The Parallel Bayesian Optimization Algorithm

Jiří Očenášek, Josef Schwarz

Brno University of Technology

Faculty of Electrical Engineering and Computer Science

Department of Computer Science and Engineering

CZ - 61266 Brno, Božetěchova 2

e-mail: ocenasek@dcse.fee.vutbr.cz, schwarz@dcse.fee.vutbr.cz

In the last few years there has been a growing interest in the field of Estimation of Distribution Algorithms (EDAs), where crossover and mutation genetic operators are replaced by probability estimation and sampling techniques. The Bayesian Optimization Algorithm incorporates methods for learning Bayesian networks and uses these to model the promising solutions and generate new ones. The aim of this paper is to propose the parallel version of this algorithm, where the optimization time decreases linearly with the number of processors. During the parallel construction of network, the explicit topological ordering of variables is used to keep the model acyclic. The performance of the optimization process seems to be not affected by this constraint and our version of algorithm was successfully tested for the discrete combinatorial problem represented by graph partitioning as well as for deceptive functions.

Introduction

The proposed algorithm belongs to an EDA class of algorithm (Estimation of Distribution Algorithm) [1], based on probability theory and statistics. They use statistical information contained in the set of selected parents to detect gene dependencies. The estimated probability model is used to generate new promising solutions according to this distribution. The process can be described as follows:

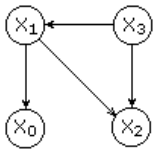
Generate initial population of size **M** (randomly);
Repeat
  Select parent population of **N** individuals according to a selection method (**N** ≤ **M**);
  Estimate the distribution of the selected parents;
  Generate new offspring of size **N'** according to the estimated model;
  Replace some individuals in current population by generated offspring;
Until termination criteria is met

Sequential BOA

BOA (Bayesian Optimization Algorithm) [2] uses Bayesian network (BN) to encode the structure of a problem. In the chromosome of length *n* each gene is treated as a variable and represented by one node in the dependency graph. For each variable X_i it is defined a set of variables Π_{X_i} it depends on, so the distribution of individuals is encoded as

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

Generally, the existence of oriented edge from X_j to X_i in the network implies the belonging of the variable X_j to the set Π_{X_i}. To reduce the space of networks, number of incomming edges into each node is limited to *k*.



$$p(X) = p(X_3) \cdot p(X_1 \mid X_3) \cdot p(X_2 \mid X_3, X_1) \cdot p(X_0 \mid X_1)$$

Fig.1 Example of Bayesian network for joint probability distribution of 4 variables

The Bayessian Dirichlet metric (BD) [3] is used to measure the quality of the network. A special case of BD metric, so-called K2 metric, is used when no prior information about the problem is available.

Many algorithms can be used to build up the network. The optimal search is NP-hard, so in the sequential implementation [4] a simple greedy algorithm was used with only one edge addition in each step. The algorithm starts with an empty network B and for each edge that can be added it computes the K2 metrics of the network B' that can be constructed from B by adding this edge. The edge giving the highest improvement is then added to the network B. This process is repeated until no more addition is possible. By the term 'edge can be added' we mean the test whether the edge keeps the network acyclic, meets the limit of incoming edges and does not belong to the network yet.

After network construction new individuals are generated. First, the variables (genes) are ordered in the topological order and each iteration, the nodes whose parents are already determined are generated using the conditional probabilities. This is repeated until all the variables are generated. Since the sequence of generation should be defined, the dependencies between variables must be acyclic.

Parallel approach

As shown in [2], the overall time to construct the Bayesian network using the greedy search driven by BD metric is $O(k2^k n^2 N+kn^3)$, where n is the length of a chromosome, k is the limit of incoming edges into each node and N is the size of parent population. The time complexity for generating N' new individuals (offspring) is only $O(knN')$, where N' is proportional to the number of parents N. The time complexity for offspring evaluation depends on the complexity of fitness computation itself and in the case of additively-decomposable functions is $O(nN')$.

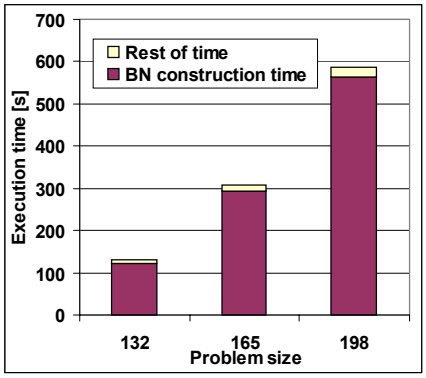


Fig.2 The BOA time complexity profile

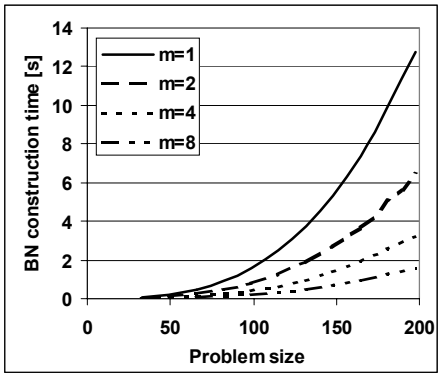


Fig.3 BN construction time for m processors

Fig.2 shows the empirical confirmation of the time-complexity terms stated above: nearly all the execution time of sequential BOA is spent to find the structure of Bayesian network. The remaining time includes fitness computation, parent selection and offspring generation. In comparison to the construction of Bayesian network all of those remaining tasks are easy to be done in parallel, but they take only less than 5% of the overall time. Fig.3 shows the time of Bayesian network construction using one processor of Sun Enterprise 450 server; for the case of m=2,4,8 the time was estimated with no communication overhead considered. Both experiments were done for $f_{3deceptive}$ function [2].

Proposed solution

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. It is possible to use up to n processors, each processor corresponds to one variable and it examines only edges leading to this variable (it has its own local copy of parent population).

The addition of edges is parallel, so we need an additional mechanism to keep the network acyclic. The most simple way how to do it is to predetermine the topological ordering of nodes in advance. At the beginning of each generation, the random permutation of numbers {0,1,…,***n***-1} is created and stored in the ***perm*** array. Each processor generates the same permutation (the initial seed of permutation generator is distributed via set of processors in the initial phase). The direction of all edges in the network should be consistent with the ordering, so the addition of an edge from X_j to X_i is allowed if ***perm***[j] < ***perm***[i]. Evidently, the variable X_i with ***perm***[i]=0 has no predecessor and is forced to be independent, thus the space of possible networks is reduced. To compensate this phenomenon we use new permutation for each generation.

The algorithm can be written as follows:

Start with an empty network B;
Generate the permutation array perm;
for i := 0 **to** (n - 1) **do in parallel**
begin
  while *any edge ending in the variable X_i can be added* **do**
  begin
    for *each possible start of new edge (variable X_j having perm[j]<perm[i])* **do**
    begin
      Compute the local increase of K2 metrics after adding edge (X_j,X_i);
    end
    Add the edge (X_j,X_i) giving the highest improvement to the network B;
  end
end

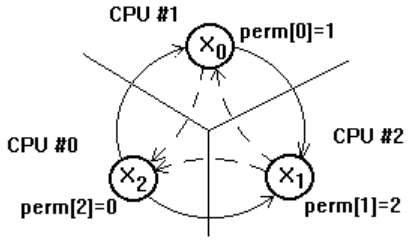


Fig. 4 Example with ***n***=3 and ***perm***=(1,2,0). The dashed edges are not allowed, so the three processors do not have to communicate to keep the network acyclic.

Moreover, the explicit ordering of nodes enables the generation of new individuals in a linear pipeline way. If we use ***n*** processors, then the i-th processor receives from (i-1)-th processor the chromozome with i positions fixed and generates the value of variable X_k, where ***perm***[k] = i.

Time complexity for parallel BOA (PBOA)

When using only m processors ($m \leq n$), the time complexity of the greedy search for a network structure is $O(\lceil n/m \rceil (k2^k nN + kn^2))$ and the time for generating N' new individuals in a pipeline is $O(\lceil n/m \rceil (kn + kN'))$. This algorithm requires no cooperation between processors, so the time estimation shown in Fig.3 is valid. When we use $m=n$ processors, the time complexity is decreased roughly from $O(n^3)$ to $O(n^2)$.

Experimental results

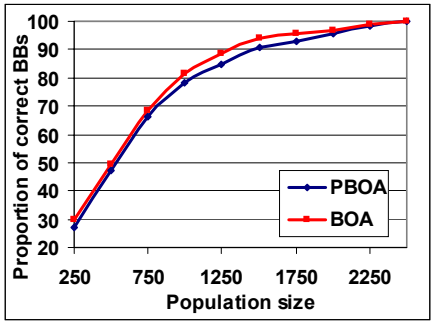


Fig.5 Proportion of correct BBs

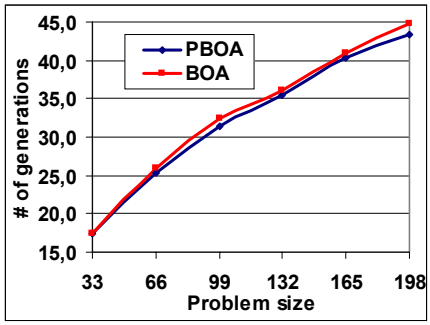


Fig.6 Number of generations until convergence

In Fig.5 and Fig.6 the experimental results for $f_{3deceptive}$ function [2] are shown. The population size needed to get the same quality of solution as well as the number of generations until successful convergence is similar for both BOA and PBOA. The same conclusion we obtained for $f_{5deceptive}$ function [2].

Table 1. The results of grid graphs bisection [5]

Graph name	Grid 100.2 with bottleneck	Grid 100.10
BOA: Min. population size for 90% success	2500	2400
PBOA: Min. population size for 90% success	2600	2500
BOA: Avg. # of generations until convergence	47,9	57,9
PBOA: Avg. # of generations until convergence	51,2	66,3
BOA: Average # of fitness evaluations	119750	138960
PBOA: Average # of fitness evaluations	133120	165750

The values in the Table 1 indicate that for real problems like graph bisectioning the PBOA needs slightly higher population size and number of generations, but this is not critical because PBOA reduces the additional time by the parallel processing too.

Conclusion and future work

This paper is focused on the parallelization of the original (sequential) Bayesian Optimization Algorithm [4] – namely on the parallel construction of Bayesian network. The proposed approach with predetermined topological ordering of BN nodes keeps the network inherently acyclic, so the time complexity of parallel BN construction is linearly reduced by the number of processors. The performance of the algorithm was tested for the discrete combinatorial problem represented by graph bisectioning [5] as well as for deceptive functions [2] and our results show that the capability of finding global optima is really not affected by the used simplification of network construction.

We have also described how to generate new individuals using the linear pipeline architecture with ***n*** processors - each processor is responsible for generation of one variable. We are currently working on a coarse-grained version of PBOA, where the cluster of workstations and message passing techniques are used. Each process receives from other processes the remaining parts of Bayesian network and is responsible for generation, evaluation and distribution of its portion of population. This helps us to overlap the communication latency between workstations.

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" and it was also supported by the Grant Agency of Czech Republic grant No. 102/98/0552 "Research and Application of Heterogeneous Models".

References

[1] Muehlenbein, H., Rodriguez, A. O.: Schemata Distributions and Graphical Models in Evolutionary Optimization. GMD Forschungs Zentrum Informationstechnik, 53754-St. Augustin, 1998, pp.1-21.

[2] Pelikan, M., Goldberg, D. E., Cantú-Paz, E.: Linkage Problem, Distribution Estimation, and Bayesian Networks. IlliGal Report No. 98013, November 1998, pp. 1-25.

[3] Heckerman, D., Geiger, D., Chickering, M.: Learning Bayesian networks: The combination of knowledge and statistical data, Technical Report MSR-TR-94-09, Redmond, Microsoft Research, 1994, pp. 1-53.

[4] Pelikan, M.: A Simple Implementation of Bayesian Optimization Algorithm in C++(Version1.0). Illigal Report 99011, February 1999, pp. 1-16.

[5] Schwarz, J., Očenášek, J.: Experimental study: Hypergraph partitioning based on the simple and advanced genetic algorithms BMDA and BOA, 5th International Mendel Conference, 1999, FME VUT Brno, Czech Republic, pp. 124-130.