KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2007
Cluj-Napoca (Romania), June 6–8, 2007, pp. 161–168

EXACT MODEL BUILDING IN HIERARCHICAL COMPLEX SYSTEMS

DAVID ICLĂNZAN[1] AND DAN DUMITRESCU[2]

Abstract. The paper proposes a novel trajectory based method which optimizes problems via explicit decomposition. The method is able to learn and deliver the problem structure in a comprehensible form to human researchers.

1. Introduction

Complex systems are characterized by multiple interactions between many different components where local and global phenomena interact in complicated, often nonlinear ways [6]. Researchers from different areas confront with them on a daily basis. To successfully address large-scale problems in complex systems, a proper problem decomposition must be employed.

A special type of complex systems are the hierarchical ones, where the system is composed from subsystems, each of which is hierarchical by itself [7]. Many systems around as are hierarchical. The successive levels used in physics are familiar to everyone: materials are composed of molecules, molecules are composed of atoms, atoms are composed electrons, protons, neutrons and so forth.

Hierarchical problems derive from hierarchical complex systems. Their efficient solving requires proper problem decomposition and assembly of solution from subsolution with strong non-linear interdependencies.

Pelikan and Goldberg [5] proposed the Hierarchical Bayesian Optimization Algorithm (hBOA), one of the few methods which is able to optimize problems with random linkages. hBOA can optimize problems which are not fully decomposable in one single level by hierarchical decomposition. Nevertheless, the decomposition information is implicitly stored in a Bayesian network which do not reveal the problem structure in a comprehensible form.

In a very recent development Yu and Goldberg [12] proposed an explicit hierarchical decomposition scheme for GAs. The method uses dependency structure matrix clustering techniques for linkage detection and it is able to approximately

2000 *Mathematics Subject Classification.* 68T20, 49M27, 91E40.
Key words and phrases. Problem Decomposition, Linkage Learning, Model Building.

capture the underlying problem structure. The inaccuracies are mainly caused by the intrinsic probabilistic nature of the algorithm.

In this paper we propose the Building Block Wise Greedy Search Algorithm (BBWGSA), a more systematic approach which can exactly capture the problems anatomy. The proposed method operates on a single point -instead of populations- and uses an explicit hierarchical decomposition scheme based on a greedy search operator which performs local search among competing building blocks (BBs) neighborhood. The search experience accumulated is used to reveal linkages and to update the BB structure.

The rest of the paper is organized as follows. Section two revisits and formally presents hierarchical problems. The third section describes the proposed method. Experiments and results are presented in section four. The paper is concluded with discussion and some outlines for future work in chapter five.

2. Hierarchical Difficulty and Hierarchical Problems

Although having a gross-scale BB structure, hierarchical problems are hard to solve without proper problem decomposition as the blocks from these functions are not separable.

The fundamental of hierarchically decomposable problems is that there is always more than one way to solve a (sub-) problem [9] leading to the separation of BBs "fitness" i.e. contribution to the objective function, from their meaning. This conceptual separation induces the non-linear dependencies between BBs: providing the same objective function contribution, a BB might be completely suited for one context whilst completely wrong for another one.

Hierarchical problems are very hard for mutation based hill-climbers as they exhibit a fractal like structure in the Hamming space with many local optima [11]. This bit-wise landscape is fully deceptive; the better is a local optimum the further away is from the global ones. At the same time the problem can be solved quite easily in the BB or "crossover space", where the block-wise landscape is fully non deceptive [9]. The forming of higher order BBs from lower level ones reduces the problem dimensionality.

If a proper niching is applied and the promising sub-solutions are kept until the method advances to upper levels where a correct decision can be made, the hierarchical difficulty can be overcome.

2.1. **Design of hierarchical problems.**

In this paper two hierarchical test functions are considered: the hierarchical IFF [9] and the hierarchical XOR [10]. These problems are defined on binary strings of the form $\{0,1\}^{k^p}$ where k is the number of sub-blocks in a block, and p is the number of hierarchical levels. The meaning of sub-blocks is separated from their fitness by the means of a boolean function h which determines if the sub-block is valid in the current context or not. In

the shuffled version of these problems the tight linkage is disrupted by randomly reordering the bits. The functions are detailed as follows.

2.1.1. *Hierarchical if and only if (hIFF).* The hIFF has $k = 2$ and it is provided by the if and only if relation, or equality. Let $L = x_1, x_2, \ldots, x_{2^{p-1}}$ be the first half of the binary string x and $R = x_{2^{p-1}+1}, x_{2^{p-1}+2}, \ldots, x_{2^p}$ the second one. Then h is defined as:

$$h_{iff}(x) = \begin{cases} 1 & \text{, if } p = 0; \\ 1 & \text{, if } h_{iff}(L) = h_{iff}(R) \text{ and } L = R; \\ 0 & \text{, otherwise.} \end{cases} \tag{1}$$

Based on h_{iff} the hierarchical iff is defined recursively:

$$H_{iff}(x) = H_{iff}(L) + H_{iff}(R) + \begin{cases} length(x) & \text{, if } h_{iff}(x) = 1; \\ 0 & \text{, otherwise.} \end{cases} \tag{2}$$

At each level $p > 0$ the $H_{iff}(x)$ function rewards a block x if and only if the interpretation of the two composing sub-blocks are both either 0 or 1. Otherwise the contribution is zero.

hIFF has two global optima: strings formed only by 0's or only by 1's. At the lowest level the problem has $2^{l/2}$ local optima where l is the problem size.

2.1.2. *Hierarchical exclusive or (hXOR).* The global optima of hIFF are formed by all 1's or all 0's which may ease the task of some methods biased to a particular allele value. To prevent the exploitation of this particular problem property the hXOR was designed.

The definition of hXOR is analogous with the hIFF, having only a modification in the validation function h, where instead of equality we do a complement check.

$$h_{xor}(x) = \begin{cases} 1 & \text{, if } p = 0; \\ 1 & \text{, if } h_{xor}(L) = h_{xor}(R) \text{ and } L = \bar{R}; \\ 0 & \text{, otherwise.} \end{cases} \tag{3}$$

The $\bar{R}$ stands for the bitwise negation of R.

3. Systematic Exploitation of the Building Block Structure

As already indicated in Section 2, hierarchical problems are fully deceptive in Hamming space and fully non deceptive in the BB space. The problem representation together with the neighborhood structure defines the search landscape. With an appropriate neighborhood structure, which operates on BBs, the search problem can be transferred from Hamming space to a very nice, fully non deceptive search landscape which should be easy to systematically exploit (ex. hill-climb).

Usually hill-climbers described in the literature use bit-flipping for replacing the current state [1, 2, 3]. This implies a neighborhood structure which contains strings that are relatively close in Hamming distance to the original state, making those

methods unsuited for solving hierarchical problems, where local optima and global optima are distant in Hamming space. But the neighborhood can be defined as an arbitrary function which assign to a valid state s a set of valid states $N(s)$. The main idea of the paper is to build a trajectory method which takes into account the BB structure of the problems and defines its neighborhood structure accordingly.

3.1. **Adaptation of the neighborhood structure.** Theoretical studies denote that a GA that uses crossover which does not disrupts the building block structure holds many advantages over simple GA [8].

Similarly, in order to be able to efficiently exploit the BB landscape, the proposed method must learn the problem structure and evolve the solution representation to reflect the current BB knowledge. The changing of representation implies the adaptation of the neighborhood structure which is the key to conquer hierarchical problems: by exploring the neighborhood of the current BB configuration the next level of BB can be detected.

In order to be able to identify linkages we enhance our method with a memory where hill-climbing results are stored. Evolutionary Algorithms with linkage learning mechanism extracts the BB information from the population. Similar techniques can be applied to devise BB structures from the experience stored in the memory. However, the solutions stored by the proposed method offer an important advantage over populations: they are noise free. While individuals from populations may have parts where good schemata's have not yet been expressed or have BBs slightly altered by mutation, the solution stored in memory are always exact local optima. In the case of fully non deceptive BB landscapes the systematic exploration of BB configurations guarantees that in the close neighborhood of these states there are no better solutions.

3.2. **Building Block Wise Greedy Search Algorithm.** The proposed method involves three main steps: (i) hill-climbing the search space according to a BB neighborhood structure; (ii) local optima obtained in (i) are used to detect linkages and extract BB information; (iii) the BB configuration and implicitly the neighborhood structure are updated.

3.2.1. *The Greedy Search Algorithm.* BB hill-climbing is rather straightforward: instead of flipping bits, the search focuses on the best local BB configuration. Each BB is processed systematically by testing its configurations and selecting the one which provides the highest (or lowest in the case of minimization) objective function value. While the best configuration of a particular BB is searched, the configurations of the other BBs are hold still.

The individual is represented as a sequence of BBs: $s = (b_1, b_2, \ldots, b_n)$ where n is the number of BBs. Each BB b_i can represent multiple configurations: $V_i = \{v | v \in \{0,1\}^l\}$ where l is the length of b_i. This allow the sustenance and parallel processing of competing schemata.

1. Choose randomly a building block b_i from s which has not yet been clustered;
2. Let L be the set of building blocks whose configuration from the memory are mapped bijectively to b_i;
3. If L is empty update the possible configurations V_i to the configurations encountered in the memory;
4. If L is not empty form a new building block $new\_b = b_i \bigcup L$ by setting the loci it's define to the union of loci from b_i and the building blocks from L. Also set the possible values V_{new_b} to all distinct configuration encountered, on the position defined by the new_b , operating on the binary representation of states from the memory;
5. Set $b_i \bigcup L$ as clustered;
6. If there exists building blocks which have not been clustered *goto 1*;

FIGURE 1. The linkage detection and new building block forming method.

3.2.2. *Linkage Detection.* Several techniques for detecting gene dependency from a population have been already presented in the literature [4, 12] which could be also employed by the proposed method. But due to the fact that the hierarchical problems under study are fully non deceptive in the BB space, a very simple method for linkage detection is considered. This process is facilitated by the advantage of having noise free states stored in memory.

The clustering of loci in new BBs is done by searching for bijective mappings. For a given block b_i, all BBs b_j are linked if distinct configurations of b_i map to distinct configurations of b_j. The configurations of b_i that can be found in the memory represent the domain while the configurations of b_j from the memory are the codomain.

Due to the transitivity property of bijective mappings (functions) all relevant BBs are discovered simultaneously. The linkage detection algorithm is presented in Figure 1. Harder problems (exhibiting overlapping BB structure for example), may require a more sophisticated linkage learning method.

All BBs linked together by a bijective mapping will form a new BB which replaces the linked loci in the BB structure. The possible configurations of the new BBs are extracted from the binary representation of states from the memory. All distinct configurations from the positions defined by the composing BBs are taken into account. If a BB can not be linked with any other BB it keeps its original place and only its possible configurations are updated in the same manner as the new BBs.

Proposed model can be summarized by the algorithm presented in Figure 2.

```
1.  Generate a random state s from the current BB structure;
2.  BB cill-climb from s and store the result in memory;
3.  If the resulted state is better then the best states seen so
far, keep the new state;
4.  If the memory is not filled up goto 1;
5.  Learn linkage from memory and update the BB configuration
according to the detected linkages;
6.  Empty memory;
7.  If termination condition not met goto 1;
```

FIGURE 2. Outline of the hill-climbing enhanced with memory and linkage learning. In steps 1-4 we accumulate the search experience (phase 1) which is exploited in steps 5-7 (phase 2).

4. Results

Enhanced with linkage learning mechanism and variable neighborhood structure the BBWGSA should be able to efficiently solve relevant problems by hierarchical decomposition. Also due to the more systematic approach of the BBWGSA we expect it to deliver uncorrupted problem structure.

We tested these hypothesis on the 128-bit, 256-bit shuffled hIFF and hXOR problems. The memory size was set to 30 on the case of the 128-bit version respectively to 40 on the 256-bit one. 25 independent runs were performed on both test suits. The BBWGSA was able to find one of the global optima in all cases. More important it detected the perfect problem structure (complete binary tree) in all runs! In Figure 3 we depict the performance of the DSGMA++ on a small test suit as reported in [12]. Albeit the problem structure is quite well approximated it contains some inaccuracies which on bigger problem instances may be problematic.

Similarly to other methods like the DSMGA++, the BBWGSA uses explicit chunking mechanism enabling the method to deliver the problem structure. While DSGMA++ and other stochastic methods have to fight the sampling errors which sometimes induce imperfections, the BBWGSA was able to detect the perfect problem structure in all runs, due to its more systematic and deterministic approach. The enhanced capability of BBWGSA to capture the problem structure is also revealed by the fact that hIFF and hXOR are solved approximately in the same number of steps as their underlying BB structures (complete binary tree) coincide. However for the DSGMA++ the time needed to optimize the two problems differs significantly, being $O(l^{1.84}\log(l))$ for the hIFF and $O(l^{1.96}\log(l))$ on hXOR.

On hIFF and hXOR the multiple runs of the BBWGSA showed an unbiased behavior, finding in almost half-half proportion both global optima.

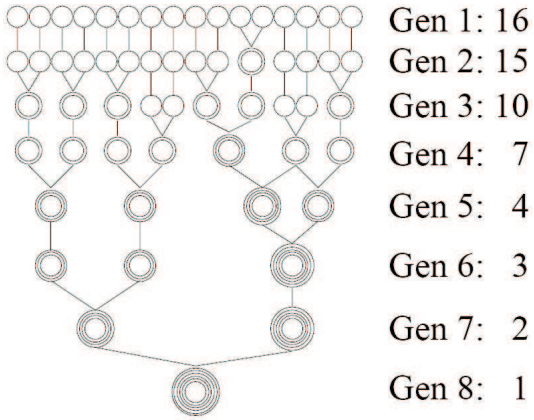

FIGURE 3. Example of problem structure obtained by the DSMGA++ for the hXOR function with 4 levels (chromosome length= $2^4 = 16$) as reported in [12]. The nodes represents genes of chromosomes. The number of circles represents the compression level. The descriptions on the right show the generation number and the chromosome length. The ideal case would be a complete binary tree with 4 layers.

A final remark concerns the stability of BBWGSA: the highest standard deviation encountered is 196 while other methods deal with standard deviations of much higher magnitude on the same test suites.

5. Conclusions

The Building Block Wise Greedy Search Algorithm (BBWGSA), a generic method for solving problems via hierarchical decomposition is proposed. The BBWGSA operates in the BB space where it combines BBs in a systematic and exhaustive manner. Exploration experience is used to learn the underlying BB structure of the search space expressed by linkages.

A very important aspect of the proposed method is that similar to DSMGA++, BBWGSA delivers the problem structure in a form comprehensible to humans. Gaining knowledge about the hidden, complex problem structure can be very useful in many real-world applications. Nevertheless, by adopting a more systematic approach, the proposed method was able to detect the perfect problem structure in all runs.

In the future we would like to enhance our method with more powerful linkage learning techniques in order to be able to tackle more difficult problem structures.

6. Acknowledgments

This work was supported by the CNCSIS, AT-70 / 2006 grant and the Sapientia Institute for Research Programs (KPI).

References

[1] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithm? some anomalous results and their explanation. *MACHLEARN: Machine Learning*, 13, 1993.

[2] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In F. J. Varela and P. Bourgine, editors, *Proc. of the First European Conference on Artificial Life*, pages 245–254, Cambridge, MA, 1992. MIT Press.

[3] H. Mühlenbein. How genetic algorithms really work: I. mutation and hillclimbing. In R. Männer and B. Manderick, editors, *Proceedings of the Second Conference on Parallel Problem Solving from Nature (PPSN II)*, pages 15–25, Amsterdam, 1992. North-Holland.

[4] M. Pelikan. *Bayesian optimization algorithm: from single level to hierarchy*. PhD thesis, 2002. Adviser-David E. Goldberg.

[5] M. Pelikan and D. E. Goldberg. Escaping hierarchical traps with competent genetic algorithms. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 511–518, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann.

[6] D. Rind. Complexity and climate. *Science*, 284(5411):105–107, April 1999.

[7] H. A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, Massachusetts, first edition, 1969.

[8] D. Thierens and D. E. Goldberg. Mixing in genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 38–47, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[9] R. A. Watson, G. S. Hornby, and J. B. Pollack. Modeling building-block interdependency. *Lecture Notes in Computer Science*, 1498:97–108, 1998.

[10] R. A. Watson and J. B. Pollack. Hierarchically consistent test problems for genetic algorithms: Summary and additional results. In S. Brave and A. S. Wu, editors, *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*, pages 292–297, Orlando, Florida, USA, 13 July 1999.

[11] R. A. Watson and J. B. Pollack. Symbiotic composition and evolvability. In J. Kelemen and P. Sosik, editors, *Advances in Artificial Life, 6th European Conf., (ECAL 2001)*, pages 480–490, Berlin, 2001. Springer.

[12] T.-L. Yu and D. E. Goldberg. Conquering hierarchical difficulty by explicit chunking: substructural chromosome compression. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1385–1392, New York, NY, USA, 2006. ACM Press.

[1] Babeș-Bolyai University, Faculty of Mathematics and Computer Science, Str. Mihail Kogălniceanu nr. 1, 400084, Cluj-Napoca, România
E-mail address: david.iclanzan@gmail.com

[2] Babeș-Bolyai University, Faculty of Mathematics and Computer Science, Str. Mihail Kogălniceanu nr. 1, 400084, Cluj-Napoca, România
E-mail address: ddumitr@cs.ubbcluj.ro