to produce better binary strings, as they have an immediate past searching experience to rely on. Continuously using the newly produced h_k 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 are retained as follows: At time t,

$$a_{k+c}(t) = \beta_i \, a_{k+c}(t-1) + (1-\beta_i)\, h_{k+c}(t-1) \tag{6}$$

with $a_k(t) + a_{k+1}(t) = 1$. We call β_i as *remembrance*. It is the fraction of the past collected information retained at bit i in the next time step. As indicated in the equation, different bits have different remembrance values. There are two reasons why this is so:

1. The more significant bits controlling larger common partitions should have more reliable information collected given the same number of samples. This supports losing more past information to increase convergence speed.
2. The hierarchical structure has an advantage on search space reduction. The reduced size suggests a smaller remembrance value be used to speed up the convergence.

Therefore, we devised an *adaptive remembrance scheme.* Let τ be a threshold value above which means converged and β be the minimum allowed remembrance. Suppose b_r is the first encountered bit considered from the most significant side that satisfies: $\mid 0.5 - a_{2(l-1-r)} \mid \, > \tau \;\; \vee \;\; \mid 0.5 - a_{2(l-r)} \mid \, < \tau$ Then β_i is set according to:

$$\beta_i = \begin{cases} \beta & l-1 \geq i \geq r \\ \dfrac{r-i+\beta}{r-i+1} & r > i \geq 0 \end{cases} \tag{7}$$

This scheme, basically, keeps the remembrance for the converged bits (b_l to b_{r+1}) constant at β, while interpolates the rest from β to $(r+\beta)/(r+1)$.

4. HIGH-DIMENSIONALITY

We solve n-dimensional problems, $F(x)$, $x \in X^n$, by extending the basic model to a cooperative-competitive one. A population described in previous sections are dedicated to a single dimension. We call such population as subpopulation. For an n-dimensional problem, we have a set of n subpopulations. We refer such a set as a subgroup. The raw fitness of each binary string is determined by how well it cooperates with the elite [5]. Suppose the current elite x^e is $[x^e_0, x^e_1, \cdots, x^e_{n-1}]$. The fitness of the jth binary string $s_{0,j}$ of the subpopulation responsible for the 0-th dimension is equal to $cf(x_{0,j}, x^e) = F(x_{0,j}, x^e_1, \cdots, x^e_{n-1})$.

Owing to the high greediness of this approach and the assumption of the independence among the dimensions, competition is introduced. Instead of keeping one subgroup, we keep G number of subgroups. They are allowed to compete with each other for the 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.

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 overlapping in *all* dimensions. There are two metrics required to calculate the repulsive force: (i) *degree*

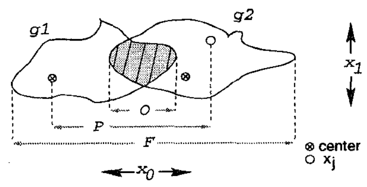

Fig. 4. Overlapping of two subgroups

of overlapping and (ii) *proximity*. For each dimension i, we measure the distance F which is the largest among all pairs of binary strings in the two subgroups under consideration. Denote $g1_{min}$ and $g1_{max}$ as the minimum and the maximum of $g1$ respectively, $g2_{min}$ and $g2_{max}$ as the minimum and the maximum of $g2$ respectively, $F = \max\{g1_{max}, g2_{max}\} - \min\{g1_{min}, g2_{min}\}$. Minimum value of F is 0 when all binary strings in $g1$ and $g2$ are identical, while the maximum possible F value equals $\max X - \min X$. We also measure the distance O of the region where they overlap (see Fig. 4). Overlapping distance O equals 0 when $g1_{max} < g2_{min}$ or $g2_{max} < g1_{min}$. Degree of overlapping $D_i(g1, g2)$ between the same dimension i of the two subgroups is defined as: $D_i(g1, g2) = \frac{O}{F}$.

Assuming that the 'center' of a dimension of a subgroup g is where the elite is located, every binary string $s_{i,j}$ in the neighboring subgroup is assigned a proximity value $P_i(g, x_{i,j})$ equal to the distance to the center of the subgroup g. Repulsive force $R_i(g1, x_{i,j})$ for the binary string is equal to $D_i(g1, g2) \times (1 - P_i(g1, x_{i,j}))$. Another quantity *interaction fitness* $I_{i,j}$ is defined to indicate how well a binary string performs in the competition: $I_{i,j} = cf(x_{i,j}, x^e) \times R_i(g, x_{i,j})$. Instead of feeding back f_j into system, $I_{i,j}$ should be used. The F in Eq. 5 is then replaced by $I_{i,j}$.

5. pccBHS AS A GENETIC OPERATOR

The design of pccBHS shares a number of similarities with the canonical genetic algorithms. Firstly, they are classified as iterative probabilistic search. Secondly, chromosome/binary string is the basic object to be manipulated. Thirdly, they are population-based approaches. Based on these similarities, we integrated them to become a hybrid algorithm in order to gain the benefits from both. However, we should state clearly that it is a preliminary model provided to initiate further research. Moreover, the cooperative part of pccBHS is not built into the hybrid model.

Before describing the model, we list below its main

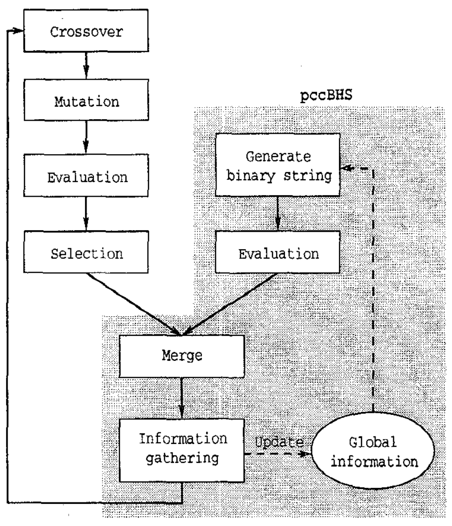

Fig. 5. Hybridization of GA and pccBHS

characteristics:

- Merging of populations from both parties will be taken. A parameter γ is introduced to control the proportions of chromosomes of GA and the binary strings of pccBHS to be passed to the next generation.
- The operator pccBHS is different in nature to the basic GA operators such as crossover and mutation. These basic GA operators can be said to be transformation functions mapping a population of chromosomes into another population of the same universal set. pccBHS is different in that it generates a complete new set of binary strings instead of transforming the set from the last generation. Hence, in one aspect, the hybrid model has two cycles (GA cycle and pccBHS cycle) running in parallel. In another aspect, pccBHS can be viewed as an operator plugged into the GA cycle. It is made possible by the presence of a rendezvous–merging of two populations.

To aid in understanding, we show the model in Fig. 5. On the left hand side of the figure, there shows the normal GA components such a crossover, mutation, selection and evaluation. Consider the cycle on the left only, if the `Merge` and `Information gathering` processes are empty, we got a normal GA cycle. On the right hand side of the figure (the shaded region), there shows a normal pccBHS algorithm. These two separate cycles are connected together by the `Merge` and the `Information gathering` processes. The former one is a simple process that joins the set of chromosomes from GA cycle with the set of binary strings from the pccBHS cycle to form a

Table 1. Test functions and results

Test functions					Test results		
Problems	n	f^*	#eval	Ref.	f^+	#eval	Cond.$^\triangle$
S1 - Shekel	1	14.59265	1,186	‡	14.59265	**915**	β=0.94
H3 - Hartman	3	3.86	2,500/972/1,459	♮	3.861400	**709**	β=0.95
H6 - Hartman	6	3.32	4,154	♭	3.320700	**4,847**	β=0.8
A30 - Ackley	30	0.001	13,997/19,420	†	-0.00078	**18,680**	β=0.4
A100 - Ackley	100	0.001	57,628/53,860	†	-0.00074	**58,216**	β=0.35
R20 - Rastrigin	20	0.9	6,098/3,608	†	-0.48987	**5,413**	β=0.45
R100 - Rastrigin	100	0.9	45,118/25,040	†	-0.54718	**45,195**	β=0.45

‡: Our GA expt.
♮: MGs [4]/Our GA expt./SA [11]
♭: Clustering [12]
†: GA [10] [EASY/BGA]
#eval: Number of function evaluations.

$\triangle$ Conditions: N=40, μ=1, Number of runs=50.
f^+: Average function value attained.

combined populations of size the sum of both. The information of the combined population is accumulated in the `Information gathering` process. The information gathered is then used in generating new chromosomes by pccBHS. As mentioned before, there is a parameter which controls the proportions of chromosomes from GA and binary strings from pccBHS. We call this parameter mix ratio $\gamma \in [0.0, 1.0]$. It reflects the relative contribution of GA and pccBHS in the gathered information. Given a γ and the size N of the final merged population, there are γN number of chromosomes contributed by GA and $(1-\gamma)N$ of binary strings contributed by pccBHS. In our model, $\gamma = 0.0$ does not mean a pure pccBHS, since all the pccBHS binary strings will be processed by GA operators.

6. EXPERIMENTS

In this section, we present two set of simulation results on solving a number of well-known and commonly used numeric functions. Experiment 1 illustrates the performance of the basic pccBHS model, while experiment 2 illustrates the performance of the hybrid model.

EXPERIMENT 1

In this experiment, we tried several well-known problems with problem size up to 100 dimensions which are listed on the left of Table 1. While the results are listed on the right of the same table. It shows clearly that the performance of our algorithm is comparable with (and even outperform) the existing advanced techniques, namely genetic algorithms (e.g. breeder genetic algorithm (BGA) [9], evolutionary algorithm with soft genetic operators (EASY) [10]), simulated annealing [11], and clustering (new Price's algorithm) [12], and multistart greedy descent [4].

EXPERIMENT 2

In this section, the performance of the hybrid model is illustrated. Several commonly used numeric functions are used: Goldstein-price, Rastrigin, and Hartman. The experimental condition used for each function is stated in the Tables. 2, 3, and 4 along with the corresponding results. Throughout the three test cases, one-point crossover is used with crossover rate 1.0, point mutation is used with probability $1/l$, and the selection is 2-tournament. All of the results indicate that the hybrid algorithm ($\gamma = \{0.0, 0.25, 0.50, 0.75\}$) outperforms the canonical GA ($\gamma = 1.0$) by using less number of function evaluations to achieve the same/similar level of performance (success rate).

Table 2. Performance of the hybrid model - Goldstein-Price (n=2),$f^+ = -3.000055$,$N = 60$, $\beta = 0.80$,$G = 1$

γ	Succ. rate	Ave. f.e.
0.00	96	2856.9
0.25	97	3100.2
0.50	98	3682.0
0.75	100	3579.6
1.00	86	5215.8

Table 3. Performance of the hybrid model - Rastrigin (n=2),$f^+ = 1.9997$,$N = 50$, $\beta = 0.90$,$G = 1$

γ	Succ. rate	Ave. f.e.
0.00	93	1838.1
(0.00	99	2364.3)†
0.25	99	1844.5
0.50	100	2018.2
0.75	100	2192.5
1.00	100	3676.0

†: $\beta = 0.95$

Table 4. Performance of the hybrid model - Hartman (n=3). f^+=3.860,$N = 30$,$\beta = 0.75$,$G = 1$

γ	Succ. rate	Ave. f.e.
0.00	96	483.1
0.25	98	489.6
0.50	100	554.9
0.75	99	571.6
1.00	88	1302.6

7. CONCLUSION

We have proposed a hierarchical view of the sample space subdivision which reduces the search size dramatically and provides a basis for controlling resolution. Coupled with the information processing cycle created by the collective contribution of samples and the global searching environment, reliable global information becomes available. With the introduction of cooperative-competitive paradigm, the algorithm can be extended to solve high-dimensional problems with comparable performance to (even outperforms) the existing promising techniques. Moreover, a hybrid algorithm is designed which is an integration of pc-cBHS with genetic algorithm. The hybrid algorithm is found to outperform the genetic algorithm tested.

In this work, we have exploited very minimal potential of the resolution control property of the hierarchy and the gathered global information. Hence, one of the future work would be the design of a better adaptive learning algorithm and a better searching mechanism. Furthermore, extension and refinement are needed to improve the primitive hybrid algorithm.

ACKNOWLEDGMENT

This research is partially supported by a Hong Kong Government RGC Earmarked Grant, Ref. No. CUHK352/96E.

APPENDIX

1. ALGORITHM OVERVIEW

```
Procedure InformationProcessingCycle
  global environment ← Empty
  While stopping critera 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
```

REFERENCES

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

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

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

[4] A.H.G. Rinnooy Kan and G.T. Timmer. Stochastic methods for global optimization. *American J. Mathematics and Management Sciences*, 4, 1984.

[5] 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, *PPSN III.* Springer-Verlag; Berlin, Germany, 1994.

[6] A. Pétrowski. A clearing procedure as a niching method for genetic algorithms. In *Proc. 1996 IEEE ICEC*, pages 798–803. IEEE; New York, NY, USA, 1996.

[7] B.L. Miller and M.J. Shaw. Genetic algorithms with dynamic niche sharing for multimodal function optimization. In *Proc. 1996 IEEE Intl. Conf. Evolutionary Computation (ICEC'96).* IEEE; New York, NY, USA, 1996.

[8] Goldberg D.E. and Richardson J. Genetic algorithms with sharing for multimodal function optimization. In *Genetic Algorithms and their Applications: Proc. 2nd Intl. Conf. Genetic Algorithms*, pages 14–21, 1987.

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

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

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

[12] 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.