PARALLEL SPIN GLASS SOLVING IN HIERARCHICAL BAYESIAN OPTIMIZATION ALGORITHM

Jiri Ocenasek
Martin Pelikan

Computational Laboratory (CoLab)
Swiss Federal Institute of Technology (ETH)
Hirschengraben 84, 8092 Zürich, Switzerland

Tel.: +41 1 6326410, fax: +41 1 6321703, e-mail: jirio@inf.ethz.ch,
Tel.: +41 1 6326322, fax: +41 1 6321703, e-mail: pelikanm@inf.ethz.ch

Abstract: *The hierarchical Bayesian optimization algorithm (hBOA) was proven to solve many emerging real world problems - like finding of ground states of spin glasses - with the same or better efficiency than the domain-specific methods. This paper describes an implementation of parallel hBOA that is based on parallel construction of Bayesian networks with decision trees. It was designed for running on a Beowulf cluster composed of Intel Pentium III computing nodes. In the experimental part we measure the efficiency and speedup of parallel hBOA on several spin glass instances. The results indicate that parallel hBOA is able to effectively utilize large number of processors and thus is suitable to solve large spin glass instances that are out of scope of the traditional methods.*

Key Words:. *Estimation of Distribution Algorithm, EDA, Bayesian Optimization Algorithm, BOA, decision trees, Bayesian network, speedup, efficiency, scalability, spin glass ground state, Beowulf cluster.*

1. Introduction

Estimation of distribution algorithms (EDAs) [1], [2], also called probabilistic model-building algorithms (PMBGAs) [3] and iterated density estimation evolutionary algorithms (IDEAs) [4], are stochastic optimization techniques. With an adequate probabilistic model they are able to effectively discover and mix the building blocks or partial solutions. In contrast to genetic algorithms, their performance is not affected by the ordering of genes and they are able to overcome the epistasis of bounded order in the fitness landscape. The relations between parameters of solved problem are discovered by machine-learning methods and incorporated into constructed probabilistic model. The solutions obtained by sampling of this probabilistic model are more superior than those obtained by pair wise recombination operators in classical genetic algorithms.

The necessary condition of successful EDA is the generality of its probabilistic model. The well known Bayesian optimization algorithm (BOA) [5] uses a very general probabilistic model - Bayesian network - to encode the relations between discrete parameters of solved problem. The local structures of Bayesian network in original BOA were stored in the tabular form, however yet the more efficient and robust representation of local structures in Bayesian network was achieved in hierarchical Bayesian optimization algorithm (hBOA) [6] by using decision trees.

The implementation of variant of parallel BOA with tabular form of Bayesian network for fine-grained architecture was described in [7] (PBOA) and its implementation for coarse-grained architecture – cluster of workstations – was described in [8] (DBOA). The variant of parallel hBOA with decision trees was proposed and simulated in [9] (MMBOA), but no real implementation was reported yet. This paper describes the first real implementation of parallel hBOA that uses parallel construction of Bayesian networks with decision trees and at the same time describes its first real-world application to spin glass benchmarks.

The following chapter defines the spin glass benchmark and motivates for its parallel solving. Chapter 3 describes the principles of sequential hBOA and explains the usage of decision trees in Bayesian networks. Chapter 4 contains the main description of implementation of parallel hBOA. The idea of parallel construction of decision trees was adopted from our MMBOA paper [9], but the probabilistic model sampling stage was redesigned to achieve better performance in non-multithreaded distributed environment with large communication latencies. The experimental part and efficiency evaluation is contained in Chapter 5.

2. Finding ground states of spin glass system

2.1. Benchmark definition

Finding the lowest energy configuration of spin glass system is an important task in modern statistical physics. From the optimization point of view a very general class of constraint-satisfaction problems can be converted into the form of spin glasses. Each configuration of spin glass system is defined by the set of spins

$$S = \{ s_i \mid \forall i \in \{1,\ldots,s^d\} : s_i \in \{+1,-1\}\},$$

where d is the dimension of spin glass grid and s is the size of spin glass grid. For the optimization by BOA the size of spin glass problem s^d is equal to the length of chromosome n, thus

$$n = s^d.$$

Each spin glass benchmark instance is defined by the set of interactions $\{ J_{i,j} \}$ between neighboring spins s_i and s_j in the grid. The energy of given spin glass configuration S can be computed as

$$E(S) = \sum_{i,j \in \{1,\ldots,n\}} J_{i,j} s_i s_j ,$$

where $J_{i,j} \neq 0$ if only i and j are neighboring positions in the grid. For general spin glass systems the interaction between neighbors is a continuous value $J_{i,j} \in <-1,+1>$, but we focus only on Ising model with either ferromagnetic bond $J_{i,j}=-1$ or antiferromagnetic bond $J_{i,j}=+1$, thus $J_{i,j} \in \{-1,+1\}$. Obviously, in the case of ferromagnetic bond the lower (negative) contribution to the total energy is achieved if both spin are oriented in the same direction, whereas in the case of antiferromagnetic bond the lower (negative) contribution to the total energy is achieved if both spins are oriented in opposite directions.

The goal of our optimization algorithm is to find a configuration S^* with the minimal energy ("ground state"):

$$S^* = \arg \min_{\forall S \in \{+1,-1\}^n} E(S)$$

In general not all the local contributions are negative, because the local interactions might be contradictory and some frustration might be present. In this case the finding of ground state of Ising spin glass system is NP hard for $d > 2$.

We encode each candidate spin glass configuration by binary chromosome of length n, where the bit of i-th gene indicates the +1 or -1 orientation of s_i spin in the configuration. Since our algorithm performs maximization, we use the negative energy $-E(S)$ as the fitness.

2.2. Motivation for parallel approach

The statistical analysis of experimental results obtained in [10] confirms that hBOA with added local search capabilities is able to find the ground state of randomly generated 2D instances of Ising spin glasses in polynomial time, with average complexity better than $O(n^{4.73})$. Our newest results indicate even better performance, such as $O(^{3.51})$. Whereas in 2D case there exist exact analytical methods that can solve spin glasses in polynomial time too [11], in 3D case these analytical methods can not be applied, since the problem becomes NP hard. The 3D spin glass instances are usually being solved by methods that combine brute-force search with domain-specific heuristics. The applicability ot these methods is limited to 3D grids of approximately 14x14x14 spins. Our empirical results show that hBOA is at least as effective as these traditional methods and we believe that by using parallel hBOA, the size of 3D instances solvable in tractable time might be extended.

3. Hierarchical Bayesian optimization algorithm

3.1. Main principles

The general procedure of EDA algorithm is similar to that of GA, but the classical recombination operators are replaced by probability estimation followed by probabilistic sampling. First, the initial population of EDA is generated randomly. In each generation, promising solutions are selected from the current population of N candidate solutions and the true probability distribution of these selected solutions is estimated. New candidate solutions are generated by sampling the estimated probability distribution. The new solutions are then incorporated into the original population, replacing some of the old ones or all of them. The process is repeated until the termination criteria are met.

Various models can be used in EDAs to express the underlying probability distribution. One of the most general models is the Bayesian network. A Bayesian network consists of two components. The first is a directed acyclic graph (DAG) in which each vertex corresponds to different gene which is treated as random variable. This graph describes conditional independence properties of the represented distribution. It captures the structure of the optimized problem. The second component is a collection of conditional probability distributions (CPDs) that describe the conditional probability of each variable X_i given its parents Π_i (ancestors) in the graph.

Together, these two components represent an unique probability distribution

$$p(X_0,..., X_{n-1}) = \prod_{i=0}^{n-1} p(X_i \mid \Pi_i) \cdot$$

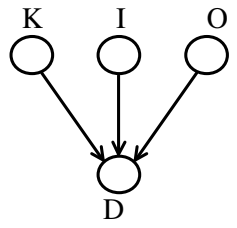
The well known EDA instances with Bayesian network are Bayesian optimization algorithm (BOA) [5], estimation of Bayesian network algorithm (EBNA) [12] and learning factorized distribution algorithm (LFDA) [13]. All these algorithms use metrics-driven greedy algorithm for construction of dependency graph. The implementation details of Bayesian network construction can be found in [14].

3.2. Advantages of decision trees

The hierarchical Bayesian optimization algorithm (hBOA) [6] is based on BOA with Bayesian networks, but it uses a more effective variant of conditional probability distributions in the form of decision trees. The second difference lies in using dvanced niching techniques (e.g. restricted tournament replacement) to avoid premature convergence. It would be unreasonable to parallelize hBOA if we had no evidence about its merits over BOA, because BOA was already parallelized in several variants like PBOA [7] and DBOA [8]. That is why the following paragraphs explain the advantages of using decision trees in hBOA instead of tabular representation of local probabilities in BOA.

In its most naive form, for each node X_i of Bayesian network the local conditional probability distributions $p(X_i|\Pi_i)$ are encoded using a tabular representation that is exponential in the number of parents of a variable X_i. For each possible assignment of values to the parents Π_i, we need to specify a distribution over the values X_i can take. A more effective representation can be obtained by encoding conditional probability distributions using decision trees. Since decision trees usually require fewer parameters, frequency estimation is more robust.

For example, consider the simple network in Fig. 1a, where the variables K, I, O and D correspond to the events "knocking noise", "immobilizer activated", "overheating sensor", and "engine defect" respectively. Assuming that all variables are binary, a tabular representation for D requires 8 parameters. One possible quantification is given in Fig. 1a. Note, however, that when the immobilizer is activated (i.e., when I = 1) the probability of engine defect D = 1 is 0, regardless of the values K and O. Thus, the interaction between D and its parents is simpler than the 8-way situation that is assumed in the tabular representation. In Fig. 1b an efficient representation based on decision trees is shown.



K I O	p(D|K,I,O)
1 1 1	0.00
1 1 0	0.00
1 0 1	0.95
1 0 0	0.95
0 1 1	0.00
0 1 0	0.00
0 0 1	0.40
0 0 0	0.05

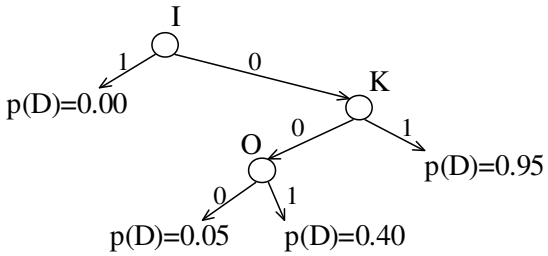


Fig. 1a) A simple Bayesian network structure and the associated conditional probability distribution for variable D (showing the probability values for D=1). 1b) Decision tree as an alternative representation of distribution.

This analogy explains the reason for utilizing decision trees in hBOA. Firstly, the estimation of probability model is more robust, estimating only the coefficients for significant combinations of Π_i. Secondly, the model building procedure allows for more precise model, since it explores networks that would have incurred an exponential penalty (in terms of the number of parameters required) and thus would have not been taken into consideration otherwise.

4. Parallel hBOA

The sequential hBOA builds all the decision trees at once. It starts with empty trees and it subsequently adds decision nodes. The quality of potential decision nodes is determined by Bayesian-Dirichlet metrics [14] which is able to determine the significance of statistical correlations between combinations of alleles in the population. A necessary condition for adding new split node is that the dependency graph remains acyclic–it must be guaranteed that no genes X_i, X_j exist such that X_i is used as a split in the tree for gene X_j and at the same time X_j is used as a split in the tree for gene X_i. In the parallel hBOA it is necessary to build each decision tree separately in different processor, so we need an additional mechanism to guarantee the acyclicity. In [9] we proposed the method that solves this problem. Each generation it uses random permutation $\boldsymbol{o}=(o_0,o_1,\ldots,o_{n-1})$ to predetermine topological ordering of genes in advance.

This means that only the genes $\{X_{o_0},...,X_{o_{i-1}}\}$ can serve as splits in the binary decision tree of target gene X_{o_i}. The model causality might be violated by this constraint, but from the empirical point of view the quality of generated models is the same as in the sequential case. The advantage is that each processor can create the whole decision tree asynchronously and independently of the other processors. Consequently, the linear speedup is achieved by removing this communication overhead.

We use a master-slave architecture for our parallel hBOA. CPU_0 acts as the master, it controls the distribution of jobs for slaves $CPU_1 \ldots CPU_m$. Due to dynamical workload assignment parallel hBOA does not require homogeneous cluster of workstations, it is adaptable to various computational resources. The way how master operates can be illustrated by this simplified pseudo-code:

```
Set t := 0;
Randomly generate Population(0) of size N;
Evaluate Population(0);
while termination criteria are not satisfied do
  Parents(t) := Tournament_selection(Population(t));
  Distribute Parents(t) to CPU1…CPUm using Broadcast();
  Generate random permutation o=(o0,…,on-1);
  Distribute the permutation o to CPU1…CPUm using Broadcast();
  for i:=0 to n-1 do
     Wait for any idle slave processor CPUj;
     Send job i to CPUj;
  end for
  Collect all decision trees T0,…,Tn-1 from CPU1…CPUm using Allgather();
  for all chromosomes in Offspring(t) do
    for i:=0 to n-1 do
      Traverse down the tree Ti conditionally on values of X_{o0} … X_{oi-1};
      Generate gene X_{oi} according to the reached leaf in Ti;
    end for
  end for
  Evaluate Offspring(t);
  Population(t+1) := Restricted_tourn._replacement(Population(t),Offspring(t));
  Set t := t+1;
end while
```

And the operation of slave processors can be illustrated as follows:

```
while termination criteria are not satisfied do
  Receive Parents(t) from CPU0;
  Receive permutation o=(o0,…,on-1) from CPU0;
  while job-queue in CPU0 is not empty do
    Receive next job number i from CPU0;
    Build the decision tree Ti for variable X_{oi} using potential splits X_{o0} … X_{oi-1};
  end while
  Send all constructed decision trees to CPU0;
end while
```

Note that not all m+1 processors are necessary, but only m processors are required, because both the processes CPU_0 and CPU_1 do not overlap too much and can be physically mapped to the same workstation.

From our analysis it follows that in each generation the complexity of model construction is $\mathbf{O}(n^2 N \log(N))$, which overcomes the complexity of parents selection $\mathbf{O}(nN)$, the complexity of fitness evaluation $\mathbf{O}(nN)$ and also the complexity of replacement $\mathbf{O}(nN^2)$. This is the reason why the construction of decision trees is the only parallelized part of our hBOA. The other parts – in the case of spin glass benchmark – do not represent performance bottleneck. Of course, for other benchmarks the fitness evaluation step might be expensive and needs to be parallelized too. But the parallelization of fitness evaluation is obvious and does not need special effort to be done effectively. For more details on parallel fitness evaluation see [15].

5. Experiments

5.1. Computational platform

Asgard is an Intel Pentium III Beowulf cluster located at the Swiss Federal Institute of Technology in Zurich. It combines the power of 502 CPUs on 251 dual-CPU nodes. Computational nodes are connected through a 100 Mb/s ethernet switches and communicate via MPI software interface.

5.2. Results

The results shown in Fig. 2 confirm high efficiency of parallel hBOA. First we measured separately the speedup of parallel model construction. The model construction appears to be successfully parallelized, for example for 50 processors and population size N=8000 we achieved speedup of 45. Moreover, the speedup grows when population size is increased, because the proportion between decision trees construction and decision trees communication increases. This is a good news for solving large spin glass instances where very large population size is usually needed.

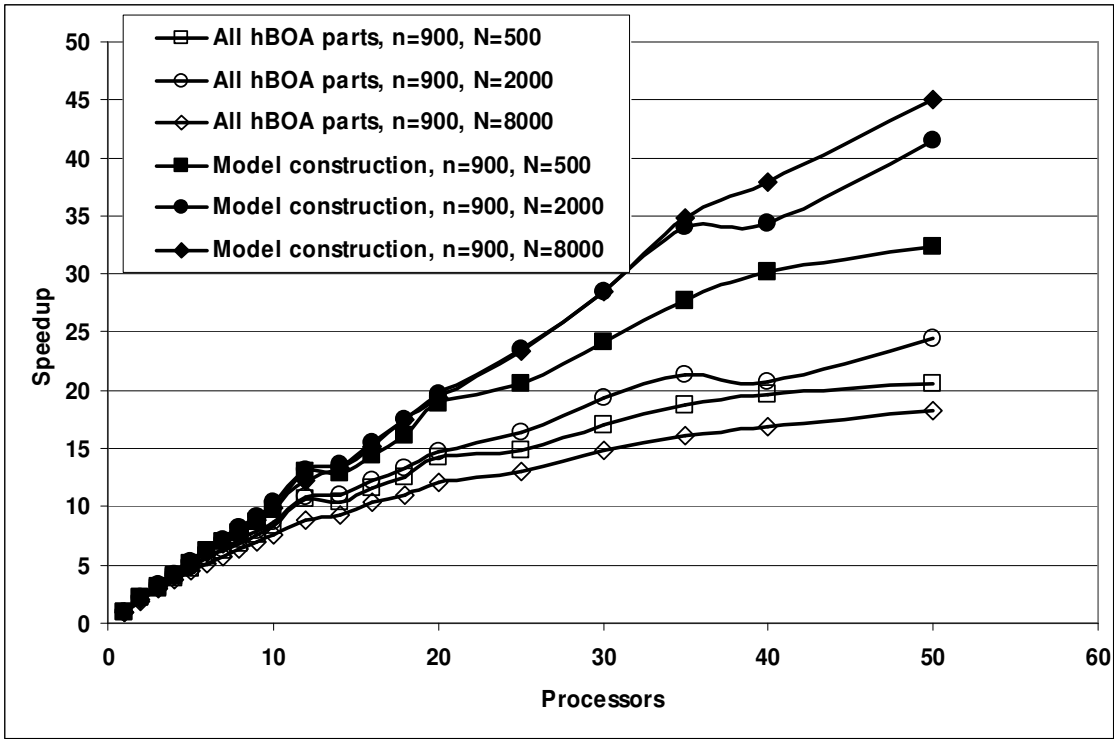


Fig. 2. Speedup of parallel hBOA on random spin glass instance with 30x30 spins.

Then we measured the speedup of the whole algorithm (including sequential selection, fitness evaluation, replacement, etc.). In Fig. 2 it seems that the speedup of the whole algorithm is also acceptable for small number of processors, but it easily gets saturated and for large number of processors it is hard to achieve speedup better than 20. This indicates that the additional parallelization of the remaining sequential parts might be needed to solve large spin glass instances using the whole Asgard capacity of 502 computational nodes.

6 Conclusions and future work

We designed and implemented a parallel hierarchical Bayesian optimization algorithm (hBOA) that works on Asgard Beowulf cluster of workstations. It uses our original method for parallel construction of probabilistic model composed of decision trees. Initially, this method was proposed and simulated for multithreaded architecture [9], but this is its first practical implementation and adaptation to non-multithreaded parallel architecture where message passing interface (MPI) is used to gather the parts of probabilistic model from parallel processes.

In the experimental part of this paper we investigate the efficiency and speedup of parallel hBOA on the real-world problem – finding of ground states of spin glass systems. This benchmark can substitute a large class of constraint-satisfaction problems that emerge in many engineering applications. Sequential hBOA can solve spin glasses very efficiently, but for large instances the absolute time for their solving is still inconvenient. Our empirical results indicate that parallel hBOA scales almost linearly with the number of processors used, so with higher number of processors it

can beat the limit of size of solvable instances and can serve as a more efficient alternative to present methods that solve spin glasses by employing brute force.

The achieved efficiency highly depends on the nature of the solved problem. In this paper we focused only on the parallel construction of probabilistic model, because the evaluation of spin glass energy is very fast and does not need to be parallelized. Of course, for more complex benchmarks a careful analysis is necessary to determine which parts of hBOA need to be performed in parallel by slave workstations and which parts are negligible and can be done sequentially by master workstation. Such detailed analysis for general benchmarks will be a subject of future work.

Acknowledgement

The authors would like to thank the members of Computational Laboratory ETH Zurich for valuable comments and discussions. We thank namely to Prof. Petros Komoutsakos, Prof. Matthias Troyer, Dr. Simon Trebst and Dr. Fabien Alet. Our experiments were run on Asgard Beowulf cluster.

References

[1] Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters, Parallel Problem Solving from Nature - PPSN IV, Springer Verlag, Berlin, pp. 178-187, 1996.

[2] Larrañaga, P.: A Review on Estimation of Distribution Algorithms. Estimation of Distribution Algorithms. A new Tool for Evolutionary Computation. P. Larrañaga, J. A. Lozano (eds.). Kluwer Academic Publishers, pp. 57-100, 2001.

[3] Pelikan, M., Goldberg, D. E., Lobo, F.: A survey of optimization by building and using probabilistic models, IlliGAL Report No. 99018, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, Illinois, 1999.

[4] Bosman, P. A. N., Thierens, D.: An algorithmic framework for density estimation based evolutionary algorithms. Utrecht University Technical Report UU-CS-1999-46, Utrecht, 1999.

[5] Pelikan, M., Goldberg, D. E., Cant\'u-Paz, E.: BOA: The Bayesian optimization algorithm. Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, vol. I, Orlando, FL, Morgan Kaufmann Publishers, pp. 525-532, 1999.

[6] Pelikan, M., Goldberg, E., Sastry, K.: Bayesian Optimization Algorithm, Decision Graphs, and Occams Razor. Illigal Report No. 2000020, , pp. 1-24, 2000.

[7] Ocenasek, J., Schwarz, J.: The Parallel Bayesian Optimization Algorithm, In: Proceedings of the European Symposium on Computational Inteligence, Physica-Verlag, Kosice, Slovak Republic, pp. 61-67, 2000.

[8] Ocenasek, J., Schwarz, J.: The Distributed Bayesian Optimization Algorithm for combinatorial optimization, EUROGEN 2001 - Evolutionary Methods for Design, Optimisation and Control, Athens, Greece, CIMNE, pp. 115-120, 2001.

[9] Ocenasek, J., Schwarz, J., Pelikan, M.: Design of Multithreaded Estimation of Distribution Algorithms, Genetic and Evolutionary Computation Conference GECCO-2003, accepted paper.

[10] Pelikan, M., Goldberg, D.E.: Hierarchical BOA Solves Ising Spin Glasses and MAXSAT, Genetic and Evolutionary Computation Conference GECCO-2003, accepted paper.

[11] Gallucio, A., Loebl, M.: A theory of Pfaffian orientations. I. Perfect matchings and permanents. Electronic Journal of Combinatorics 6, Research Paper 6, 1999.

[12] Etxeberria, R., Larrañaga, P.: Global optimization using Bayesian networks. Second Symposium on Artificial Intelligence (CIMAF-99), Habana, Cuba, pp 332-339, 1999.

[13] Mühlenbein, H., Mahnig, T.: FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. Evolutionary Computation, 7(4), pp. 353-376, 1999.

[14] Heckerman, D., Geiger, D., & Chickering, M. (1994). Learning Bayesian Networks: The combination of Knowledge and Statistical Data (Technical Report MSR-TR-94-09), Redmont, WA: Microsoft Research, 1995, pp. 1-53.

[15] Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Boston, MA: Kluwer Academic Publishers. 2000.