Babeș-Bolyai University of Cluj-Napoca
Faculty of Mathematics and Computer Science
Department of Programming Languages

Summary of Doctor of Philosophy Dissertation

New Techniques in Competent Search and Optimization

by

David-Andrei ICLĂNZAN

Supervisor: prof. dr. Dumitru DUMITRESCU

Cluj-Napoca, 2010

Contents

Chapter 1

Introduction

The ability to create intelligent machines able to emulate the biological intelligence of homo sapiens has intrigued humans since ancient times. The sound and fruitful investigation of intelligent machines begun with the advent of the computer, the rise of a new mathematical theory of information, and with the major discoveries in neurology. Artificial Intelligence (AI) was established within computer science, as a major field of research and study, focusing on creating machines that can engage on behaviors that humans consider intelligent.

Most of the problems in AI can be solved by performing an intelligent search through many possible solutions (Russel & Norvig, 1995). Alan Turing outlined three flavors of search that can lead to intelligent behavior: the logic-driven search approach, the cultural search approach, and the evolutionary search approach (Turing, 1969).

Many machine learning algorithms apply search algorithms based on optimization. As exhaustive, uninformed search is impractical in real world scenarios due to the tremendous search space sizes. Informed search procedures use heuristics, and make assumptions to guide the search and reduce the search space by eliminating choices that are unlikely. Many search bias techniques are inspired and mimic mechanisms acting in nature.

This dissertation investigates additional aspects of informed search where bias is introduced in an intelligent manner, as a result of data analysis and exploitation of search experience. In particular, we apply machine learning techniques to infer information about black box functions. This knowledge is used in conjunction with optimization techniques to solve hard optimization problems quickly, reliably, and accurately. The dissertation also analyzes techniques that improve scalability, for handling complexity and large scale optimization and general principles to produce open-ended solutions to given problems.

1.1 Problem Statement

One of the greatest and most interesting challenges are the problems arising from the study of Complex Systems. This field studies how relationships between parts relates to the collective behavior of a system and how the system interacts and forms relationships with its environment.

These systems are characterized by multiple interactions between many different components where local and global phenomena interact in complicated, often nonlinear ways (Rind, 1999). In many complex systems around us interactions do not manifest at a single level; they have a hierarchical organization, where the system is composed from subsystems, each of which is hierarchical by itself (Simon, 1969). To successfully address large-scale problems in complex systems, a proper problem decomposition must be used.

The aim of the thesis is to develop principled methodology and a general framework able to exploit search experience via data analysis and machine learning techniques, enabling the identification and exploitation of dependencies and modularity. The learning and adapting component extends simple methods making them more efficient, robust and most importantly scalable. Detecting rules and substructures of the problem on the fly, performing a dynamic decomposition, can guide the search, and it can render intractable problems tractable and solve them efficiently.

1.2 Contributions

The main contributions of this dissertation to the field include:

- Introducing the adaptive neighborhood structure in local-search optimizers where search strategies are extended and can adapt by using data analysis and machine learning techniques.
- Building the Model Based Local Search framework and showing that is scales up logarithmically on hierarchical problems.
- Establishing new benchmark functions for testing the influence of neutrality and massive multimodality and scalability on large-scale problems.
- Demonstrating the sustainability, scalability and reliability of the Model Based Local Search on problems that are intractable by classical methods.
- Eliminating the costly model search by introducing new automatic linkage detection methods based on neural networks.
- Introducing memory efficient on-line model building technique.

- Proposing the efficiency enhancement and the reduction of complexity in model building methods by incorporating pre-filtering based on pairwise interaction analysis.

- Development of a model evaluation free, fast, scalable, easily parallelizable model building method, based on the Markov Clustering Algorithm. The proposed model building technique can potentially be used to build models of three different categories: Bayesian networks, overlapping linkage models, and non-overlapping marginal product models

- Investigating the fundamentals of evolutionary algorithms and outlining new topological features of fitness landscapes for which crossover is an efficient operator.

- Proposing an unconventional sustainable competent search model based on cooperation, specialization and exploitation of search experience. The open-ended synthesis is guaranteed by the non-convergent part of the search, which endlessly explores for new traits which if found, are fostered by the convergent mechanism.

The thesis is based on the following publications:

(Iclanzan & Dumitrescu, 2010) David Iclanzan and D. Dumitrescu. Graph clustering based model building. In *PPSN XI - to appear in Lecture Notes in Computer Science*, Krakow, Poland, 11-15 September 2010. Springer. (accepted).

(Iclănzan et al., 2010) D. Iclănzan, D. Dumitrescu, and B. Hirsbrunner. Pairwise Interactions Induced Probabilistic Model Building. *Exploitation of Linkage Learning in Evolutionary Algorithms*, pages 97–122, 2010.

(Iclanzan et al., 2009a) David Iclanzan, D. Dumitrescu, and Béat Hirsbrunner. Correlation guided model building. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 421–428, New York, NY, USA, 8-12 July 2009. ACM.

(Szilágyi et al., 2009) László Szilágyi, David Iclanzan, Sándor M. Szilágyi, D. Dumitrescu, and Béat Hirsbrunner. A generalized c-means clustering model optimized via evolutionary computation. In *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'09, Jeju Island, Korea)*, pages 451 – 455, 2009.

(Iclanzan et al., 2009b) David Iclanzan, Béat Hirsbrunner, Michèle Courant, and D. Dumitrescu. Cooperation in the context of sustainable search. In *IEEE Congress on Evolutionary Computation (IEEE CEC 2009)*, pages 1904 – 1911, Trondheim, Norway, 18-21 May 2009.

(Szilagyi et al., 2009) Sandor M. Szilagyi, Laszlo Szilagyi, David Iclanzan, and Zoltan Benyo. A weighted patient specific electromechanical model of the heart. In *Proc. 5th International Symposium on Applied Computational Intelligence and Informatics (SACI 2009)*, pages 105–110, Timisoara, Romania, 28-29 May 2009.

(Szilágyi et al., 2008) László Szilágyi, David Iclanzan, Sándor M. Szilágyi, and D. Dumitrescu. Gecim: A novel generalized approach to c-means clustering. In José Ruiz-Shulcloper and Walter G. Kropatsch, editors, *CIARP*, volume 5197 of *Lecture Notes in Computer Science*, pages 235–242. Springer, 2008.

(Iclanzan & Dumitrescu, 2008c) David Iclanzan and D. Dumitrescu. Large-scale optimization of non-separable building-block problems. In Günter Rudolph, Thomas Jansen, Simon M. Lucas, Carlo Poloni, and Nicola Beume, editors, *PPSN*, volume 5199 of *Lecture Notes in Computer Science*, pages 899–908. Springer, 2008.

(Iclanzan & Dumitrescu, 2008d) David Iclanzan and D. Dumitrescu. Towards memoryless model building. In *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, pages 2147–2152, Atlanta, GA, USA, 2008. ACM.

(Iclanzan & Dumitrescu, 2008a) David Iclanzan and D. Dumitrescu. Going for the big fishes: Discovering and combining large neutral and massively multimodal building-blocks with model based macro-mutation. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 423–430, Atlanta, GA, USA, 2008. ACM.

(Iclanzan & Dumitrescu, 2008b) David Iclanzan and D. Dumitrescu. How can artificial neural networks help making the intractable search spaces tractable. In *2008 IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 4016–4023, Hong-Kong, 01-06 June 2008.

(Iclanzan & Dumitrescu, 2007c) David Iclanzan and D. Dumitrescu. Overrepresentation in neutral genotype-phenotype mappings and their applications. In *Symbolic and Numeric Algorithms for Scientific Computing, 2007. SYNASC. International Symposium on*, pages 427–432, Timisoara, Romania, 26-29 September 2007. IEEE Computer Society.

(Iclanzan et al., 2007) David Iclanzan, P.I. Fulop, and D. Dumitrescu. Neuro-Hill-Climber: A new approach towards more intelligent search and optimization. In *Symbolic and Numeric Algorithms for Scientific Computing, 2007. SYNASC. International Symposium on*, pages 441–448, Timisoara, Romania, 26-29 September 2007. IEEE Computer Society.

(Iclanzan, 2007a) David Iclanzan. The creativity potential within Evolutionary Algorithms. In Fernando Almeida e Costa et al., editor, *Advances in Artificial Life, 9th European Conference, ECAL 2007, Lisbon, Portugal, September 10-14, 2007, Proceedings*, volume 4648 of *Lecture Notes in Computer Science*, pages 845–854. Springer, 2007.

(Iclanzan, 2007b) David Iclanzan. Crossover: the divine afflatus in search. In Peter A. N. Bosman, editor, *Late breaking paper at Genetic and Evolutionary Computation Conference (GECCO'2007)*, pages 2497–2502, London, United Kingdom, 7-11 July 2007. ACM Press.

(Iclanzan & Dumitrescu, 2007b) David Iclanzan and D. Dumitrescu. Overcoming hierarchical difficulty by hill-climbing the building block structure. In Dirk Thierens et al., editor, *GECCO '07: Proceedings of the 9th annual conference on Genetic and Evolutionary Computation*, volume 2, pages 1256–1263, London, 7-11 July 2007. ACM Press.

(Iclanzan & Dumitrescu, 2007a) David Iclanzan and D. Dumitrescu. Exact model building in Hierarchical Complex Systems. In *Studia Universitatis Babes-Bolyai, Informatica Series*, volume Special Issue: KEPT 2007 - Knowledge Engineering: Principles and Techniques, Proceedings, pages 161–168, Cluj-Napoca, Romania, 6-8 June 2007. Universitas Napocensis, Presa Universitara.

(Szilagyi et al., 2006c) Sandor M. Szilagyi, Laszlo Szilagyi, David Iclanzan, and Zoltan Benyo. Unified neural network-based adaptive ECG signal analysis and compression. *SB-UPT TACCS*, 56(65)(4):27–36, 2006.

(Iclanzan et al., 2006) David Iclanzan, Sandor M. Szilagyi, Laszlo Szilagyi, and Zoltan Benyo. Advanced heuristic methods for ECG parameter estimation. In *CONTI 2006: Proceedings of the 7th International Conference on Technical Informatics*, pages 215–220, Timisoara, Romania, 8-9 June 2006. Universitatea Politechica.

(Szilagyi et al., 2006b) Sandor M. Szilagyi, Laszlo Szilagyi, David Iclanzan, and Zoltan Benyo. Adaptive ECG signal analysis for enhanced state recognition and diagnosis. In *CONTI 2006: Proceedings of the 7th International Conference on Technical Informatics*, pages 209–214, Timisoara, Romania, 8-9 June 2006. Universitatea Politechica.

(Szilagyi et al., 2006a) Laszlo Szilagyi, Sandor M. Szilagyi, David Iclanzan, and Zoltan Benyo. Quick ECG signal processing methods for on-line holter monitoring systems. In *CONTI 2006: Proceedings of the 7th International Conference on Technical Informatics*, pages 221–226, Timisoara, Romania, 8-9 June 2006. Universitatea Politechica.

(Iclanzan & Dumitrescu, 2006b) David Iclanzan and D. Dumitrescu. ECG parameter estimation using advanced stochastic search methods. In Dorin Isoc and Eugen Stancel, editors, *2006 IEEE-TTTC International Conference on Automation, Quality and tesing, Robotics. Digest of Junior Section*, pages 17–22, Cluj-Napoca, Romania, 25-28 May 2006. Universitatea Technica Cluj.

(Iclanzan & Dumitrescu, 2006a) David Iclanzan and D. Dumitrescu. Competitive vs. cooperative optimization. In *Proceedings of the 8th International Scientific Student Conference on Technical Sciences*, pages 115–121, Timisoara, Romania, 7-9 April 2006. Universitatea de Vest. Distributed on CD.

(Iclanzan, 2006) David Iclanzan. Genetic engineering as an optimization paradigm. In *Proceedings of FMTU 2006*, pages 115–121, Cluj-Napoca, Romania, 24-25 March 2006. Transylvanian Musem Society.

(Iclanzan, 2005) David Iclanzan. Genetic engineering algorithm. In *Proceedings of the 7th International Scientific Student Conference on Technical Sciences*, Timisoara, Romania, 22-24 April 2005. Universitatea de Vest. Distributed on CD.

1.3 Keywords

Efficiency enhancement, model-building, machine learning, adaptive neighborhood structure, model based local-search.

Chapter 2

Background and Related Work

This chapter gives a brief introduction to global optimization with emphasize on evolutionary algorithms. The shortcomings of classical methods leading to premature convergence and the ones of fixed representations and operators are discussed. The last part presents major ideas behind competent methods, designed for supporting sustainable evolution. It is briefly discussed how and at what costs can model building alleviate the problem of building block disruption.

2.1 Global Optimization

2.2 Evolutionary Algorithms

2.2.1 Genetic Algorithms

2.2.2 Evolutionary Programming and Evolution Strategies

2.2.3 Genetic Programming

2.2.4 Issues with Classical Evolutionary Methods

2.3 Sustainable Evolutionary Computation and Competent Methods

2.3.1 Premature Convergence

2.3.2 Building Blocks and Model Building

Chapter 3

Correlation Guided Model Building

This chapter presents basic results demonstrating how simple variable correlation data can be extended and used, to efficiently guide the model search, decreasing the number of model evaluations by several orders of magnitude and without significantly affecting model quality.

As a case study, we replace the $O(n^3)$ model building of the Extended Compact Genetic Algorithm by a correlation guided search of linear complexity.

3.1 Introduction

3.1.1 General measurement of module-wise interactions

Let M_R contain the absolute values from the correlation coefficient matrix but with self-correlations set to zero, i.e $M_R(x, x) = 0$. The first module-wise metric simply averages the different pairwise interactions:

$$d_1(X, Y) = \frac{\sum_{x \in X} \sum_{y \in Y} M_R(x, y)}{\binom{|X \bigcup Y|}{2}} \tag{3.1}$$

As it averages, d_1 is influenced by outliers, penalizing the incorporation of non-correlated components. For quantifying the strong interactions even in subsets of the modules, we introduce a metric which just sums up interactions:

$$d_2(X, Y) = \sum_{x \in X} \sum_{y \in Y} \sigma(x, y) \cdot M_R(x, y) \tag{3.2}$$

where

$$\sigma(x,y) = \begin{cases} 1 & \text{, if } M_R(x,y) \text{ is statistically significant;} \\ 0 & \text{, otherwise.} \end{cases} \tag{3.3}$$

In d_2 non-correlated components can not heavily bias the outcome. This also means that too complex module formation are not penalized. The two metrics are complemental and can be used together. If d_2 is high but d_1 has a small value, we should consider splitting the modules in multiple pieces: there are strong interactions but not all (x, y) variable pairs are correlated.

3.2 Case Study on eCGA

The eCGA (Harik, 1999) is a multivariate extension of the Compact Genetic Algorithm (Harik et al., 1999) based on the key principle that learning a good probability distribution of the population is equivalent to the linkage learning process. The measure of a good distribution is quantified based on minimum description length (MDL) models. MDL is pillared on the concept that any regularity in a given set of data can be used to compress the data.

Starting from a random population, the eCGA applies the process of evaluation, selection, MPMs based model-building and sampling until a halting criterion is met.

Model building in eCGA involves a very expensive computational task as the determination of MD: based criteria function for each tested model, requires the model to be fitted against the (large) population. The method has $O(n^3)$ complexity over the combined complexity criterion evaluation.

3.2.1 Hybridization of eCGA

A commonly used efficiency enhancement mechanism is the hybridization with local-search techniques. In the case of the eCGA, incorporating local-search in the sub solution search space leads to better results and greater robustness (Lima et al., 2005).

3.2.2 Guided linear model building

When searching for a proper MPM, eCGA greedily searches the space of possible models evaluating *all* pairwise partition merges. The model is extended sequentially with the best possible improvement obtained by joining modules. The main idea of the proposed search is to not process the list of possible extensions blindly and exhaustively: the best extension(s) to be considered are based on correlation analysis between the partitions. The search stops immediately when the model extension does not improve the combined complexity criterion. The reasoning behind this is that all other partition merges that are not analyzed would have

Algorithm 1: Correlation guided model-building

```
1  Build initial model m where each variable is an independent partition;
2  Compute M_R and M_T;
3  if this is the first generation and there are no significant values in M_T
   then
4      Request a bigger population and suggest performing search for second
       order interactions;
5      Halt the search;
6  repeat
7      [p, q] ← StrongestInteraction(m, M_R, d);
8      Form new model m' based on m but with p and q merged into a joint
       partition;
9      Evaluate combined complexity criterion C_c(m');
10     if m' improves over m then
11         m ← m';
12 until No improvement was found;
```

(according to the correlation based measurements) lower degree of interactions then the last proposed extension. If that extension was rejected by the combined complexity criterion (C_c) then all the remaining ones would be also discarded, there is no point to continue the analysis.

The correlation guided model-building is presented in Algorithm 1.

In terms of combined complexity criterion evaluations, the proposed method is very efficient, being linear.

3.3 Test suite

We test the eCGA with correlation guided model-building on two problems that combine the core of two well known problem difficulty dimensions:

- Intra-BB difficulty: *deception* due trap functions.

- Extra-BB difficulty: non-linear dependencies due to hierarchical structure, which at a single hierarchical level can be interpreted as exogenous *noise* – generated by interactions from higher levels.

3.3.1 Concatenated k-Trap function

3.3.2 Hierarchical XOR

3.4 Results and Discussion

The linear runtime of the correlation guided model-building in eCGA provides a huge *qualitative* advantage over the $O(n^3)$ classic model-building complexity. A heuristic based model-building with a $O(n^2)$ complexity had been shown to speed up the eCGA up to more than 1000 times (Duque et al., 2008).

Therefore, instead of providing a *quantitative* run-time comparison between eCGA with the proposed and the classical model-building, we concentrate the empirical investigation on the scaling, model quality – number of generations until convergence and the effect of hybridization.

3.4.1 Test setup

We tested the correlation guided eCGA with and without local-search hybridization (denoted as eCGA_h) on concatenated 4-Trap and lhXOR with $l = 3$ for problem sizes $psize = \{32, 64, 256\}$. Population sizes are $15psize$ for eCGA_h and $55psize$ for eCGA. These values were not tunned. A number of 10 runs were averaged for each test case. Results are presented in Figure 6.1 and discussed in the followings.

3.4.2 Analysis

In all cases the algorithms have found a global optima and the correct structures. Figure 6.1 shows the number of function evaluations needed per problem size.

Chapter 4

Model Based Local Search

In this chapter we introduce the Model Based Local Search framework, namely a hill-climber operating over the building block space that can efficiently address hierarchical problems by learning the problem structure from search experience. The neighborhood structure is adapted whenever new knowledge about the underlying building block structure is incorporated into the search. This allows the method to climb the hierarchical structure by revealing and solving consecutively the hierarchical levels.

We show that for fully non-deceptive hierarchical building block structures propose approach can solve hierarchical problems in linearithmic time, clearly outperforming population based recombinative methods.

4.1 Motivation

One of the most important research goal regarding Evolutionary Algorithms (EAs) is to understand the class of problems for which these algorithms are most suited. Despite the major work in this field it is still unclear how an EA explores a search space and on what fitness landscapes will a particular EA outperform other optimizers such as hill-climbers.

4.2 Hierarchically Decomposable Functions

4.2.1 Hierarchical Problems

Hierarchical if and only if (hIFF)

Hierarchical XOR (hXOR)

The Hierarchical Trap Function (hTrap)

4.2.2 Hill-climbers and neighborhood structure

4.2.3 Building Block wise search

As already indicated, hierarchical problems are fully deceptive in Hamming space and fully non-deceptive in the building block space. The problem representation together with the neighborhood structure defines the search landscape. Recent local-search literature authors have emphasized the importance of using a good neighborhood operator (Watson et al., 2003) With an appropriate neighborhood structure – which operates on building blocks – the search problem can be transferred from Hamming space to a very nice, fully non-deceptive search landscape, which should be easy to hill-climb.

While EAs exploit building block structure by probabilistic recombination, this approach applies a systematic combination and analysis of building blocks.

4.2.4 Online adaptation

It is important for a GA to conserve building blocks under crossover. Theoretical studies denote that a GA that uses crossover which does not disrupt the building block structure, holds many advantages over simple GA (Thierens & Goldberg, 1993). To achieve this goal linkage learning is applied and the solution representation is evolved along with the population, during the search process.

Similarly, in order to be able to hill-climb the building block landscape, the building block structure of the problem must be learned and the representation of the individual must be evolved to reflect the current building block 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 building block configuration the next level of BBs can be detected.

4.3 Building Block Hill-Climber

BBHC involves four main steps: (i) initialization of the algorithm with each single locus as a BB; repeatedly (ii) hill-climb the search space according to a BB-wise

neighborhood structure; (iii) local optima obtained in (ii) is used to detect linkages and extract building block information; (iv) the building block configuration and implicitly the neighborhood structure are updated. This section describes the framework of BBHC and details the implementation.

4.3.1 The Model Based Local Search Framework

Figure 4.1 depicts the two main phases of the model based local search, BBHC optimization. The first one refers to the accumulation of search experience, provided by the repeated hill-climbing. The second phase concerns the exploitation of search experience by linkage learning and building block structure update.

The input of the second phase is the search experience stored in memory. Dependencies are detected and the output consists of an updated building block structure, which enables the first phase to combine new building blocks. In hierarchical problems, modeled after the suggestions of the BBH, the assembling of lower level building blocks leads to the development of higher order ones. Thus the sequence of phases can effectively overcome hierarchical levels successively by discovering and incorporating building block knowledge into the search process.

4.3.2 The Building Block hill-climbing

4.3.3 Linkage detection and building block structure update

As hierarchical problems under study have a nice non-deceptive structure in the building block space, a very simple method for linkage detection is considered.

The clustering of loci in new building blocks is done by searching for bijective mappings.

Due to the transitivity property of bijective mappings, all relevant building blocks are discovered simultaneously. The linkage detection algorithm is presented in Figure 4.4.

All building blocks linked together by a bijective mapping are united into a new building block. The candidate configurations of the new building blocks are extracted from the binary representation of states from the memory. If a building block can not be linked with any other building block it keeps its original place and only its possible configurations are updated in the same manner as for the new building blocks.

4.3.4 The memory size

The proposed model can be summarized by the algorithm presented in Figure 4.5.

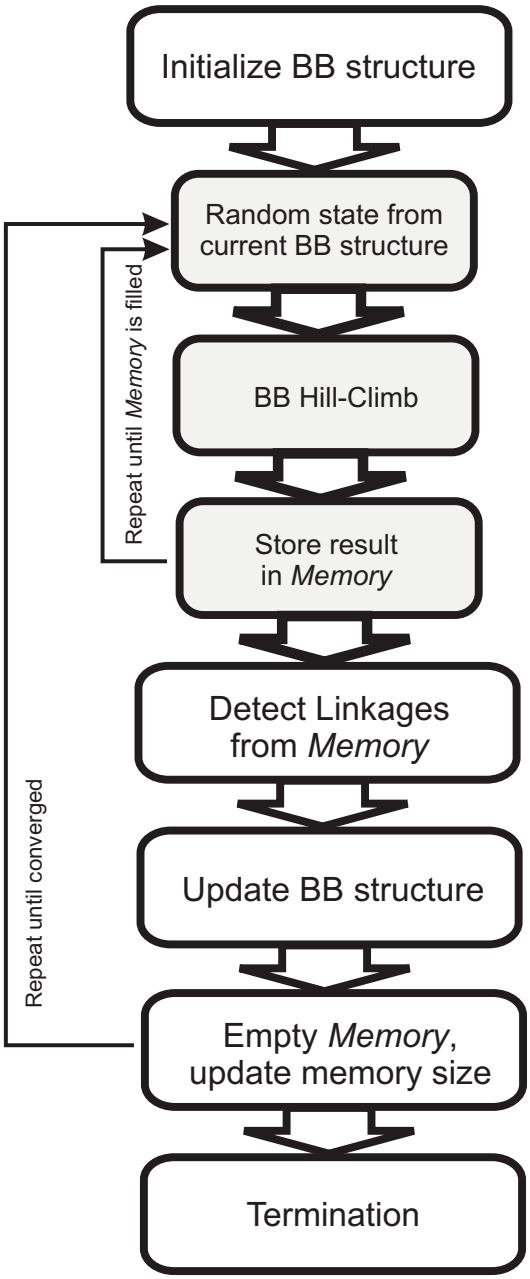


Figure 4.1: The framework of BBHC with the two main phases: accumulation and exploitation of search experience.

4.4 Results

We tested the scalability of BBHC on 128-bit, 256-bit, 512-bit and 1024-bit shuffled hIFF and hXOR problems, respectively on 81-bit, 243-bit and 729-bit shuffled hTrap problem instances. Lower problem sizes were not addressed as they may

be too easy to solve; any conclusion from them may be misleading. For each test suit a total number of 100 independent runs were averaged.

The arithmetic scaling results with the ratio between neighborhood points is presented in Figure 6.1. On the test suites, the proposed method scales up almost linearly with the problem size, with the slope between neighbor points decreasing towards 2 as the problem size doubles on hIFF and hXOR and towards 3 on hTrap as problem size is tripled.

The experimental results were approximated with functions of the form $f(x) = ax^b \cdot log(x)$ where a and b are determined by the least square error method. In Figure 4.7 the log-log scaled plot of the test results and approximation functions are shown. The approximated number of objective function evaluations scales as $O(l^{0.97} \cdot log(l))$ on hIFF and hXOR and $O(l^{0.91} \cdot log(l))$ on hTrap, where l is the problem size. The approximations are very close to the expected linearithmic time. The best results reported till now scale up sub-quadratically with a lower bound of $O(l^{1.5} \cdot log(l))$ (Pelikan, 2005).

The hBOA, one of the best known optimizers that operate via hierarchical decomposition, with hand tuned parameters solves the 256-bit shuffled hIFF in approximately 88000 function evaluations. The BBHC performance on the same test suit is 20666, approximately four times quicker than the hBOA. Due to the linearithmic scaling the BBHC is able to solve the 512-bit version of the same problem approximately twice as fast as the hBOA does the 256-bit one, requiring only 45793 function evaluations on average!

Similarly to other methods like the DSMGA++, the BBHC 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 BBHC was able to detect the perfect problem structure in all runs, due to its more systematic and deterministic approach. The enhanced capability of BBHC 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 building block structures (a balanced binary tree) coincide. For the DSGMA++ the time needed to optimize the two problems differs significantly, being $O(l^{1.84} \cdot log(l))$ for the hIFF and $O(l^{1.96} \cdot log(l))$ on hXOR.

Figure 4.2: Building block configuration of the 8-bit state.

HC(s)

1. Choose randomly an unprocessed building block b_i from $B(s)$;
2. Choose randomly an unprocessed building block configuration $v_j \in V_i$;
3. Set $v(b_i)$ in s to v_j;
4. If the change results in a decrease of the objective function undo the change;
5. If there exists unprocessed building block configuration of V_i then *goto 2*;
6. If there exists unprocessed building block from $BB(s)$ then *goto 1*;

Figure 4.3: The pseudo code of the deterministic, greedy building block search.

4.5 Summary

Preliminary scalability test of the proposed method indicates that BBHC holds not only a quantitative advantage over other methods but also a qualitative one too: it scales linearithmic with the problem size.

BBForm(s,M)

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

Figure 4.4: The linkage detection and new building block forming algorithm, where M is the memory.

BBHC(x, c, b, k) returns *best_state*

1. Initialize the building block knowledge with each single locus from x as a building block;
2. Initialize the memory size:
   $size[M] := c + log_b(length(x))$;
3. Generate a random state s according to the current building block structure knowledge $BB(s)$:
   $s := RandomState(BB(s))$;
4. building block hill-climb from s and store the result in memory: $M := M \bigcup HC(s)$;
5. If the resulted state is better then the best states seen so far, keep the new state:
   $s := best(s, best\_state)$
6. If M is not filled up *goto 3*;
7. Learn linkage from memory and update the building block configuration according to the detected linkages: $BBForm(s, M)$;
8. Empty memory: $M = \emptyset$;
9. Update the memory size:
   $size[M] := c + log_b(\aleph(BB(s)))$;
10. If there was an improvement in the last k epochs and the number of maximum function evaluations was not exceeded *goto 3*;

Figure 4.5: Outline of the hill-climbing enhanced with memory and linkage learning. In steps 3–6 we accumulate the search experience (phase 1) which is exploited in steps 7–9 (phase 2).

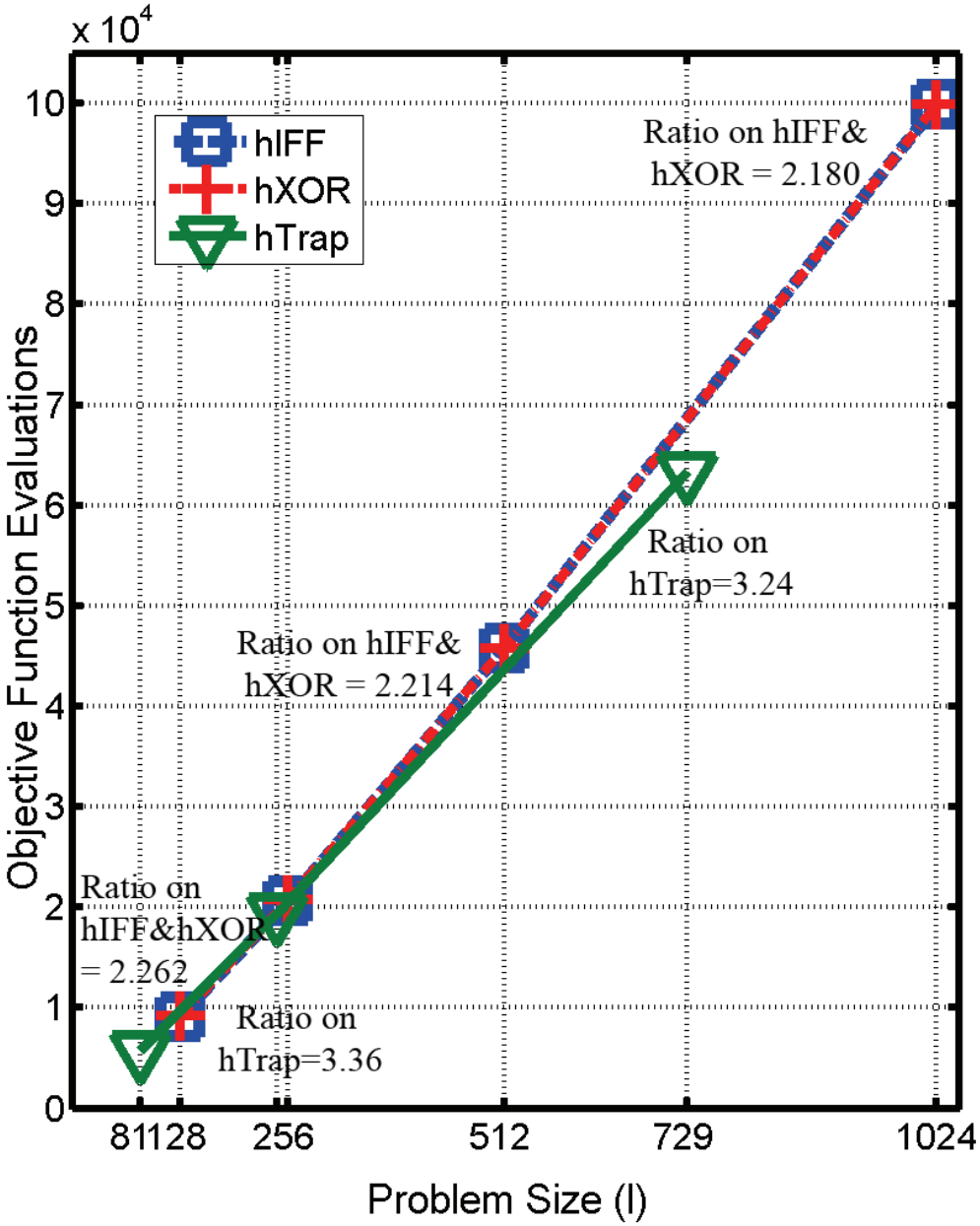


Figure 4.6: Arithmetic scaling of BBHC on hIFF, hXOR and hTrap. The number of function evaluations scales almost linearly. The ratio between neighborhood points is decreasing towards 2 as the problem sizes are doubled in the case of hIFF and hXOR and towards 3 in the case of the hTRap.

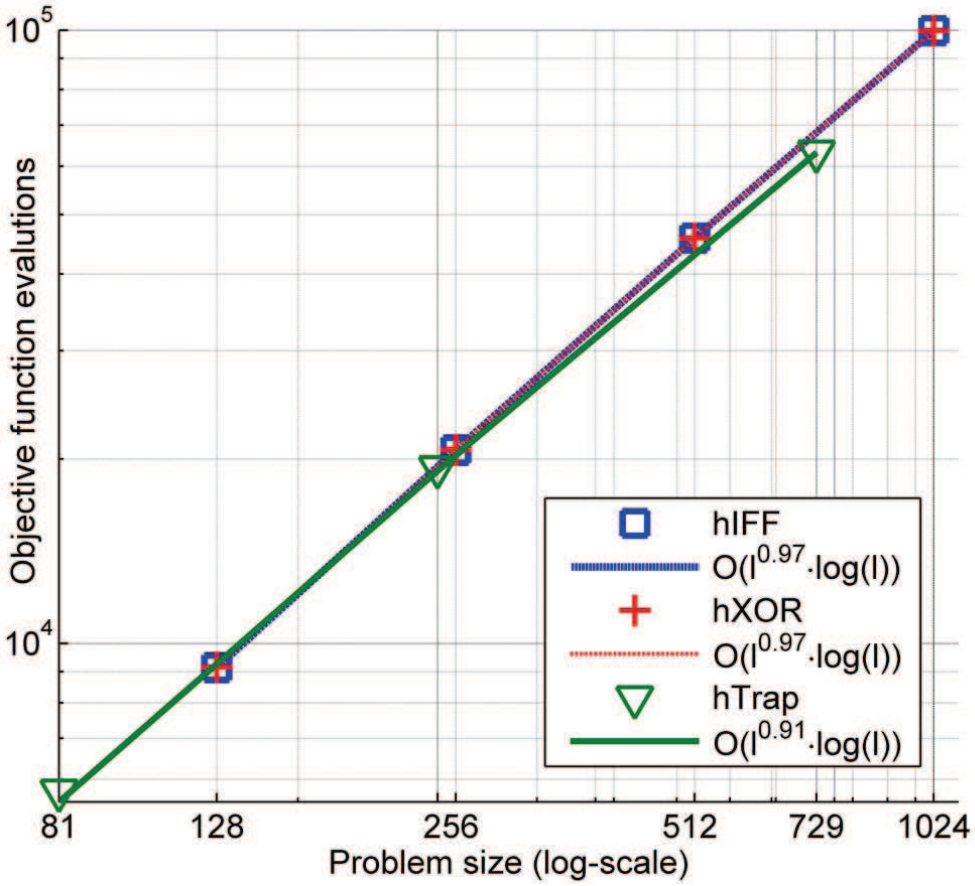


Figure 4.7: The number of function evaluations of BBHC approximately scales as $O(l^{0.97} \cdot log(l))$ on hIFF and hXOR and $O(l^{0.91} \cdot log(l))$ on hTrap, where l is the problem size.

Chapter 5

Overcoming Neutrality and Deceptiveness

In this chapter we present a competent methodology, capable of efficiently detecting and combining *large* modules, even in the case of unfavorable genetic linkage and no intra-block fitness gradient to guide the search or deceptiveness, relying upon the Model Based Local Search framework introduced in the previous chapter. This is achieved by investing the function evaluations in a model based local-search with strong exploratory power and restricting the model building to a relatively small number of semi-converged samples.

5.1 Introduction

5.2 How Much Is the Fish?

5.2.1 Population Sizes and Model Building Cost in PMBGAs

5.3 Test functions

5.3.1 Extended Shuffled Royal Road Function

5.3.2 Hierarchical Massively Multimodal Deceptive Function

5.4 Model Based Macro-Mutation

The search is based on the MBLS framework presented in chapter 4, using a macro mutation operator for the local-search.

5.4.1 Module aware representation

5.4.2 Macro-Mutation Hill-Climber

A method with great exploratory power, the Macro Mutation Hill-Climber (MMHC) had been shown to be a very powerful hill-climbing method, which can outperform GAs even on problems where each building-block corresponds to a deceptive trap function, provided that the problem has a tight linkage (Jones, 1995).

5.4.3 Learning the structure

Learning with Self Organizing Maps

We use a Self Organizing Map (SOM) to detect dependent inputs. SOMs are trained using *unsupervised learning* to produce a two dimensional, discretized representation of the input space of the training samples, called a map (Kohonen, 1982).

The interesting feature of SOMs, exploited here, consist in the fact that the mapping is *topology preserving* and similar inputs tend to have similar weights.

Dependencies are deduced form the internal representation of the SOM based on the heuristic that similar inputs should produce similar patterns in their associated weights i.e dependent inputs have roughly the same values for their weights.

Algorithm 2: Model Based Macro Mutation

Data: $M, V, n_S, z, c_2, @stopping\_cond, \epsilon_1$.

while *not* $@stopping\_cond$ **do**
  $n \leftarrow |M(s)|$;
  /* Phase I */
  for $i = 1, n_S$ **do**
    /* Generate a random state s according to the current building-block knowledge $M(s)$ */
    $s \leftarrow RandomState(M)$;
    /* Apply macro-mutation according to the current model for $c_2 n^2$ evals */
    $s \leftarrow MBMM(s, [M, V], c_2 n^2)$;
    $mem[i] \leftarrow s$;
  /* Phase II */
  $T \leftarrow NormalizeDataRanges(mem)$;
  $F \leftarrow 0_{n \times n}$;
  for $l = 1, z$ **do**
    $net \leftarrow InitializeSOM()$;
    /* Randomly select four samples */
    $S \leftarrow ChoseRandomSamples(T, 4)$;
    $net \leftarrow Train(net, S)$;
    /* Detect possible modules via weight analysis */
    $nm \leftarrow GetLinkages(net.Weights, \epsilon)$;
    /* Update the frequency matrix */
    $F \leftarrow Update(F, nm)$;
  /* Get modules from the frequency matrix */
  $b \leftarrow GetBaseModules(F, \epsilon_1)$;
  /* Merge overlapping modules */
  $b \leftarrow unique(\{b_i = b_i \bigcup b_j \text{ if } b_i \bigcap b_j \neq \emptyset;\ \forall i, j\})$;
  /* Collapse the search space and update the building-block configuration according to the detected modules */
  $[M, V] \leftarrow UpdateModuleKnowledge(b)$;

Noise Filtering

The MBMM is summarized in Algorithm 2.

5.5 Results

The performance of the proposed MBMM method was tested on the $ESRR$ function with even base module sizes from 8 to 16 and on $HMMD$ function of order 6 and 8. On the $ESRR$ function the parameters that reward certain block configurations were set to $r_1 = 1, r_2 = 2, r_3 = 4$. The $HMMD$ functions

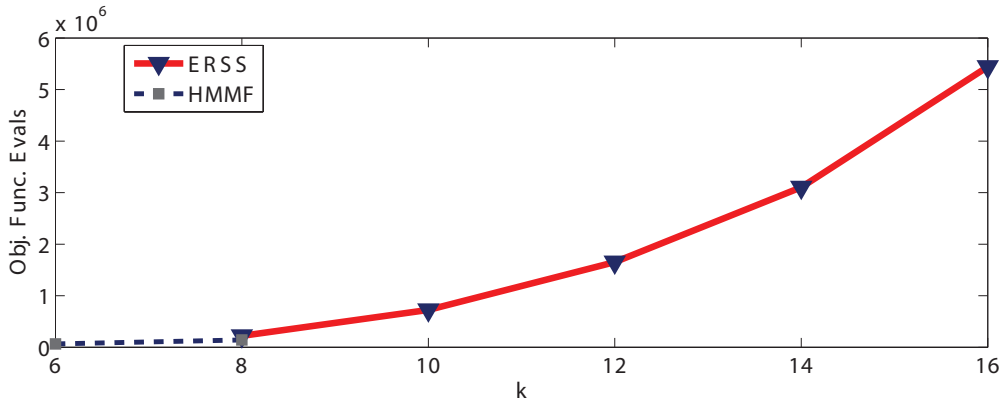


Figure 5.1: Performance and scaling of the MBMM on the proposed test suites.

were used with $r_1 = 1$ and $r_2 = 2$. The ζ threshold for fitness variance in the model based macro-mutation was dynamically determined: using 100 trials, we computed the average fitness improvement δ_+ of the successful mutations working on randomly generated states. A major fitness improvement was regarded as one above this value with more then 50%, thus $\zeta = 1.5 \cdot \delta_+$.

For test suites with block sizes up to 10, a total number of 100 independent runs were averaged. For problems with base module sizes of 12 the number of averaged runs was 30, respectively 10 for block sizes of 14 and 16.

The number of function evaluations used by an epoch of local-search was set to $5n^2$. The method showed a very good behavior, with 100% success rate on every test suite and with a very accurate and prompt model building.

The performance and scalability of the method is depicted in Fig. 5.1. As k grows, random sampling required to discover blocks grows exponentially. Nevertheless, even for k as large as 16, the proposed method, using objective function guidance and biased search for module discovery, is able to find and combine all blocks accurately, within the bounds of 6e6 objective function evaluations.

We repeated our experiments for the *ESRR* test suite, up to block sizes of

k	LS - $c_2 n^{c_1}$ ($c_2 \leq k$)	Succ. rate	Avg. nr. obj. func. evals.
8	$5n^2$	100%	3.442e5
10	$5n^2$	100%	1.425e6
12	$10n^2$	100%	7.547e6
14	$5n^{2.39}$	100%	2.655e7

Table 5.1: Numerical results of the MBMM with the blind version macro-mutation on the *ESRR*.

14, using the simple, blind macro-mutation strategy, without the fitness variance analysis and preliminary model building. Here, whenever a new state is accepted, we only perform a simple greedy search upon this state, in order to discover better solutions, if any, in the close neighborhood of the new state.

The results are summarized in Table 5.1. The first column contains the basic module sizes. The scaling of function evaluations with regard to the block/problem size (remember that $n = k^2$), invested by the local-search in a single run in order to detect correct module settings, is presented in the 2^{nd} column. Column 3 contains the rate of success for each test suite, while the average total number of function evaluations until global optimum is reached, is reported in the last column.

$ESRR$ of size 8 and 10 is easily solved again in all cases, allocating the $5n^2$ function evaluations to the local-search.

Due to the strong exploration performed by the biased model based macro-mutation, neutrality of $ESRR$ the massive multimodality and average case deceptiveness of the $HMMD$ function is overcame, as shown in the results. The method correctly identified one of the global optima and provided an exact model building in all runs. Even if the search space is extremely hard, due to the random shuffling and because of the astronomical number of aleatory placed local optima, the MBMM is able to quickly solve this problem by building and exploiting an accurate problem decomposition.

5.6 Conclusions

Investing the function evaluations into an exploratory local-search can facilitate the discovery of large modules. The method can also decide between the most fit module settings and their most competing schemata. Thus, the total number of samples needs to be just large enough, to make the delimitation of different modules possible with suitable machine learning techniques.

Chapter 6

Large Scale Optimization

The chapter presents principled results demonstrating how the identification and exploitation of variable dependencies by means of Artificial Neural Network powered online model building, combined with the Model Based Local Search, opens the way towards large-scale optimization of hard, non-separable building block problems.

6.1 Introduction

In order to solve non-separable problems up to millions of variables, we need a method that is computationally efficient in terms of model building and also very efficacious in terms of memory usage.

For meeting these desiderates, we consider the extension of the model based local-search presented in (Iclanzan & Dumitrescu, 2007b) with an online learning and model building mechanism. The proposed Online Model Based Local-Search (OMBLS) framework employs an adaptive neighborhood structure which facilitates the operation directly on modules. Nevertheless, this approach does not use a memory to store semi-converged solutions for later analysis, one point is sampled at the time and the search experience is accumulated and information about the problem structure is inferred from a single data structure, resulting in very low memory requirements.

6.2 The Mixed Hierarchical Test Function

To obtain a single, large, scalable test problem which embeds all the test features found in separate suites, we consider the mixing of three standard and well known hierarchical test functions: the hierarchical IFF (Watson et al., 1998), the

hierarchical XOR (Watson & Pollack, 1999) and the hierarchical trap function (Pelikan & Goldberg, 2001).

6.3 Online Model Based Local-Search

6.3.1 Employed Local-Search

We use a simple greedy search among these configurations.

6.3.2 Linkage Learning within OMBLS

A network which seeks to preserve the topological properties of the input space is the Self Organizing Map (SOM) (Kohonen, 1982). The network is trained using *unsupervised learning* to produce a two dimensional, discretized representation of the input space, called a map.

The SOM training algorithm can be very well iteratively updated online with "live" data, directly inputting the results (locally converged states) of the model based local-search, rather than training with samples from a memory.

By analyzing the weights of the network, we are able to decide which of the current variables are linked, but in order to collapse the search space we also need their context-optimal settings. As a consequence, provided with only the variable relationships, for each new composite block the method must search all the possible combinations of sub-modules in the context of randomly generated states and retain the best λ ones.

6.3.3 Collapsing the Search Space

6.3.4 OMBLS Algorithm

Starting from a representation in concordance with the original problem, in a first phase, search experience is accumulated by training the SOM online with the "live" states provided by repeated local-search working on the current representation, which always express the most two fit schemata found at a lower level. The local-search strategy used must be powerful enough to discover fully optimized modules at a single hierarchical level.

After convergence of the network, in the second phase the structure of the input space is inferred from the weights of the network and expressed by variable linkages. Furthermore, an exhaustive search is performed according to the detected linkages, to find the best context-optimal settings for each new module.

Formally the OMBLS is outlined in Algorithm 3.

Algorithm 3: Online Model Based Local-Search

```
   Data: n, n_S, ϵ, @stopping_cond.
   /* Initially each binary variable is a base module          */
 1 for i = 1, n do
 2     M[i][0] ← {0};
 3     M[i][1] ← {1};
 4 while not @stopping_cond do
       /* Phase I                                               */
       /* Build the SOM                                         */
 5     net ← InitializeSOM(n);
 6     for i = 1, n_S do
           /* Generate a random binary state of length n        */
 7         s ← RandomState(n);
           /* Apply module-wise greedy search                   */
 8         s ← GreedySearch(s, M);
           /* Train the network online using vector quantization */
 9         net ← Train(net, s);
       /* Phase II                                              */
       /* Detect possible modules via weight analysis           */
10     nm ← GetLinkages(net.Weights, ϵ);
       /* Identify the two most fit schemata for new modules by
          exhaustive search                                     */
11     CO_set ← RetrieveBestTwoSettings(nm);
       /* Collapse the search space and update the building-block
          configuration according to the detected modules and
          their context-optimal settings                        */
12     n ← |CO_set|;
13     for i = 1, n do
14         if module i is new then
15             M[i][0] ← CO_set[i][0];
16             M[i][1] ← CO_set[i][1];
```

6.4 Run-Time and Scaling of the OMBLS

Assuming that the used machine learning technique successfully detects the correct dependencies, the global convergence of the OMBLS on the studied test suite can be proven, by showing that there is a path towards global optima on each component, easily followed by the method.

In the case of the module-wise greedy search, at one hierarchical level, the number of objective function evaluations is in concordance with the number of modules l, the number of context-optimal setting $\lambda = 2$ for each module and the

number of epochs used to feed the network n_S:

$$T_{LS} = 2 \cdot n_S \cdot l \tag{6.1}$$

The search for the context-optimal settings will take a number of objective function evaluations exponential in the size (denoted by k_i) of the newly discovered modules M' and the number of context-optimal settings:

$$T_{COS} = \sum_{i=1}^{|M'|} 2^{k_i} \tag{6.2}$$

For p hierarchical levels, the upper bound is given by the summation of the model based local-search T_{LS} and the search for context-optimal settings T_{COS} on each hierarchical level.

$$T = \sum_{i=1}^{p} (T_{LS} + T_{COS}) \tag{6.3}$$

As we know that the module sizes on hIFF and hXOR equal $k_1 = 2$ and for hTrap the module size is $k_2 = 3$, we got the following upper bound on hMix for p hierarchical levels:

$$T_{hMix_p} = 2 \cdot \sum_{\substack{l=k_1 \\ l=l*k_1}}^{k_1^p} (2 \cdot n_S \cdot l + \frac{l}{k_1} \cdot 2^{k_1}) + \sum_{\substack{l=k_2 \\ l=l*k_2}}^{k_2^p} (2 \cdot n_S \cdot l + \frac{l}{k_2} \cdot 2^{k_2}) \tag{6.4}$$

To empirically confirm this result and to test the efficiency of the SOM based online linkage detection technique, the scalability of the OMBLS have been tested on hMix with $p = \{4, 6, 8, 10, 11\}$ hierarchical levels, with the resulting problem sizes $n = \{113, 857, 7073, 61097, 181243\}$.

The method found one of the four global optima in all cases confirming the efficiency of the online SOM based linkage learning. The scaling of the method on hMix is presented in Fig. 6.1. The experimental result was approximated with a function of the form $f(x) = ax^b \cdot log_2(x)$ where a and b are determined by the least square error method. As depicted, OMBLS scales on hMix approximately as $\theta(x^{0.909} \cdot log_2(x))$, where x is the problem size.

In our case, as the cost of the greedy search is linear in the number of modules, from Eq. (6.4) results a very efficient sub-linearithmic running time, confirmed empirically by our experiments.

The memory requirement of the OMBLS is very low, being linear in the problem size.

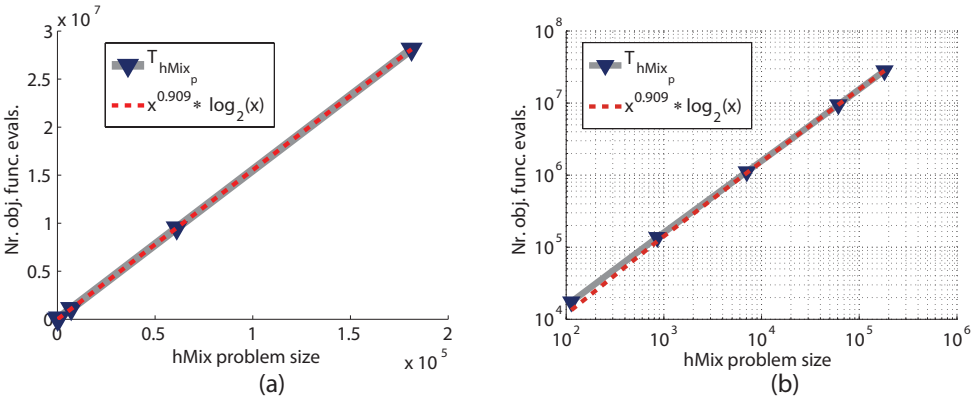


Figure 6.1: Sub-linearithmic scaling of OMBLS with module-wise greedy local-search strategy on hMix: (a) arithmetic plot; (b) logarithmic plot.

6.5 Summary

OMBLS opens the way towards large-scale optimization of hard, non-separable building-block problems. Further work will focus on probabilistic recoding of the information ("soft" chunking) in order to tackle problems with more complicated structure. Another line of research will focus on the parallelization of the proposed framework: instead of sequential epochs, several model based local-searches can be run concomitantly on different computational units; model building can be also greatly parallelized as the search for context-optimal settings for each building-block, respectively the pairwise distance computation of different modules can be run in parallel.

Chapter 7

Graph Clustering Based Model Building

The chapter describes a new unsupervised model induction strategy built upon a maximum flow graph clustering technique. The new approach offers a model evaluation free, fast, scalable, easily parallelizable method, capable of complex dependence structure induction. The method can be used to infer different classes of probabilistic models.

7.1 Introduction

As already disscussed in Chapter 3, model building in EDAs can be computationally very expensive.

In this chapter we further explore the confluence between clustering algorithms and EDAs, where graph clustering algorithms are applied to pairwise interaction statistic matrices to reveal dependency structures. We term this class of methods as Graph Clustering assisted EDAs (GCEDAs). We are especially interested in finding efficient clustering algorithms allowing the induction of various probabilistic model classes.

Our proposed approach has the advantage of being capable of automatically delivering a dependency structure, without the need for a model search and costly goodness-of-fit evaluations.

7.2 Preliminaries

7.2.1 Graph Clustering assisted EDAs

Good probabilistic models describe which variables interact and how. Model discriminative metrics, used in multivariate EDAs to guide the incremental model building, measure variable interactions also quantifying for over fitting.

7.2.2 Graph clustering paradigm, stochastic matrices and flows

Maximum flow clustering algorithms rely on the following core idea: by simulating a special flow within a graph, which promotes flow where the current is strong, and reduces flow where the current is weak will reveal the cluster structure within the graph, as the flow across borders between different groups diminish with time, while it increases within the group.

7.2.3 Markov Clustering Algorithm

The MCL algorithm van Dongen (2000a) is a fast and scalable unsupervised graph clustering algorithm, based on simulation of stochastic flow in graphs. It offers several advantages, like a simple, elegant mathematical formulation, robustness to topological noise Brohée & van Helden (2006), support for easy paralellization and adaptation via a simple parameter enables the obtaining of clusters of different granularities.

MCL iteratively simulates random walks within a graph by applying two operators called expansion and inflation, until convergence occurs. At the end of each inflation step a pruning step is also performed, in order to reduce the computational complexity by keeping M sparse.

7.2.4 Interpretation of MCL clustering as dependency models

MCL iterants M_t are generally diagonally positive semi-definite matrices. Using the property that minors of a diagonally positive semi-definite matrix are non-negative, in van Dongen (2000b) it is shown that M_t-s have a structural property which associates a directed acyclic graph (DAG) with each of them. These DAGs generalize the star graphs associated with the MCL limits.

We present several approaches on how the information from MCL iterants can be conveyed in dependency models able to represent and exploit linkages.

In the *first* approach, the DAGs represented by M_t-s are directly used for defining the structure for a Bayesian network, with the edges representing the conditional dependencies between variables. Then, the parameters, which consist of the conditional probabilities of each variable given the variables that this variable depends on, are extracted from the data in the same way as in BOA Pelikan et al. (1999) or EBNA Etxeberria & Larranaga (1999). The obtained

Bayesian network will encode the joint probability distribution of the variables and can be used to sample the next generation. This approach presents two small impediments. First, one has to decide from which M_t to construct and use the Bayesian network, thus a few model evaluations against the data still have to be computed. The second issue relates to the rare occasions where a MCL iterant contains cycles. Before performing the parameter extraction, these cycles must be detected and eliminated.

In a *second* approach, DAGs are interpreted as clusters by taking as cores all end nodes (sinks) from the DAG, and by attaching to each core all the nodes that reach it with a flow amount greater than a threshold. This procedure may result in clusters containing overlap. The extracted linkages can be used to perform building block wise crossover like in DSMGA Yu et al. (2003) or DSMGA++ Yu & Goldberg (2006) or they can be used to build overlapping linkage model based probability distributions.

The *third* approach is the cheapest one, as it deals only with the last iterant, when the stochastic flow matrix M is completely converged. Here, the nodes have found one "attractor" node to which all of their flow is directed, corresponding to only one non-zero entry per column in M. Nodes sharing the same "attractor" node are grouped in clusters. This approach is suitable for modeling non-overlapping building blocks, by building marginal product models as in eCGA. Harik (1999).

Excepting the first approach, the dependency structure inferring is completely autonomous, as its does not need to check the model fit with regard to the data.

7.3 MCL assisted EDA

Wishing to present an EDA with unsupervised model building, capable of modeling complicated variable interactions, we employ the second interpretation of the MCL iterants to obtain a overlapping linkage model based probabilistic model. We name this algorithm Markov Clustering EDA (MCEDA). The details of the algorithm are presented in the followings.

7.3.1 Global statistics extraction

In this chapter, the degree of pairwise dependency between variables is calculated using sampled mutual information between two variables and record into an adjacency matrix A, which will be the input of the graph clustering algorithm.

The transformation of A in a stochastic Markov matrix is handled by the MCL algorithm by normalization of the columns to sum up to 1.

$$M(i,j) = \frac{A(i,j)}{\sum_{l=1}^{n} A(l,j)} \tag{7.1}$$

Function ExtractBBs(Mlist) returns BBlist

```
1 BBlist ← ∅;
2 //each iterant from the MCL algorithm is processed
3 foreach M_t from Mlist do
4     //for all nodes find significant incoming flows
5     for i ← 1 to size(M_t) do
6         pBB ← find(M_t(i,:) > F_min(i));
7         if length(pBB) > 1 then
8             BBlist ← BBlist ⋃ pBB;
9 BBlist ← unique(BBlist);
```

7.3.2 The Overlapping Linkage Model (OLM)

In MCEDA, the multivariate variable interactions are modeled with the use of overlapping linkage models (OLMs), which closely resembles the marginal product model adopted by the eCGA Harik (1999). The difference is that OLM models subsets of variables jointly as clusters, allowing overlaps, in contrast with partitions, which always divides the variables in collectively exhaustive and mutually exclusive blocks. The clusters can naturally represent building blocks, providing a direct linkage map of the variables, thus we will use this terms interchangeably in the context of OLMs. Clusters together with the marginal distributions over them form the OLMs.

7.3.3 Dependency structure building and sampling

The clusters that form the basis of the OLMs are extracted from the iterants M_t of the MCL algorithm.

For each node a potential building block is formed, by grouping together all nodes that reach it with a flow amount greater than a threshold F_{min}. Basic clusters of size 1 are only allowed, if the described single position is not contained in any other cluster. After all iterants have been processed, the procedure returns the unique entries of the potential building block list. This sorting must be performed, as the same cluster may be detected several times from different iterants, or even from the same M_t in the rare cases when it contains cycles.

The building block extraction is depicted in Function `ExtractBBs`.

After the building blocks are determined, their probability distribution is estimated by simply counting the frequencies in the data.

7.3.4 The Markov Clustering EDA

Algorithm 4 summarizes the structure and workings of the method.

Algorithm 4: The Markov Clustering EDA

```
pop ← RandomInit();
repeat
    ps ← Selection(pop); //select promising solutions
    {ps ← ReduceEntropy(ps)}; //optionally reduce entropy by LS
    A ← MutualInformation(ps); //extract global statistics
    Mlist ← MCL(A); //apply graph clustering
    BBlist ← ExtractBBs(Mlist); //extract dependency structure
    freq ← FrequencyCount(BBlist, ps); //compute marginal
    probabilities
    olm ← BuildMPM(BBlist, freq); //combine results into a OLM
    pop ← Sample(olm); //generate a new population using the model
until convergence criteria is met;
```

7.4 Experiments

In this chapter the class of additively decomposable functions (ADFs) with deceptive trap subproblems is considered, a test bed which is widely used in the literature as benchmarking problemsPelikan et al. (1999); Yu et al. (2005); Correa & Shapiro (2006).

7.4.1 Test functions

The *concatenated trap-5* Pelikan et al. (1999) is an ADF based on unitation (number of ones from a binary string) measures, exhibiting a single global optimum in the string formed exclusively from ones. Non-separability can be introduced by applying a fixed length, circular overlapping scheme between the trap-5 functions Yu et al. (2005). For example, for a problem with 3 subproblems and overlap length $l = 2$, the fitness is given by $trap5(y1y2y3y4y5) + trap5(y4y5y6y7y8) + trap5(y7y8y9y1y2)$, where yi is a random permutation of the variables xi, meant to break the tight linkage. Every building block shares $2l$ variables, l with each of its two neighbor.

7.4.2 Numerical results

Experiments are performed with the simple concatenated trap-5 function without overlap (denoted by ctf5o0), with overlap 1 (ctf5o1) and overlap 2 (ctf5o2). In order to test the scalability, for each ADF the number of subproblems k is scaled from 6 to 18 by increments of 3, resulting in various problem sizes up to 90 variables.

Figure 7.1 b) presents the scaling of the methods for the different problem types and sizes. The results show a similar scaling of the two methods. MCEDA

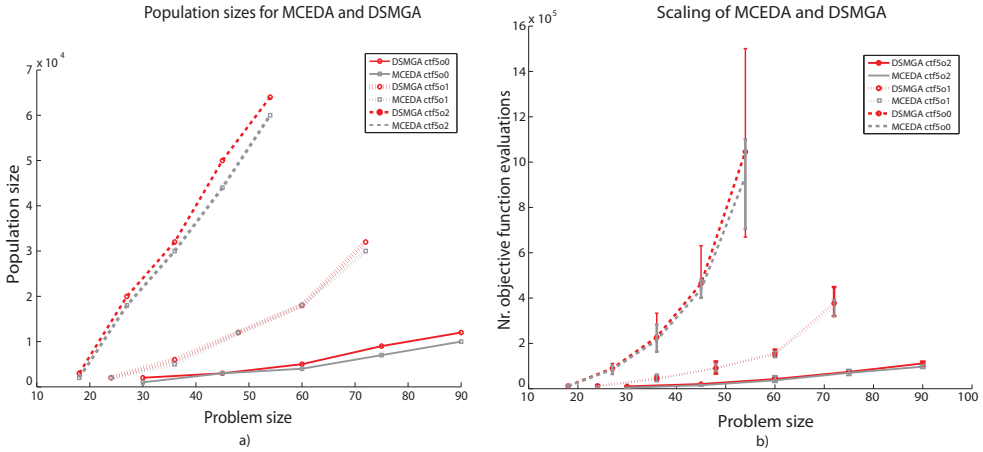


Figure 7.1: The scaling of MCEDA and DSMGA on the test problems.

uses slightly fewer objective function evaluations and works with smaller population sizes than the DSMGA. The performance difference is most likely explained by the following two factors:

- DSMGA uses a crisp value when dealing with variable interactions. The mutual information matrix is transformed in a binary matrix according to a threshold, where two variables are considered fully interacting or completely independent. In contrast, the MCEDA works directly with the normalized mutual information values, which can describe more nuanced, weighted levels or interactions, facilitating the earlier discovery of better models.

- The MCEDA has a better diversity maintenance mechanism as a higher number of building-blocks are extracted due to the usage of early iterants of the MCL process. Furthermore, the method samples according to the exactly observed frequencies in the data. DSMGA may confront the hitchhiking phenomena Mitchell & Holland (1993) where some low fitness alleles are promoted together with high-quality building-blocks in above average individuals and the right crossover must be performed to eliminate them.

The MCL graph clustering is much faster than the MDL based clustering used by DSMGA, having a worst case complexity $O(nk^2)$ van Dongen (2000a), where k is the pruning factor (at most how many non-zero entries will be in a row of the stochastic matrix - a very small number in practice, $k << n$). Clustering of 5000 nodes by the MCL takes only a few seconds, compared with minutes, in the case of DSMGA model-building.

7.5 Summary

Graph clustering offers a broad and flexible set of opportunities for model building as several dependency structure types can be inferred from the results. They offer the possibility to entirely bypass the model fit-to-data evaluations or they can be applied in a controlled mode, where they only guide the search. In the second scenario, model evaluations are still applied, albeit much more infrequently, with the role to guard against over fitting, discriminate between potential model candidates.

Chapter 8

Crossover Utility

Historically, attempts to capture the topological fitness landscape features which exemplify this intuitively straight-forward process, have been mostly unsuccessful. Population-based recombinative methods had been repeatedly outperformed on the special designed abstract test suites, by different variants of mutation-based algorithms.

Departing from the Building Block Hypothesis, this chapter's work seek to exemplify the utility of crossover from a different point of view, emphasizing the creative potential of the crossover operator. We design a special class of abstract test suites, called Trident functions, which exploits the ability of modern GA to mix good but *significantly different* solutions. This approach has been so far neglected as it is widely believed that disruption caused by mating individuals that are too dissimilar may be harmful.

However, hybridizing different designs induces a complex neighborhood structure unattainable by trajectory-based methods which can conceal novel solutions. In this chapter we demonstrate that the proposed class of problems can be solved efficiently only by population-based panmictic recombinative methods, employing diversity maintaining mechanisms.

8.1 Introduction

8.2 Historical Background

8.3 Hybridization of Differences

We reason that in order to defeat hill-climbers, problems must contain a degree of deception, which can not be overcome by a neighborhood operator induced

by one point in the search space. This of course will hinder GAs performance also, as the mutation works in the neighborhood of one individual and short-term selection may favor deceptive search paths. However, EAs possess a great asset by having a more complex neighborhood structure generated by the recombination operator, which takes into account at least two individuals. This may help the methods to escape the local optima and overcome deceptiveness.

The problem representation together with the neighborhood structure defines the search landscape. We argue that there are problems where only search landscapes transformed by crossover may be efficiently exploitable. In the followings we give an example for such a class of problems called the Trident functions (TF).

8.3.1 The Trident Function

TF accepts bit strings of the length $2k$ where $k \geq 2$ and uses a function of unitation (which depends on the number of ones in a bit string, and not on their positions) as underlying structure:

$$base(x) = \|2 \cdot u(x) - |x|\| \tag{8.1}$$

where $u(x)$ is the unitary of x (the number of ones) and $|x|$ is the length of x.

The base function has its minimum in 0 which is generated by strings with an equal number of 1's and 0's: $u(x) = |x| - u(x)$. The maximum is attained by strings formed by all 1's or all 0's with a corresponding value of $|x|$.

The next component of the TF is a contribution function which rewards *certain configurations* of strings that have an equal number of 1's and 0's. Let $L = x_1, x_2, \ldots, x_{\frac{n}{2}}$ be the first half of the binary string x of length n and $R = x_{\frac{n}{2}+1}, x_{\frac{n}{2}+2}, \ldots, x_n$ the second one. Then, we define the contribution function based on the exclusive OR (XOR) relation:

$$contribution(x) = \begin{cases} 2 \cdot |x| & \text{, if } L = \bar{R}; \\ 0 & \text{, otherwise.} \end{cases} \tag{8.2}$$

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

Please note that the contribution function does not have a basin of attraction; it rewards fully an input or it does not reward it at all. Finding the maxima of such a function is equivalent to the *needle in the haystack* problem. As there are no better search methods for this class of function than the random-search, these function are also resistant to biased mutation-based search.

The TF is defined as the sum of the base and the contribution function:

$$trident(x) = base(x) + contribution(x) \tag{8.3}$$

Figure 8.1 presents the graphical interpretation of the Trident function.

TF has its maximum in the points rewarded by the contribution function. Here it takes the value $2 \cdot |x|$ as the base function in these points attends the minimum 0.

TF is very hard for mutation-based algorithms because the base function leads away the search from the region where global optima lay. Even if a random state is generated with equal number of 1's and 0's, it is very unlikely for large problem instances that the contribution function will reward that string. If the algorithm does a biased search, it will be immediately drawn away from the minimum of the base function, towards regions with higher base function fitness.

The TF can defeat macromutation hill-climbers also, as local and global optima are very distant in the Hamming space. The chance of jumping from local optima to a global one is minimal as $\frac{n}{2}$ bits must be changed simultaneously.

Also, there are no "hidden" structures which could be easily exploited. The "building-blocks" L and R are rewarded if and only if their context i.e. the counterpart half of the string is compatible. As TFs have $2^{\frac{n}{2}}$ global solutions, the probability of this happening for randomly generated strings is $P_{hit} = \frac{2^{\frac{n}{2}}}{2^n} = \frac{1}{2^{\frac{n}{2}}}$.

What about GAs? Global optima can be found quite easily if the GA is mixing good but different solutions. Let us take the example where $n = 8$ and we have two strings at each local optimum: $s_1 = 00000000$ and $s_2 = 11111111$. The one-point crossover between s_1 and s_2 will produce the optimal strings $s_3 = 00001111$ and $s_4 = 11110000$ with the probability $P = \frac{1}{n-1} = \frac{1}{7}$. When using two-point crossover, we have $\frac{n}{2} - 1 = 3$ favorable cases. The favorable crossing points pairs are $\{(1, 7), (2, 6), (3, 5)\}$. Optimal strings may not result only from the breeding of individuals located at local optima. For example, the one-point crossover between $s_5 = 00100001$ and $s_6 = 11111101$ between loci 4 and 5 will also produce an optimal solution $s_7 = 00101101$. The important aspect is to combine *different* candidate solutions.

The TF portray the problems where several highly different good solutions exist, and hybridizing these solutions *may* result in a completely new, valuable design. Even if crossover does not produce above average individuals on a regular basis, it may create occasionally an exceptional organism. Thus, crossover has a generative potential which we believe should not be neglected by restricting the recombination to genotypically similar individuals.

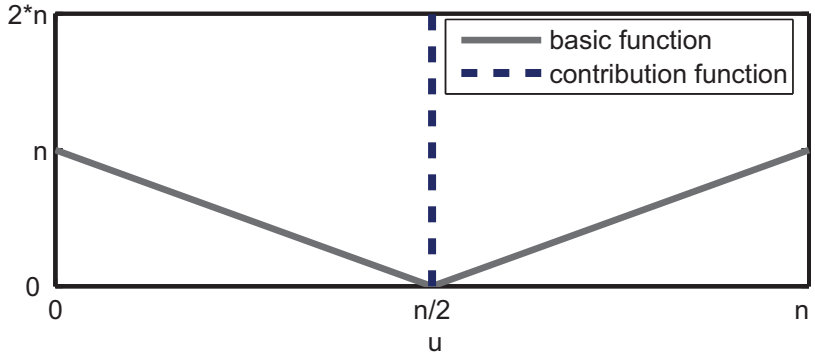


Figure 8.1: The Trident Function. u is the unitary of the input string. The base function is deceptive, leading away the search from the area which contains global optima. The contribution function has no basin of attraction so its maximum is very hard to detect. Note that the contribution function does not reward all strings with $u = \frac{n}{2}$; only special configurations are rated.

8.3.2 Natural Metaphor

8.4 Results

8.4.1 Random Mutation Hill-Climber

8.4.2 Macro Mutation Hill-Climber

8.4.3 Simple Genetic Algorithm

8.4.4 Deterministic Crowding

8.4.5 Numerical Results

The numerical results of the experiments are summarized in Table 8.1. On the 64 and 128-bit versions of the TF, only the results of the DC are reported as the hill-climbers and the SGA failed in all runs on these suites.

As expected, the worst behavior on the TFs was shown by the RMHC. Even for the very easy 16-bit version of the problem, the success rate is only 85%. Similar to the other hill-climber, the MMHC, solutions are found at very high cost and only due to the random restart mechanism. The number of function evaluations required to identify optima, exceeded by orders of magnitude the amount that would be required by random-search. As the problem size increases, becoming unaddressable by random sampling, the hill-climbers fail in all runs due the deceptive nature of the TF.

When SGA succeeded, its performance was the fastest, being much better then it would be required by random-search. This shows that even simple recombinative algorithms have the potential to exploit the features of the TF landscape.

Table 8.1: Performance of the studied algorithms on the TFs. Column "Succ. rate" contains the number of successful runs where the methods find global optima. Column "Avg. nr." contains the number of average function evaluations needed to find the global optima and "Max. nr." counts the maximum number of evaluations needed, provided that all runs were successful. In the case of the population-based methods, column "Nr. opt." contains the average number of different optima within the correctly converged population.

TF size	**16**			
	Succ. rate	Avg. nr.	Max. nr	Nr. opt.
RMHC	**85%**	400020	-	/
MMHC	**100%**	4213	33528	/
SGA	**100%**	241	1390	1.46
DC	**100%**	250	2111	23.94

TF size	**32**			
	Succ. rate	Avg. nr.	Max. nr.	Nr. opt.
RMHC	**3%**	337136	-	/
MMHC	**37%**	471018	-	/
SGA	**25%**	8435	-	1.56
DC	**100%**	15771	23883	6.69

TF size	**64**			
	Succ. rate	Avg. nr.	Max. nr	Nr. opt.
DC	**100%**	46816	61366	6.02

TF size	**128**			
	Succ. rate	Avg. nr.	Max. nr.	Nr. opt.
DC	**100%**	112557	157890	4.32

However, as the problem size increases, the SGA fails to find optima due to the lack of improper initial sampling. Therefore, the population is quickly shifted towards the basin of attraction of a single local optimum. A significant increase of the population could address this problem but then success would come at high costs.

The only competent algorithm on the TFs was the DC. It succeeded in absolutely all runs, being able to identify global optima within a maximum of 16% of the allowed function evaluations. In all cases, several optimum points were detected. The success of the algorithm derives from its diversity maintaining mechanism combined with the panmictic population.

We once again emphasize the importance of the capability to mix different

designs; only then recombination can become creative. An algorithm with diversity maintaining mechanism, but with crossover restricted to similar individuals, would also fail on the TFs.

8.5 Summary

The TFs may represent design problems where several good, locally optimal drafts are easy to find, dominating the search space (deception) and the real good designs result from the hybridization of different drafts. Furthermore, the complex layouts defining the best solution only emerge in "reactive regions" where the correct particular features appear simultaneously; there is no sequence of improving designs to these solutions (needle in the haystack). Nevertheless, crossover possesses the creative potential i.e the more complex neighborhood structure, which enables it to identify these solutions by mixing features from different drafts, until the correct configuration is detected.

Chapter 9

Strongly Non Convergent, Cooperative, Specialized Search

Many current EA suffer from a tendency to prematurely lose their capability to incorporate new genetic material, resulting in a stagnation in suboptimal points. To successfully apply these methods on increasingly complex problems, the ability to generate useful variations leading to continuous improvements is vital. Nevertheless, there is a major difficulty in finding computational extensions to the evolutionary paradigm that ensures a continuous emergence of new qualitative solutions, as the essence of the Darwinian paradigm – the natural selection – acts as a stabilizing force, keeping the population into an evolutionary equilibria.

In this chapter we propose a new approach, replacing the survival of the fittest paradigm with a cooperative framework, where individuals are highly specialized on different exploring and exploitive strategies. This results in a highly efficient, non-convergent, sustainable search process, where new optima emerge continually.

9.1 Introduction

A major problem experienced when running EAs on complex, high dimensional problems is that as selection pressure increases, the population tends to converges to a local optima as no further improvements can be made as the method lost its ability to discover and incorporate new genetic material.

Algorithm 5: Exploring individuals

input : A state s and a set of exploring strategies p_{ES}
output: A new state

```
1 begin
    // Choose probabilistically an exploring strategy
2   @ES ← ProbabilisticallyChoose(p_ES);
    // Explore. The method may or may not take use of the
       provided seed
3   s ← @ES(s);
    // Return the result
4   report s;
```

9.2 Cooperative Paradigm

EAs mimic the complex schemes involved in the transmission of biological information in dynamic and complex ecosystems. However, as applications of EAs mostly consider fixed fitness landscapes, a major criticism of EAs is that biological metaphors may be unnecessarily complex and not even highly efficient.

the performance of EAs, especially in the framework of static optimization. A cooperative paradigm can help to alleviate these phenomena.

9.2.1 Selection

Natural Selection is not a Strong Optimizing Force

Selection is a Stabilizing Force

9.2.2 Cooperative Search

Every metaheuristic approach should be designed with the aim of effectively and efficiently exploring a search space (Blum & Roli, 2003).

Consequently, we propose a framework where exploration and exploitation are regarded as completely separated but complementary subtask of the search process and their proper solving is targeted by different techniques and *specialized* individuals.

Exploration

The search may contain several, different exploring techniques, which are chosen and applied with a certain, fixed or adaptive, probability in each epoch as depicted in Algorithm 5.

The role of exploring individuals is to locate above average and unvisited region of the search space. The first goal can be achieved by simple broad sampling.

Algorithm 6: Exploiting individuals

input : A starting state s and a set of exploiting strategies p_{IS}
output: State after exploitation
Data: s_{old}
`// Original state of the individual`
1 **begin**
  `// Choose probabilistically an exploiting strategy`
2 @IS $\leftarrow$ `ProbabilisticallyChoose(`p_{IS}`);`
  `// Apply lamarckian operator starting from` s
3 $s' \leftarrow$ @IS(s);
  `// Deterministic crossover between new and old state`
4 $s' \leftarrow$ `Merge(`s', s_{old}`);`
5 report s';

The second objective requires techniques that divide the search space or learn its topology to be able to favor unvisited regions.

The starting states and regions proposed by the exploring individuals are recorded in a data structure that provides the communication channel, information sharing with the exploiting individuals.

If the computation time of the objective function requires many resources, the exploration subtask is highly tolerant to the computing of an approximate fitness value, as it has to decide only which states are above average. This can be answered without knowing the exact value of the sampled states. A proxy fitness function, which is positively correlated with the original function, but much easier to compute may be enough.

Exploitation

Exploiting individuals apply local-search strategies to quickly identify high quality solutions, starting from states reported by the other group via the shared data structure. Different local-search strategies are applied in the same probabilistic manner as the in the case of exploring individuals as shown in Algorithm 6.

The presented general cooperative framework contains many degrees of freedom. Next subsection present the method used in our experiments.

An Instance of the Proposed Paradigm

To make a strong case that a simple non-convergent cooperative behavior is enough to qualitatively advance the search performance, we do not use the possibilities offered by the framework that can intelligently bias the search by performing some sort of analysis.

Algorithm 7: Deterministic crossover

input : Two states s_1 and s_2
output: Deterministically improved s_1

```
1 begin
2     for i=1:length(s_1) do
3         if s_1[i] == s_2[i] then
4             continue;
5         s' ← s_1;
6         s'[i] ← s_2[i];
7         if s' better than s_1 then
8             s_1 ← s';
9     return s_1;
```

Algorithm 8: Cooperative, specialized search

Data: $pop, pop\_size, p_{ES}, p_{IS}$,@stopping_cond.

```
1 begin
2     pop ← InitPop();
3     while not @stopping_cond() do
4         parfor i = 1, pop_size do
              // Apply Alg. 5 to explore
5             s ← Explore(pop[i], p_ES);
              // Apply Alg. 6 to exploit
6             s ← Exploit(s, p_IS);
7             if s better than pop[i] then
8                 pop[i] ← s;
          // Synchronization
9         pop ← MultiParentCrossover(pop);
```

We do not implement any bias towards unvisited regions, therefore improvements over the time line of the search can not be accounted to biased sampling towards unexploited regions.

As depicted in Algorithm 8, simply exploration and exploitation are applied in a sequential manner. This corresponds to a FIFO list used for communication between exploring and exploiting individuals.

In the synchronization procedure (Algorithm 8, line 9) a deterministic multi-parent crossover is used. This is very similar to Algorithm 7 with the difference that here a systematic greedy search is performed in the entire gene pool of the exploiting subpopulation. The resulting super individual replaces the individual with highest fitness from the subpopulation.

Exploring individuals apply one of the following techniques with fixed prob-

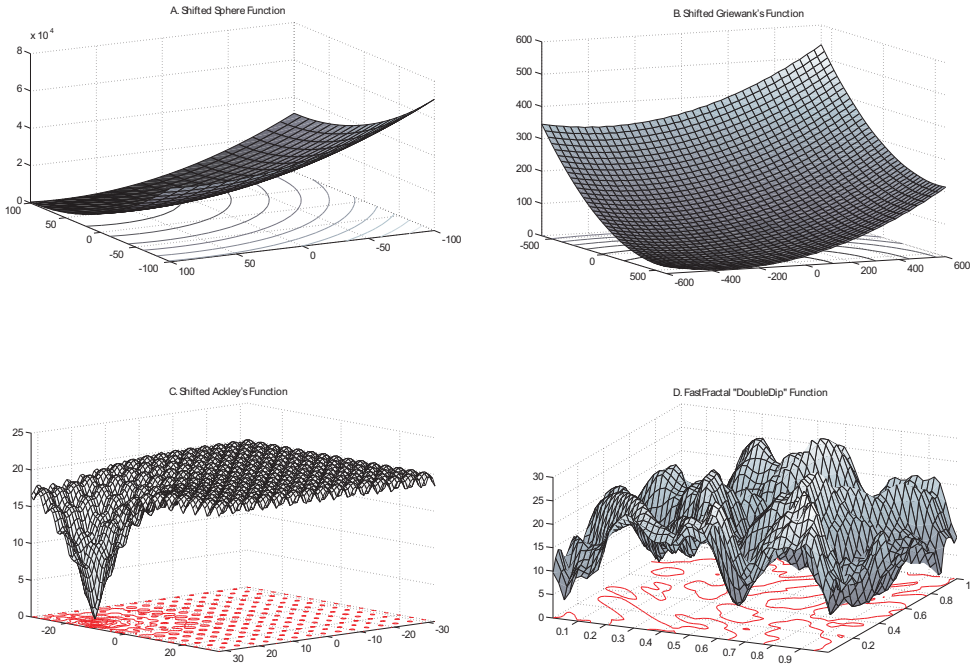


Figure 9.1: 3D map of the functions for two dimensions.

ability: (i) random sampling within the bounds of b_{rs} objective function evaluations and (ii) quick simulated annealing (Kirkpatrick et al., 1983) to identify a good starting point for exploitation. The idea behind the second method is to quickly freeze (by using a large cooling factor) the sampling point in the basin of attraction of a local optima.

Exploiting individuals randomly choose with equal probability between using (i) pattern search (Torczon, 1997) and (ii) Shor r-Algorithm (Shor et al., 1985). These techniques are known as good local search algorithms for non-smooth, non-differentiable functions.

9.2.3 Conceptual Relation to Other Metaheuristics and Paradigms

9.3 Experimental Setup

In (Tang et al., 2007) benchmark functions are given for high-dimensional optimization, providing means for systematic evaluation and comparison of the scalability of different search techniques and paradigms. All of them are scalable for any size of dimension. These problems were used at the IEEE CEC 2008 Competition on Large Scale Global Optimization and are regarded by the community as an adequate test suite due to the many feature and function properties covered.

Considered functions are of the form $F(x) : [a, b]^D \to \mathbb{R}$, where $a, b \in \mathbb{R}$ are the bounds of the domain of definition and $x = [x_1, x_2, \ldots, x_D]$ usually operate on $z = x - o$ where $o = [o_1, o_2, \ldots, o_D]$ is the shifted global optimum.

Our results are reported on four out of the seven functions but our finding extend to the other functions also. The mathematical formulas and properties of the functions used are described in the following subsections. Their 3D plot is presented in Fig. 9.1.

9.3.1 Shifted Sphere Function

9.3.2 Shifted Griewanks Function

9.3.3 Shifted Ackleys Function

9.3.4 FastFractal "DoubleDip" Function

9.3.5 Parameterization of the Methods

In the exploring phase, the algorithm uses 1000 objective function evaluations with random search, respectively 3500 with the quick simulated annealing to locate an above average point. The simulated annealing use an exponential cooling schedule with a cooling factor of 0.7.

Exploitation methods used:

- Pattern search
- Shor r-algorithm
- Both known as good local search algorithms on non-smooth, non-differentiable functions.

The exploiting local-search procedures run for a maximum of 500 iterations each time they are used.

For comparison we use a genetic algorithm with roulette wheel selection a population size of 1000, crossover rate of 0.8 and Gaussian mutation. This population size showed better results like a larger population of 10000 individuals, providing a good compromise between the number of individuals and number of generations.

We run the algorithms on each test problems with dimensions of 10, 100 respectively 1000. Both methods performance is recorded within the bound of one million function evaluations for each test suite and the results are averaged over 25 independent runs.

9.4 Experimental Results

The results are presented in Fig. 9.2. For functions F_1, F_5 and F_6 where the exact x^* global optima is known, we have semi-logarithmic plots with $\log(F(x_{best}) - F(x^*))$ vs. FES, showing the evolution in time of the so far best solution x_{best} on a logarithmic scale. For F_7 the simple evolution of the best solution is showed.

For the small dimensionality of 10 the final results of the two methods are very close for functions F_1, F_6 and F_7. Qualitatively the results do not differ, their absolute difference being less than 0.01, nevertheless with a small advantage for the genetic algorithms which exploits the basin of attraction of the optima more thoughtfully.

For function F_5 the genetic algorithm is not able to locate the global optima, not even in the case of only 10 dimensions, probably due to the strong interdependence of variables. Crossover is not able to promote linked variables provided that their linkage is not tight.

As dimensionality increases, one can observe that the performance of the genetic algorithms worsens, not being able to make improvements for long times even in the case of the unimodal sphere function. The method, due to the high selection pressure is unable to move to another region of the space, which may contain better (local) optima. Due to the dimensionality explosion, useful variations are hardly discovered. This worsening may imply a change of several orders of magnitude in the quality of the solutions as in the case of F_1, F_5.

On the F_7 "FastFractal 'Double Dip' " function we can observe a longer improvement phase of the genetic algorithm. Here as new levels of detail emerge at every resolution, the method does not need to move to new regions of the search space to be able to make improvements. Furthermore, as large part if not any member of the population is concentrated in the basin of attraction of the best found local-optima, the exploitation is quite effective. Nevertheless, the resolution of the function is still finite and after a high enough number of function evaluations the same behavior emerges: as the local-optima is "depleted", the algorithm can hardly make any additional improvements as this would require to move in into a new region of the search space. Selection pressure allows this only if the newly generated points (which much lay in the basin of attraction of another optima) already have competitive fitness value. This beneficial random variation is very unlikely for hard search spaces.

On the other hand, the cooperative search shows a more robust behavior on the whole spectrum of problems. Even as dimensionality increases it is able to quickly find very good solutions and can make sustained progress toward the global optima most of the time.

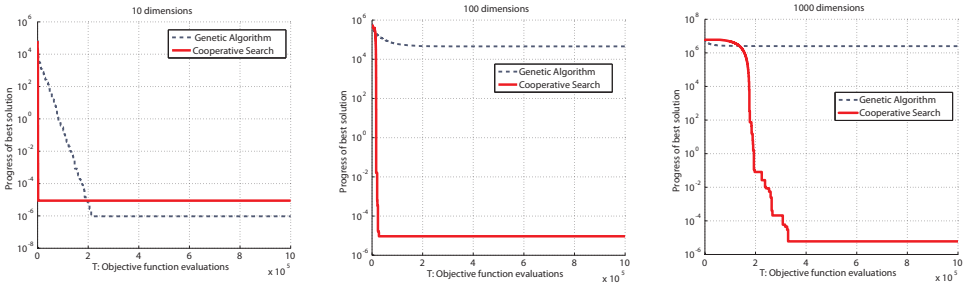
For the functions F_1, F_5, F_6 of dimensions 10 and 100 the solutions are always global optima or are located very close to them, satisfying the $F(x_{best}) - F(x^*) \leq 0.1$ relation.

The only certain case where the cooperative search does not find the global optima and also it does not exhibit a perpetual improvement is the F_6 functions for 1000 dimensions. Here, the bound of one million function evaluations is simply not enough for the method to overcame the massive multimodality of the function even if new regions are continuously explored. We can not state for certain what happens in the case of the F_7 function of dimension 10 where no improvements are made, as the value of the global optima is not known for this function. Nevertheless, as for the harder cases of the same function, our method is capable of sustained improvement, it is very likely that the found value represents the global or a very strong local optima.

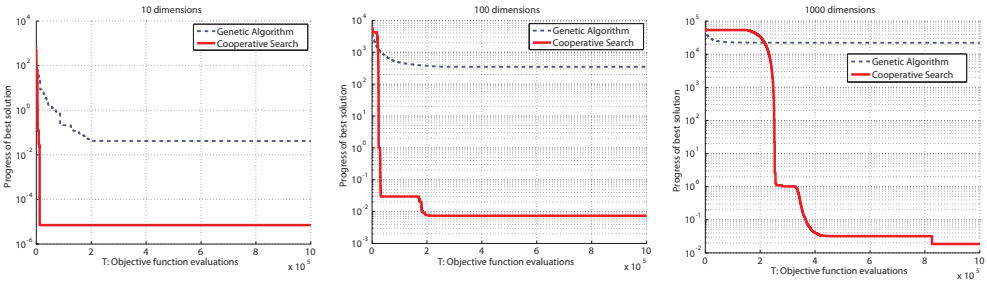
The F_7 function of dimensionality 100 and 1000 characterizes the cooperative search method in the best way as in this cases it is known that the global optima is not found within the bound of one million objective function evaluations (longer experiments have revealed better solutions). On these functions one can observe that the quickly identified low energy states are progressively improved during the search process with an ever descending slope. This contrasts the prematurely converged, constant line, which characterizes the genetic algorithm performance on the test functions of higher dimensions.

While genetic algorithm may get easily stuck in local optima due to the selection pressure the cooperative, specialized search offers a competent alternative as confirmed by the experiments.

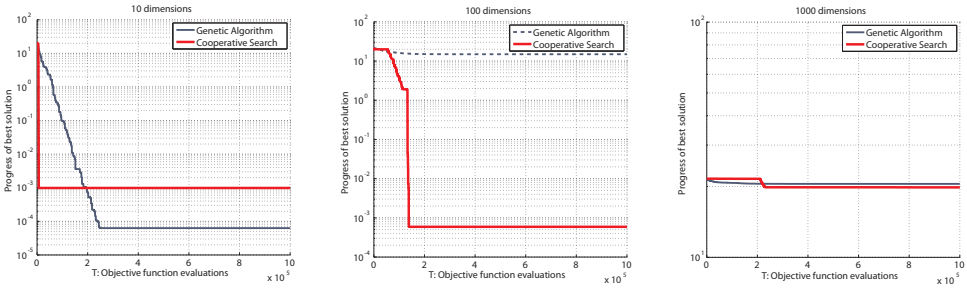
A. F_1 Shifted Sphere Function:



B. F_5 Shifted Griewanks Function:



C. F_6 Shifted Ackleys function:



D. F_7 FastFractal "DoubleDip" Function:

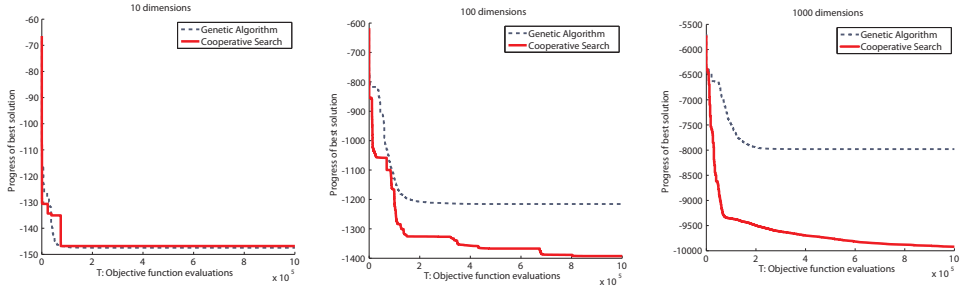


Figure 9.2: Results on the test suite for dimensions 10, 100 and 1000.

Chapter 10

Conclusions and Future Work

This chapter presents a summary of the investigations of this dissertation toward improving sustainability and scalability of evolutionary algorithms as well as efficient building block identification and mixing. Some future research directions are also presented.

10.1 Summary of Results

This dissertation is composed of two parts within the same context of improving scalability of competent search and optimization methods.

The first part introduces a machine learning and data analysis assisted local-search framework. Assisted by some novel, neural-network based, *automatic*, (online) model building techniques. Proposed framework can efficiently scale up to large problem and building block sizes as:

- There is no model search.
- Building blocks identification are a result of systematic search, not only initial random sampling.

The second part addresses one of the fundamental problems in EA, namely the convergent nature of the traditional evolutionary computation framework, which is the root of the premature convergence phenomena. To eliminate this issue and to be able to provide a sustainable, continuous development, an unconventional competent search model is proposed, based on cooperation, specialization and exploitation of search experience.

In Chapter 3 we suggest the usage of correlation analysis to assist model building. When branching from a current model in search for a better one, correlation analysis is employed to quickly identify the most promising extension,

while all other extension options are pruned. This can alleviate the model search cost by orders of magnitudes without affecting the quality of the delivered model. In a case study the model building complexity of the eCGA is reduced form $O(n^3)$ to linear, while still inferring the perfect problem structures on hard, hierarchical problems.

Chapter 4 advances the concept of model based local search, namely a building block hill-climber. This is a generic method for solving problems via (hierarchical - recursive) decomposition. Here resources are invested in finding good solutions, which are later analyzed by machine learning techniques for building block identification. The proper mixing of building blocks is assured by the local search strategy, which operates in the building block space, where it combines building blocks in a systematic and exhaustive manner. The continuous update of the building block representation of the individual results implicitly in the adaptation of the neighborhood structure to the combinative neighborhood of the current building blocks. In hierarchical problems – where building block hypothesis holds – moving the search to the combinative vicinity of the current building block representation facilitates the discovery of new building block, as low-order building blocks can be combined to form higher-order ones.

As they use only *random sampling* to supply initial building-blocks, classical probabilistic model building GA are lower bounded in the population size by the exponential of the order of dependencies covered by the probabilistic model. For larger order of dependencies, the cost of model building from the exponentially growing populations, quickly exceeds feasible limits and uprise beyond economical practicality.

In consequence, Chapter 5 introduces a new paradigm and method)Model Based Macro-Mutation Hill-Climber), where initial building-blocks supply is achieved by employing strong exploration techniques, which nevertheless follow *objective function guidance* whenever available. The macro-mutation based search used, has the capability of sampling a wast portion of the search space, howbeit greatly differs from random sampling by being a hill-climber. It does a biased search by never accepting states with lower fitness. Furthermore, by analyzing objective function variance it can filter out detrimental mutations which destroy already formed blocks. Investing the function evaluations into an exploratory local-search can facilitate the discovery of large modules. The method can also decide between the most fit module settings and their most competing schemata. Thus, the total number of samples needs to be just large enough, to make the delimitation of different modules possible with suitable machine learning techniques.

Chapter 6 analyzes scale up on very large problems of the model based local search framework. A variant of the model based trajectory framework is described, namely the Online Model Based Local-Search (OMBLS) that learns the problem structure online by means of topology preserving SOMs. OMBLS

operates via hierarchical decomposition, detected modules are used to collapse the search space and reformulate the optimization problem with discovered modules and their context-optimal settings as new search variables. For boundedly-difficult, non-overlapping, non-separable building-block problems, the cost of the OMBLS is upper bounded by the number of hierarchical levels, multiplied with the running complexity required by the module-wise local-search to converge to context-optimal settings at one hierarchical level. If context-optimal settings can be discovered by a strategy in linear time and the order of dependencies is limited to small k, the proposed framework holds a qualitative advantage over other methods as it scales at most linearithmicaly.

An unsupervised model learning, based on a graph clustering algorithm is described in Chapter 7. Beside its speed, the great advantage of the technique is that it allows the induction of various probabilistic model classes.

Chapter 8 re-examines some questions concerning the fundamentals of EA. The shortcomings of existing building block style test functions are surveyed and a view which promotes and emphasizes the generative potential of EAs is discussed. According to this conception, the great strength of the crossover operator lays in its capability to hybridize different designs, rather than in that of promoting similarity among the population.

Consequently, a new class of test problems, called Trident functions (TF) is introduced. The TF is dominated by a fully deceptive base function as global optima coincide with the minima of this function. The discrete optimal solutions are defined by a contribution function which rewards points from the search space where certain different genotypical features appear concomitantly. As the contribution function does not have an attractor basin, the deceptiveness of the base function can not be overcome using only simple neighborhood structures.

To alleviate the problem of premature convergence, Chapter 9 proposes a non-convergent cooperative search framework. For a balanced search, the exploration and exploitation are regarded as different subtask of the search process and their proper solving is targeted by different techniques and specialized individuals. Test results confirm the ability to identify high quality solutions and to generate and exploit useful variations in a continuous manner, that eventually lead to the global optima in most of the cases.

Methods built on the competent optimization techniques presented in this thesis, have been applied to real-world problems including ECG signal approximation (Iclanzan & Dumitrescu, 2006b; Iclanzan et al., 2006), learning clustering parameters (Szilágyi et al., 2008, 2009) and heart model optimization (Szilagyi et al., 2009).

10.2 Future Research

Future work will consider applying correlation guidance to enhance and qualitatively improve model-building in other competent EDA. Also the devising of new techniques that can efficiently use the information obtained by analyzing restricted groups of binary variables in feature space is of great interest.

Upcoming work will also advance the study and exploration of the GCEDA framework by interpreting clusterings as directed acyclic graphs, in order to build Bayesian networks. Other research will focus in the development of highly parallel model building and large scale optimization.

We look to exploit the fast automatic model building and low memory requirement of the proposed methods in Chapter 6 on problems with million(s) of variables, where the classical solving with population based methods would imply very large memory requirements and huge computational costs for model building.

Also instantiation of the proposed model based local search framework with probabilistic models will be investigated.

Future work will also focus on exploiting the advantages and possibilities offered by the cooperative paradigm described in Chapter 9. Lines of research include:

- Favoring the exploration of the whole search space by dividing it in subregions or by learning its topology. An intelligent exploration can prove effective on highly deceptive functions.
- Incorporating analysis and machine learning techniques that can reveal and exploit search space characteristics. Linkage learning and model building may be employed or the search strategies used may be adaptively altered.
- Take advantage of the naturally parallel nature of the framework.
- Use proxy function evaluation in the exploring subtask of the search.

Bibliography

Blum, C. & Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3), 268–308. [cited at p. 52]

Brohée, S. & van Helden, J. (2006). Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7, 488. [cited at p. 38]

Correa, E. & Shapiro, J. (2006). Model complexity vs. performance in the bayesian optimization algorithm. *Parallel Problem Solving from Nature-PPSN IX*, (pp. 998–1007). [cited at p. 41]

Duque, T. S., Goldberg, D. E., & Sastry, K. (2008). Enhancing the efficiency of the ECGA. In *Proceedings of the 10th international conference on Parallel Problem Solving from Nature* (pp. 165–174). Berlin, Heidelberg: Springer-Verlag. [cited at p. 14]

Etxeberria, R. & Larranaga, P. (1999). Global optimization using Bayesian networks. In *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)* (pp. 151–173). [cited at p. 38]

Harik, G. (1999). *Linkage Learning via Probabilistic Modeling in the ECGA*. Technical Report IlliGAL Report no. 99010, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign. [cited at p. 12, 39, 40]

Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1999). The compact genetic algorithm. *IEEE-EC*, 3(4), 287. [cited at p. 12]

Iclanzan, D. (2005). Genetic engineering algorithm. In *Proceedings of the 7th International Scientific Student Conference on Technical Sciences* Timisoara, Romania: Universitatea de Vest. Distributed on CD. [cited at p. 6]

Iclanzan, D. (2006). Genetic engineering as an optimization paradigm. In *Proceedings of FMTU 2006* (pp. 115–121). Cluj-Napoca, Romania: Transylvanian Musem Society. [cited at p. 6]

Iclanzan, D. (2007a). The creativity potential within Evolutionary Algorithms. In F. A. e Costa et al. (Ed.), *Advances in Artificial Life, 9th European Conference, ECAL 2007, Lisbon, Portugal, September 10-14, 2007, Proceedings*, volume 4648 of *Lecture Notes in Computer Science* (pp. 845–854).: Springer. [cited at p. 5]

Iclanzan, D. (2007b). Crossover: the divine afflatus in search. In P. A. N. Bosman (Ed.), *Late breaking paper at Genetic and Evolutionary Computation Conference (GECCO'2007)* (pp. 2497–2502). London, United Kingdom: ACM Press. [cited at p. 5]

Iclanzan, D. & Dumitrescu, D. (2006a). Competitive vs. cooperative optimization. In *Proceedings of the 8th International Scientific Student Conference on Technical Sciences* (pp. 115–121). Timisoara, Romania: Universitatea de Vest. Distributed on CD. [cited at p. 6]

Iclanzan, D. & Dumitrescu, D. (2006b). ECG parameter estimation using advanced stochastic search methods. In D. Isoc & E. Stancel (Eds.), *2006 IEEE-TTTC International Conference on Automation, Quality and tesing, Robotics. Digest of Junior Section* (pp. 17–22). Cluj-Napoca, Romania: Universitatea Technica Cluj. [cited at p. 6, 63]

Iclanzan, D. & Dumitrescu, D. (2007a). Exact model building in Hierarchical Complex Systems. In *Studia Universitatis Babes-Bolyai, Informatica Series*, volume Special Issue: KEPT 2007 - Knowledge Engineering: Principles and Techniques, Proceedings (pp. 161–168). Cluj-Napoca, Romania: Universitas Napocensis, Presa Universitara. [cited at p. 5]

Iclanzan, D. & Dumitrescu, D. (2007b). Overcoming hierarchical difficulty by hill-climbing the building block structure. In D. T. et al. (Ed.), *GECCO '07: Proceedings of the 9th annual conference on Genetic and Evolutionary Computation*, volume 2 (pp. 1256–1263). London: ACM Press. [cited at p. 5, 31]

Iclanzan, D. & Dumitrescu, D. (2007c). Overrepresentation in neutral genotype-phenotype mappings and their applications. In *Symbolic and Numeric Algorithms for Scientific Computing, 2007. SYNASC. International Symposium on* (pp. 427–432). Timisoara, Romania: IEEE Computer Society. [cited at p. 5]

Iclanzan, D. & Dumitrescu, D. (2008a). Going for the big fishes: Discovering and combining large neutral and massively multimodal building-blocks with model based macro-mutation. In *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation* (pp. 423–430). Atlanta, GA, USA: ACM. [cited at p. 4]

Iclanzan, D. & Dumitrescu, D. (2008b). How can artificial neural networks help making the intractable search spaces tractable. In *2008 IEEE World Congress on Computational Intelligence (WCCI 2008)* (pp. 4016–4023). Hong-Kong. [cited at p. 4]

Iclanzan, D. & Dumitrescu, D. (2008c). Large-scale optimization of non-separable building-block problems. In G. Rudolph, T. Jansen, S. M. Lucas, C. Poloni, & N. Beume (Eds.), *PPSN*, volume 5199 of *Lecture Notes in Computer Science* (pp. 899–908).: Springer. [cited at p. 4]

Iclanzan, D. & Dumitrescu, D. (2008d). Towards memoryless model building. In *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation* (pp. 2147–2152). Atlanta, GA, USA: ACM. [cited at p. 4]

Iclanzan, D. & Dumitrescu, D. (2010). Graph clustering based model building. In *PPSN XI - to appear in Lecture Notes in Computer Science* Krakow, Poland: Springer. (accepted). [cited at p. 3]

Iclanzan, D., Dumitrescu, D., & Hirsbrunner, B. (2009a). Correlation guided model building. In *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation* (pp. 421–428). New York, NY, USA: ACM. [cited at p. 3]

Iclănzan, D., Dumitrescu, D., & Hirsbrunner, B. (2010). Pairwise Interactions Induced Probabilistic Model Building. *Exploitation of Linkage Learning in Evolutionary Algorithms*, (pp. 97–122). [cited at p. 3]

Iclanzan, D., Fulop, P., & Dumitrescu, D. (2007). Neuro-Hill-Climber: A new approach towards more intelligent search and optimization. In *Symbolic and Numeric Algorithms for Scientific Computing, 2007. SYNASC. International Symposium on* (pp. 441–448). Timisoara, Romania: IEEE Computer Society. [cited at p. 5]

Iclanzan, D., Hirsbrunner, B., Courant, M., & Dumitrescu, D. (2009b). Cooperation in the context of sustainable search. In *IEEE Congress on Evolutionary Computation (IEEE CEC 2009)* (pp. 1904 – 1911). Trondheim, Norway. [cited at p. 4]

Iclanzan, D., Szilagyi, S. M., Szilagyi, L., & Benyo, Z. (2006). Advanced heuristic methods for ECG parameter estimation. In *CONTI 2006: Proceedings of the 7th International Conference on Technical Informatics* (pp. 215–220). Timisoara, Romania: Universitatea Politechica. [cited at p. 6, 63]

Jones, T. (1995). *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, Albuquerque, NM. [cited at p. 26]

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220, 671–680. [cited at p. 55]

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43, 59–69. [cited at p. 26, 32]

Lima, C. F., Sastry, K., Goldberg, D. E., & Lobo, F. G. (2005). Combining competent crossover and mutation operators: a probabilistic model building approach. In *GECCO '05* (pp. 735–742). NY, USA: ACM. [cited at p. 12]

Mitchell, M. & Holland, J. H. (1993). When will a genetic algorithm outperform hill climbing? In S. Forrest (Ed.), *Proceedings of the 5th International Conference on Genetic Algorithms* (pp. 647–647). San Mateo, CA, USA: Morgan Kaufmann. [cited at p. 42]

Pelikan, M. (2005). *Hierarchical Bayesian optimization algorithm: Toward a new generation of evolutionary algorithms*. Springer Verlag. [cited at p. 19]

Pelikan, M. & Goldberg, D. E. (2001). Escaping hierarchical traps with competent genetic algorithms. In L. S. et al. (Ed.), *GECCO '01:* (pp. 511–518). San Francisco, California, USA: Morgan Kaufmann. [cited at p. 32]

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. In W. B. et al. (Ed.), *GECCO '99*, volume I (pp. 525–532). Orlando, FL: Morgan Kaufmann Publishers, San Fransisco, CA. [cited at p. 38, 41]

Rind, D. (1999). Complexity and climate. *Science*, 284(5411), 105–107. [cited at p. 2]

Russel, S. & Norvig, P. (1995). *Artificial Intelligence: a Modern Approach*. Prentice-Hall. [cited at p. 1]

Shor, N. Z., Kiwiel, K. C., & Ruszcayǹski, A. (1985). *Minimization methods for non-differentiable functions*. New York, NY, USA: Springer-Verlag New York, Inc. [cited at p. 55]

Simon, H. A. (1969). *The Sciences of the Artificial*. Cambridge, Massachusetts: MIT Press, first edition. [cited at p. 2]

Szilágyi, L., Iclanzan, D., Szilágyi, S. M., & Dumitrescu, D. (2008). Gecim: A novel generalized approach to c-means clustering. In J. Ruiz-Shulcloper & W. G. Kropatsch (Eds.), *CIARP*, volume 5197 of *Lecture Notes in Computer Science* (pp. 235–242).: Springer. [cited at p. 4, 63]

Szilágyi, L., Iclanzan, D., Szilágyi, S. M., Dumitrescu, D., & Hirsbrunner, B. (2009). A generalized c-means clustering model optimized via evolutionary computation. In *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'09, Jeju Island, Korea)* (pp. 451 – 455). [cited at p. 3, 63]

Szilagyi, L., Szilagyi, S. M., Iclanzan, D., & Benyo, Z. (2006a). Quick ECG signal processing methods for on-line holter monitoring systems. In *CONTI 2006: Proceedings of the 7th International Conference on Technical Informatics* (pp. 221–226). Timisoara, Romania: Universitatea Politechica. [cited at p. 6]

Szilagyi, S. M., Szilagyi, L., Iclanzan, D., & Benyo, Z. (2006b). Adaptive ECG signal analysis for enhanced state recognition and diagnosis. In *CONTI 2006: Proceedings of the 7th International Conference on Technical Informatics* (pp. 209–214). Timisoara, Romania: Universitatea Politechica. [cited at p. 6]

Szilagyi, S. M., Szilagyi, L., Iclanzan, D., & Benyo, Z. (2006c). Unified neural network-based adaptive ECG signal analysis and compression. *SB-UPT TACCS*, 56(65)(4), 27–36. [cited at p. 5]

Szilagyi, S. M., Szilagyi, L., Iclanzan, D., & Benyo, Z. (2009). A weighted patient specific electromechanical model of the heart. In *Proc. 5th International Symposium on Applied Computational Intelligence and Informatics (SACI 2009)* (pp. 105–110). Timisoara, Romania. [cited at p. 4, 63]

Tang, K., Yao, X., Suganthan, P. N., MacNish, C., Chen, Y. P., Chen, C. M., & Yang, Z. (2007). *Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization*. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, http://nical.ustc.edu.cn/cec08ss.php. [cited at p. 55]

Thierens, D. & Goldberg, D. E. (1993). Mixing in genetic algorithms. In S. Forrest (Ed.), *Proc. of the 5th International Conference on Genetic Algorithms* (pp. 38–47). San Francisco, CA, USA: Morgan Kaufmann. [cited at p. 16]

Torczon, V. J. (1997). On the convergence of pattern search algorithms. *SIAM J. Optim.*, 7(1), 1–25. [cited at p. 55]

Turing, A. M. (1969). Intelligent machinery. In B. Meltzer & D. Michie (Eds.), *Machine Intelligence*, volume 5 chapter 1, (pp. 3–23). Edinburgh, UK: Edinburgh University Press. [cited at p. 1]

van Dongen, S. (2000a). *Graph Clustering by Flow Simulation*. PhD thesis, U. of Utrecht. [cited at p. 38, 42]

van Dongen, S. S. (2000b). *A stochastic uncoupling process for graphs*. Technical Report INS-R0010, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam. [cited at p. 38]

Watson, Beck, Howe, & Whitley (2003). Problem difficulty for tabu search in job-shop scheduling. *AIJ: Artificial Intelligence*, 143. [cited at p. 16]

Watson, R. A., Hornby, G., & Pollack, J. B. (1998). Modeling building-block interdependency. In *PPSN V: Proc. of the 5th International Conference on Parallel Problem Solving from Nature* (pp. 97–108). London, UK: Springer, LNCS. [cited at p. 31]

Watson, R. A. & Pollack, J. B. (1999). Hierarchically consistent test problems for genetic algorithms: Summary and additional results. In S. Brave (Ed.), *GECCO '99: Late Breaking Papers* (pp. 292–297). Orlando, Florida, USA. [cited at p. 32]

Yu, T.-L. & Goldberg, D. E. (2006). Conquering hierarchical difficulty by explicit chunking: substructural chromosome compression. In *GECCO '06:* (pp. 1385–1392). NY, USA: ACM Press. [cited at p. 39]

Yu, T.-L., Goldberg, D. E., Yassine, A., & Chen, Y.-P. (2003). Genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm. In *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 of *LNCS* (pp. 1620–1621). Chicago: Springer-Verlag. [cited at p. 39]

Yu, T.-L., Sastry, K., & Goldberg, D. E. (2005). Linkage learning, overlapping building blocks, and systematic strategy for scalable recombination. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation* (pp. 1217–1224). New York, NY, USA: ACM. [cited at p. 41]