ACCELERATED BAYESIAN OPTIMIZATION ALGORITHMS FOR ADVANCED HYPERGRAPH PARTITIONING

Josef Schwarz[(1)]
Jiri Ocenasek[(2)]

[(1)]Brno University of Technology, Faculty of Information Technology,
Department of Computer Systems, Božetěchova 2, 612 66 Brno, CZ
Tel.: +420-5-41141210, fax: +420-5-41141270, e-mail: schwarz@fit.vutbr.cz,

[(2)]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

Abstract: *The paper summarizes our recent work on the design, analysis and applications of the Bayesian optimization algorithm (BOA) and its advanced accelerated variants for solving complex – sometimes NP-complete – combinatorial optimization problems from circuit design. We review the methods for accelerating BOA for hypergraph-partitioning problem. The first method accelerates the convergence of sequential BOA by utilizing specific knowledge about the optimized problem and the second method is based on the parallel construction of a probabilistic model. In the experimental part we analyze the advantages of acceleration techniques and prove that BOA is able to solve hypergraph partitioning problems reliably, effectively, and without the need for specifying control parameters and encoding schemes as in recombination-based genetic algorithms.*

Key words*: optimization problems, decomposition and allocation problems, graphical probabilistic model, Bayesian network, Bayesian-Dirichlet metric, Bayesian optimization algorithm, problem knowledge, parallelization, hypergraph partitioning.*

1 Introduction

Bayesian optimization algorithm (BOA) belongs to the advanced evolutionary algorithms based on the estimation and sampling of graphical probabilistic models. This algorithm does not suffer from the disruption of building blocks known from the standard genetic algorithms. It is able to automatically discover the linkage between parameters of the optimized problem and incorporate them into the constructed Bayesian network. Consequently, BOA does not require the specification of the control parameters and genetic operators. The original BOA – firstly published in [1] – was capable of solving binary single-criterion problems decomposable into subproblems of bounded order. Recently the hierarchical Bayesian optimization algorithm (hBOA) [2] and mixed Bayesian optimization algorithm (MBOA) [3] were proposed that are able to solve hierarchically decomposable problems and MBOA is even capable of employing the linkage between continuous parameters.

This paper reviews our recent work on acceleration of BOA and summarizes the ideas on how to utilize BOA for solving real-world optimization tasks that can be transformed into the form of decomposition or allocation of hypergraphs. There were two important goals of that work:

- Optimization of combinatorial problems
- Acceleration using the problem knowledge and parallelization techniques

The main attention was focused on testing of various types of decomposition of hypergraphs using the well known benchmarks and objective functions. These various decomposition algorithms can be used in many contexts, e.g. for decomposition of complex systems, telecommunications networks, complex circuit structures etc. In addition, the accelerated versions of BOA can be easily modified for solving other combinatorial or numerical optimization problems.

The following section defines a general combinatorial optimization problem and section 3 shows the basic principles of BOA. In our knowledge-based BOA algorithm (KBOA) [4] reviewed and tested in section 4 the partial knowledge (prior information) about the problem was used to modify the probabilistic model. In addition, an injection of building blocks was used to improve the quality of initial population.

Also, a very promising reduction of optimization time due to parallel construction of Bayesian network – known from our pipelined parallel Bayesian optimization algorithm (PBOA) [5] and distributed Bayesian optimization algorithm (DBOA) [6] – is described in section 5.

2 Combinatorial problems specification

It is possible to specify two basic combinatorial tasks [7]:

A. Permutation problems – search for the global optima of the function *f* is defined across symmetrical group G_n consisting of all permutations of *n* parameters

$$f: G_n \rightarrow R, \quad (1)$$

where the permutation is defined by an n-tuple of different integers

$$P=(p_1, p_2, ..., p_n) \quad (2)$$

The goal of optimization (minimization) can be defined as

$$P_{opt} = \arg\min_{P \in G_n} f(P) \quad (3)$$

One of the typical permutation problems is the TSP problem.

B. Optimization task with exponential cardinality of the search space. Let $R_m=\{1, 2,...,r\}$ be a set including first *r* natural numbers. The Cartesian product R_m^n includes n-tuples $\pi=(\pi_1, \pi_2,..., \pi_n) \in R_m^n = R_m \times R_m \times ... \times R_m$, with components π_i being integer values from closed interval $[1, r]$. The function $f: R_m^n \rightarrow R^+$ is a mapping from Cartesian product R_m^n onto real numbers. The optimization problem can be defined as

$$\pi_{opt} = \arg\min_{\pi \in R_m^n} f(\pi) \quad (4)$$

3 Bayesian optimization algorithm (BOA)

Generally, the solution of optimization problem can be represented by a vector of parameters. Many combinatorial optimization algorithms have no mechanism for capturing inter-parameter dependencies and they search the state space only in the neighborhood of previous best solution. However, the discovery of inter-parameter dependencies allows the optimization to effectively concentrate the sampling on more regions of the search space which have appeared to be promising in the past. The second disadvantage of classical genetic algorithms (GAs) lies in the necessity of setting many parameters for genetic operators, e. g. crossover, mutation and selection rate and the choice of proper encoding of solutions. That is why we have analyzed and used Bayesian optimization algorithm as a sophisticated representative of the estimation of distribution algorithms (EDAs), also called probabilistic model-building genetic algorithms (PMBGAs), see [8], [9]. The classical crossover and mutation operators used in standard GA are replaced in EDAs by probability estimation and sampling techniques.

In EDAs the statistics about the search space is explicitly maintained by creating models of the good solutions found so far. Obviously, the complexity of such techniques depends on the complexity of used probabilistic models which have to be adequate to the structure of optimized problems. The Bayesian network was proven to be one of the most general probabilistic models capable to capture the structure of most combinatorial problems.

Let us denote:

$D = (X^1, X^2,..,X^N)$ with $X^j \in D$, is the population of the solutions/strings/individuals
$X = (X_0, X_1,..,X_{n-1})$ is a string/individual of length *n* with X_i as a variable
$x = (x_0, x_1,..,x_{n-1})$ is a string/individual with x_i as a possible instantiation of variable X_i, $x_i \in \{0,1\}$
$P = (p(x_0), p(x_1),..,p(x_{n-1}))$, with $p(x_i) \in [0,1]$ is the vector of univariate marginal probabilities
$p(X_0, X_1,..., X_{n-1})$ denotes the *n* dimensional distribution
$p(x_0, x_1,..., x_{n-1}) = p(X_0=x_0, X_1=x_1,..,X_{n-1}=x_{n-1})$ denotes an instance of the *n* dimensional distribution

Without the loss of generality we will focus on binary problems. Let *f* be an objective/cost function defined across the set of binary vectors of length *n*

$$f: \{0,1\}^n \rightarrow R, \quad (5)$$

which evaluates each binary vector/string *x* by a real number. The aim is to find the global extreme of the function *f*.

In the case of minimization the goal is to find

$$x_{opt} = \arg\min_{x \in \{0,1\}^n} f(x) \tag{6}$$

The performance of EDAs that work on the basis of probabilistic model construction and sampling can be specified by the following code (see also Fig.1):

```
Generate initial population of individuals D(0) of size M (randomly);
While termination criteria is false do
begin
   Select the parent population D^s(t) of N individuals according to a
selection method;
   Estimate the probability distribution of the selected parents D^s(t);
   Generate new offspring O(t) according to the estimated prob. model;
   Replace part of D(t) by generated offspring O(t), yielding D(t+1);
end
```

In Bayesian optimization algorithm (BOA) the Bayesian network (BN) is used to encode the structure of an optimized problem. Each variable X_i in the string is treated as a random variable and represented by one node in the dependency graph. For each variable X_i there exists a set of variables Π_{X_i} called parents that conditionally determine the value of X_i. The joint distribution of individuals is encoded as

$$p(X) = \prod_{i=0}^{n-1} p\left(X_i \mid \Pi_{X_i}\right) \tag{7}$$

Generally, the existence of oriented edge from X_j to X_i in the dependency graph implies that the variable X_j belongs to the set Π_{X_i}. To reduce the space of possible networks, the number of incoming edges into each node is limited to *k*. The equation (7) can be interpreted as an encoding of the structure of optimized problem by representing conditional (in)dependency among variables.

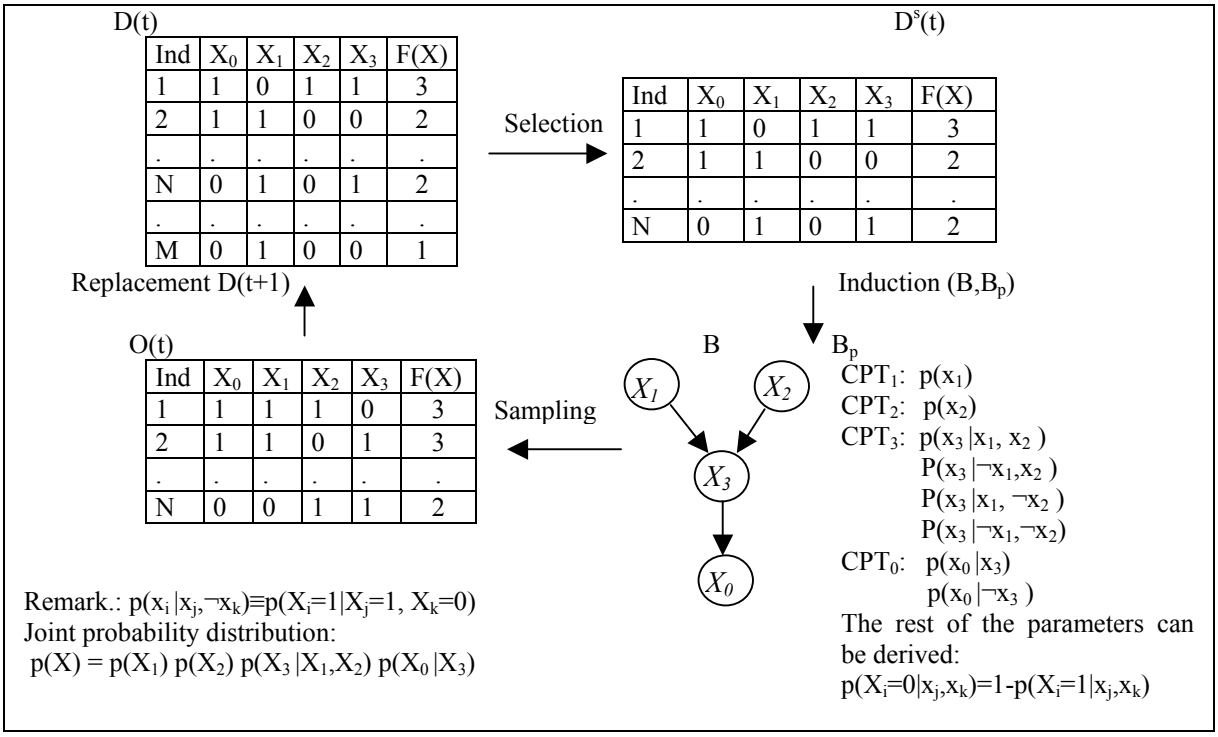


Fig. 1: Illustration of Bayesian optimization algorithm

The key step is the estimation of the probability *p(X)* via finding the Bayesian network with maximum score measure. The first step is the construction/induction of the dependency graph of the Bayesian network from the population of promising individuals/solutions $D^s(t)$. The second step of learning of Bayesian network includes the estimation of conditional probability tables (CPTs) given the dependency graph. The learning of new model is repeated in each generation of the evolution process.

The Bayesian Dirichlet (BD) metric [10] is used to measure the quality of Bayesian networks. To construct quickly a good network, the greedy algorithm is used in such a way that in each step the best edge is added according to BD metric. After the network construction new instances are generated using the univariate and conditional probabilities. An example of the Bayesian network is shown Fig.1 (right bottom).

4 Hypergraph decomposition and allocation

The decomposition of systems, e.g. telecommunication systems, database systems, and VLSI circuits, is a challenging task from engineering practice (see [11],[12]). These large systems are determined by a set of entities and multivariate relations among them. Such systems can be represented by hypergraphs and the problem of their decomposition can be transformed into the problem hypergraph partitioning. Let us assume a hypergraph H=(V,E), with n=|V| nodes and m=|E| edges. The goal is to find such partitions V1,V2 ($V=V1\cup V2$, $V_1\cap V_2=\emptyset$) of V that minimize the number of external hyperedges that have nodes in V1 as well as in V2. The set of such external hyperedges is denoted as $E_{cut}(V1,V2)$ and the number of hyperedges in this set is denoted as C_1 (cut value). The objective function to be minimized is then defined as:

$$C_1(V_1,V_2) = |E_{cut}(V_1,V_2)| \quad (8)$$

$$C_1(V_1,V_2) = |\{e\in E \mid e\cap V_1 \neq \emptyset,\ e\cap V_2 \neq \emptyset\}| \quad (9)$$

The balance between size of V1 and size of V2 is constrained by parameter α:

$$(1-\alpha)\, n/2 \le |V_1|, \quad |V_2| \le (1+\alpha)n/2, \qquad \alpha \in (0,1) \quad (10)$$

This parameter α specifies the type of decomposition. It is a bisection in case of its zero value – see fig. 2a else it is partitioning in general. Allowing α to be non-zero can provide smaller cut value C_1, see fig.2b. Another way how the different sizes of partitions can be simply expressed is their difference $C_2 = ||V_1| - |V_2||$.

The *k*-way decomposition can be realized either by recursive approach using iterative bisection or by nonrecursive approach. The disadvantage of nonrecursive approach is the necesity to use a bulk alphabet for encoding. Recursive algorithms are more frequently used. The objective function is defined as

$$C_1(V_1,V_2,\ldots,V_k) = |E_{cut}(V_1,V_2,\ldots,V_k)| = |\{e\in E \mid \exists\, i,j, i\neq j\ e\cap V_i \neq \emptyset,\ e\cap V_j \neq \emptyset\}| \quad (11)$$

under constraint

$$(1-\alpha)\, n/k \le |V_i| \le (1+\alpha)\, n/k, \qquad \alpha \in (0,1),\ i,j = 1,\ldots,k \quad (12)$$

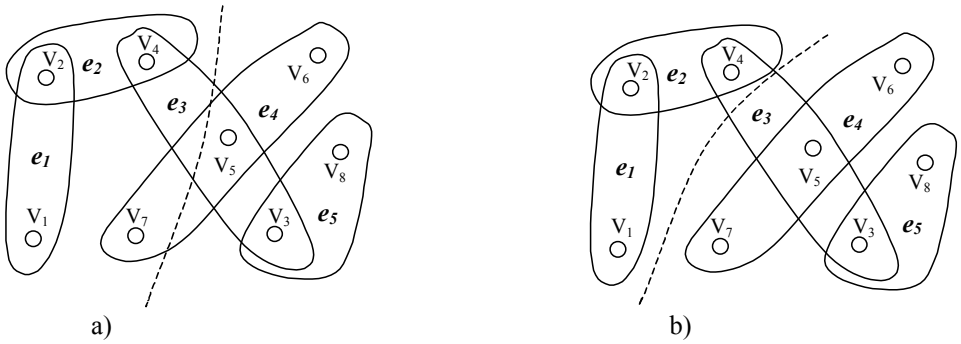


Fig. 2: Hypergraph decomposition a) bisection (with the zero balance α=0 and C_1=2), b) partitioning with α=1/4 and C_1=1.

The usage of a bisection yields lower time complexity, but this approach does not allow for detection of natural clusters in a hypergraph structure. One solution to this problem is the usage of objective function that combines both the balance minimization and the cut value minimization:

$$R_c = C_1/F_R(V_1, V_2, \ldots, V_k), \quad (13)$$

where C_1 is the cut value and F_R function includes the balance criteria. One possibility is to define function F_R as a product of partition sizes $F_R(V_1,V_2,\ldots,V_k) = |V_1|*|V_2|*\ldots*|V_k|$. Additional variants can be found in [13].

4.1 Bisection of hypergraphs using problem knowledge

For many combinatorial problems the information about the structure of the problem is available. In the case of hypergraph partitioning, the structure of the problem is given by the list of edges between nodes of hypergraph. But the decomposition of the problem is not simple because the variables of the cost functions are overlapping. We can only expect that adjacent nodes in the hypergraph are dependent but we have no exact knowledge about the degree of independence between non-adjacent nodes.

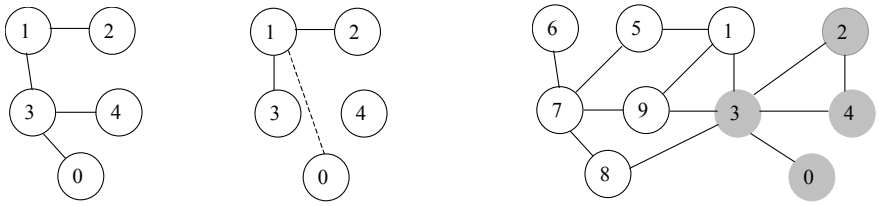
The purpose of our paper [4] was to propose a KBOA algorithm that learns the structure of the problem and use it together with the partial (local) information about the linkage to improve the performance of BOA [12]. Two possible approaches were tested:

1. The first approach was based on the prior information about the problem in the BD metric. It is out of the scope of this paper to define and explain the whole BD metric, but it is important to note that it has the form

$$p(D,B \mid \xi) = p(B \mid \xi).p(D \mid B,\xi) \tag{14}$$

The term $p(D \mid B,\xi)$ (unwinded because of its complexity) denotes the main part of BD metrics – the likelihood of population D given network B. The term $p(B \mid \xi)$ represents the prior probability of the network B and is usually unused in BOA. We use this term to penalize the networks that are not similar to the partitioned hypergraph. We use simple assignment $p(B \mid \xi) = c\kappa^{\delta}$, where c is a normalization constant, $\kappa \in (0,1]$ is a penalty constant factor (usually close to zero) and δ is the number of edges in the final Bayesian network having no match in the hypergraph to be partitioned. Consequently, the greedy algorithm that constructs the dependency graph uses local information about the problem by preferring the edges that have its counterpart in the hypergraph.

2. The second approach lies in the injection of building blocks of predefined size into the initial population. The standard BOA generates initial population randomly, but we have tested the usage of clusters based upon the knowledge of the hypergraph structure. The detection of such a cluster in a hypergraph is shown in the Fig. 3:



hypergraph dependency graph hypergraph with a cluster

a) *b)*

Fig. 3: The usage of prior information a) matching between hypergraph and dependency graph, b) cluster identification.

4.2 Experimental results

We used four types of graph structures for experiments:

- Regular graphs Gridn.c with grid structure [12], [14] where the notion n specifies the number of nodes, c specifies the minimal cut size. As an example a bisection of graph Grid100.2 is represented in Fig.4a having a 2-edge bottle-neck in the dashed cut line.
- Random geometric graphs Un.d [14], [15]. Random geometric graph on n vertices is placed in the unit square and coordinates of its nodes are chosen uniformly. There exists an edge between two vertices if their Euclidean distance is h or less, where the expected vertex degree is specified by $d = n\pi h^2$.
- Caterpillar graphs CATk_n [14], with k articulations, six legs for each articulation, and n nodes, see Fig. 4c with k=3 and n=21. This type of graph serves as a hard benchmark.
- Hypergraphs representing real circuits labeled by ICn [16]. The global optima is not known. The hypergraphs can be also specified by pair nodes/edges: IC67/138, IC151/419, IC116/329, Fract149/147. The structure of these circuits can be characterized as a random logic.

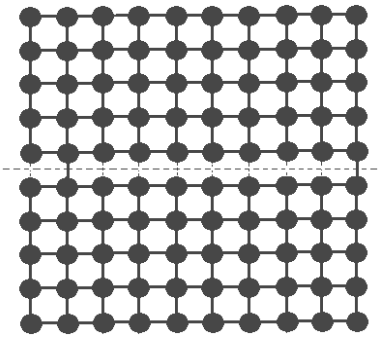


Fig.4a Grid graph

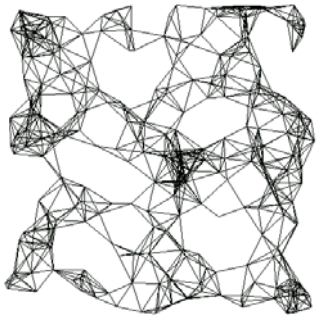


Fig.4b Geometric random graph

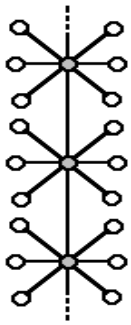


Fig.4c Caterpillar graph

We have performed various experiments to demonstrate the efficiency of the developed algorithm KBOA compared to the original BOA algorithm. The best known solutions are determined by KBOA itself or by the well known hMetis program [17]. We performed ten independent runs for KBOA and BOA on each of the 14 test graphs of various type and size. Our results indicate that for all types of graphs the minimum population size required by KBOA to reach the global optima is about five times lower than the minimum population size required by BOA. Due to using lower population size KBOA exhibits rapid decrease of computation time with respect to BOA.

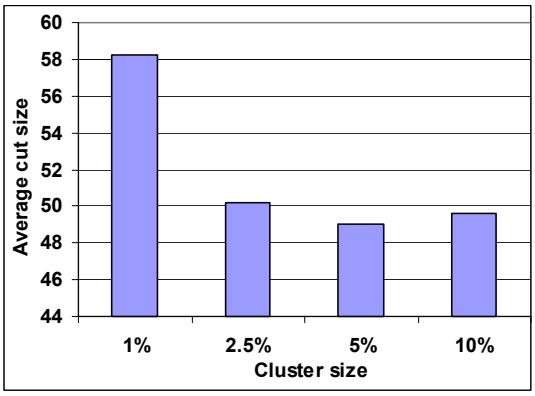


Fig. 5: Cluster size versus average cut size for IC

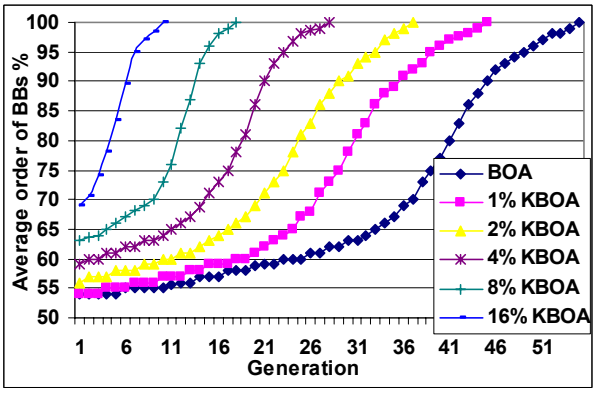


Fig. 6: Evolution of building blocks

In Fig. 5 the performance of KBOA algorithm on the hypergraph IC151 for different size of injected clusters is shown. The size of clusters is expressed as a percentage of the number of hypergraph nodes. The ideal cluster size found is about 5%. In the case of smaller clusters the growth of proper building blocks is weak and in the case of too big clusters the optimization leads into local optima.

In Fig.6 the average growth of the order of correct building blocks order in the population is depicted. This experiment is done for grid graph Grid100.2 with cluster size as a parameter. For the experiment the population size is set to N=300-2400 for KBOA and to N=2400 for BOA to get the global optimum for all sizes of cluster. Because two symmetrical optimal solutions exist, we separate all the strings into two groups according to their similarity to each of the possible solution and the calculation of order of building blocks based on the Hamming distance is summarized. It is clear that for small cluster size BOA the average order of BBs in the first generation is about 50% which is typical for initial random population. A strong influence of the cluster size on the speed up of the evolution is evident. Let us note that in the 1% KBOA the cluster injection is not used and it is affected only by the phenomenon of prior probability (penalization) of the Bayesian network (see chapter 5) and only this phenomenon differentiates 1% KBOA from the BOA.

5 Parallelization of BOA algorithm

The main problem of BOA algorithm that flows from the size of solved problems is the time complexity. This time can be reduced up to a factor of n by employing parallel computation. The parallelization of BOA means the parallelization of all its parts, but in this paper we focus on the parallelization of probabilistic model construction because it is not as obvious as the parallelization of fitness computation and it needs a special effort to be done effectively.

5.1 Parallel construction of BN

The goal is to utilize more processors when searching for a good network. Our consolation is that the BD metric is separable and can be written as a product of n factors, where i-th factor expresses the influence of edges ending in the variable X_i. Thus, for each variable X_i the set of parent variables Π_{X_i} can be determined independently. It is possible to utilize up to n processors, each processor corresponds to one variable X_i and it examines only edges leading to this variable. Note, that even for the estimation of local CPDs each processor needs the full copy of parent population.

Usually we can use only m processors. If $m<n$, then each processor evaluates only edges ending in its subset of nodes. For example see Fig. 7 where n=6 and m=2, such that each processor manages incoming edges of n/m=2 nodes.

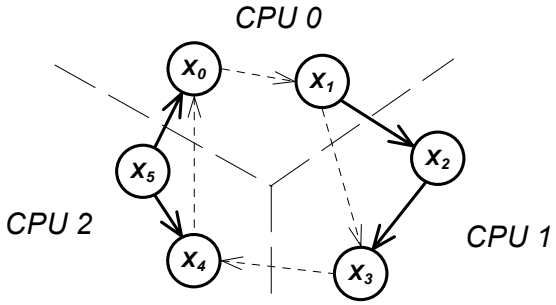


Fig. 7: CPU_0 is adding edges ending in nodes 0 and 1; CPU_1 is adding edges ending in nodes 2 and 3; CPU_2 is adding edges ending in nodes 3 and 4. Note that in the case of naïve parallel BN construction the acyclicity might be violated (dot line)!

In Fig. 7 you see that the naïve parallel BN construction might produce the unwanted cycles. The addition of edges is parallel, so we need an additional mechanism to keep the dependence graph acyclic. The most advantageous way how to handle this problem is to predetermine the topological ordering of nodes in advance. At the beginning of each generation, the random permutation of numbers $\{0,1,\ldots,n\text{-}1\}$ is created and stored in the $\boldsymbol{o}=(o_0,o_1,\ldots,o_{n-1})$ array. Each processor uses the same ordering. The direction of all edges in the network should be consistent with the ordering, so the addition of an edge from X_{o_j} to X_{o_i} is allowed if only $j<i$. Evidently, the variable X_{o_0} has no predecessor and is forced to be independent, thus the space of possible networks is reduced. To compensate this phenomenon the processors generate new permutation after each generation.

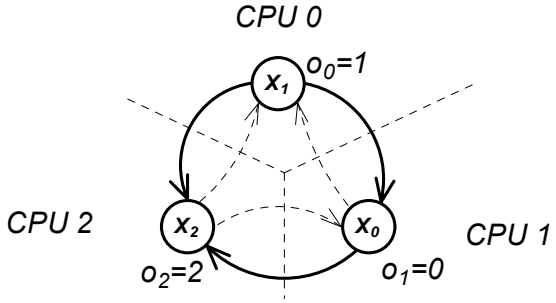


Fig.8: Example of BN construction with m=n=3. Permutation vector **o**=(1,0,2) determines the topological ordering of nodes such that the dashed edges are not allowed.

5.2 The advantages of parallel BN construction

The advantages of predetermined ordering of nodes are noticeable. We completely removed the communication overhead even if the quality of generated network is almost the same as in the sequential case. Each processor can create its part of probabilistic model independently of the other processors, because the acyclicity constrains are implicitly satisfied.

The parallel BN construction can be written as:

```
Start with an empty network B;
Generate random permutation o=(o0,…,on-1);
Assign ending nodes to each of m≤n parallel processes;
Distribute the permutation o among the parallel processes;
for each ending node oi do in parallel                          ...  O(n/m)
begin
   for j := 0 to i-1 do                                         ...  O(n)
   begin
     Compute the score of eventual appending (oj,oi) to B;      ...  O(N)
   end
   while incoming_node_degree d(oi) < min(k,i) do               ...  O(k)
   begin
      for each unapplied starting node oj having j<i do         ...  O(n)
      begin
        select the edge (oj,oi) giving the highest score;
      end
      Append the selected edge (oj,oi) to the network B;
      Remove the node oj from the set of unapplied parents of oi;
      for each unapplied starting node oj having j<i do         ...  O(n)
      begin
        Update the score of eventual appending (oj,oi) to B;... O(2^k.N)
      end
   end
end
```

The overall time complexity of parallel BN construction can be expressed by the term $\mathbf{O(k\ n^2\ 2^k\ N\ /\ m)}$. After simplification it gets $\mathbf{O(n^3/m)}$. Note that a linear speedup can be reached, when compared with time complexity $\mathbf{O(n^3)}$ of sequential BN construction. This is in agreement with the empirical results from experiments with our pipelined parallel Bayesian optimization algorithm (PBOA) [5] and distributed Bayesian optimization algorithm (DBOA) [6].

6 Conclusion and future work

In this paper we have reviewed and presented our two main approaches for the acceleration of the Bayesian optimization algorithm (BOA) [1]. The first approach was problem-dependent. The goal was to incorporate problem-specific knowledge into the construction of Bayesian networks for hypergraph decomposition. The resulting algorithm KBOA [4] modifies the construction of Bayesian networks so that the edges that have their counterpart in the decomposed hypergraph are preferred. This enhancement positively influences convergence speed. Higher efficiency was achieved by injecting high quality clusters into the initial population. This was experimentally shown (see Fig. 6) to be a promising tool for enhancement of the population genotype. Consequently, population size can be significantly reduced – in our experiments by a factor of five – what results in significantly shorter optimization time.

The second reviewed acceleration approach was problem independent. In PBOA [5] and DBOA algorithms [6] we achieved linear speedup by parallelization. In this paper we focused on the parallel construction of Bayesian network and recalled the common principle of PBOA and DBOA – a principle of the restricted ordering of genes that allows for asynchronous construction of dependency graph, without the need for synchronization due to acyclicity checks.

Future work should focus on the utilization of hierarchical Bayesian optimization algorithm [2] for hypergraph partitioning. Although the experiments with BOA do not indicate that the hypergraph partitioning benchmark exhibits strong hierarchical interactions, the first experiments with hierarchical BOA on hypergraph partitioning seem to be promising. Moreover, hBOA can use problem knowledge in the same way as BOA and also its parallelization was already solved and proposed in [18] and [19].

Acknowledgement

This research has been carried out under the financial support of the Research intention no. CEZ: J22/98: 262200012-"Research in information and control systems" (Ministry of Education, CZ) and the research grant GA 102/02/0503 "Parallel system performance prediction and tuning" (Grant Agency of Czech Republic). The authors would like to thank Prof. Petros Komoutsakos and Dr. Martin Pelikan from Computational Laboratory of the Federal Institute of Technology ETH Zürich for reviewing the final version of this paper and for valuable comments.

References

[1] Pelikan, M.: A Simple Implementation of Bayesian Optimization Algorithm in C++ (Version 1.0). Illigal Report 99011, February 1999, pp. 1-16.

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

[3] Ocenasek, J., Schwarz, J.: Estimation of Distribution Algorithm for mixed continuous-discrete optimization problems. In: 2nd Euro-International Symposium on Computational Intelligence, IOS Press, Kosice, Slovakia, 2002, pp. 227-232.

[4] Schwarz, J., Očenášek, J.: The knowledge-based evolutionary algorithm KBOA for hypergraph bisectioning. Proceedings of the Fourth Joint Conference on Knowledge-based Software Engineering, IOS Press, Brno, Czech Republic, 2000, pp.51-58.

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

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

[7] Kvasnička, V, Pospíchal, J., Tiňo, P. Evolučné algoritmy. Bratislava, STU, 2000.

[8] Pelikan, M., Goldberg, D. E., & Lobo, F.: A Survey of Optimization by Building and Using Probabilistic Model, Illigal Report 99018, September 1999, pp. 1-12.

[9] Ocenasek, J.: Parallel Estimation of Distribution Algorithms. PhD. Thesis, Faculty of Information Technology, Brno University of Technology, Brno, Czech Rep., 2002, pp. 1-154.

[10] 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.

[11] Schwarz, J.: Bayesian evolutionary algorithms applied in decomposition and allocation problems. Habilitation thesis. FIT VUT Brno, 2003, pp.135.

[12] Schwarz, J., Očenášek, J.: Experimental Study: Hypergraph Partitioning Based on the Simple and Advanced Genetic Algorithm BMDA and BOA, In: Proceedings of the Mendel'99 Conference, Brno University of Technology, Faculty of Mechanical Engineering, Brno, 1999, pp. 124-130, ISBN 80-214-1131-7.

[13] Alpert, C. J., Kahng, A. B.: Recent Directions in Netlist Partitioning: A Survey, Integration: The VLSI Journal 19 (1995), pp. 1-81.

[14] Bui, T. N., Moon, B. R.: Genetic Algorithm and Graph Partitioning. IEEE Transactions on Computers, Vol.45, No.7, July 1996, pp. 841-855.

[15] Johnson, D. S., Aragon, C., McGeoch, L., & Schevon, C.: Optimization by Simulated Annealing: An experimental Evaluation, Part 1, Graph Partitioning, Operations Research, Vol.37, pp. 865-892, 1989.

[16] A Benchmark Set, University of California, Los Angeles, VLSI CAD Laboratory, http://vlsicad.cs.ucla.edu/~cheese/benchmarks.html.

[17] Karypis, G., Kumar, V.: Hmetis - A Hypergraph Partitioning Package, version 1.5.3, University of Minnesota, Department of Computer Science&ENGINEERING, Army HPC Research Center Minneapolis, http://www-users.cs.umn.edu/~karypis/metis/hmetis/download.shtml.

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

[19] Ocenasek, J., Pelikan, M.: Parallel spin glass solving in hierarchical Bayesian optimization algorithm. 9th International Conference on Soft Computing, Mendel-2003, Brno, Czech Rep., June 2003, accepted paper.