TASK QUARTERLY 6 No 3 (2002), 509–521

EVOLUTIONARY ALGORITHM FOR LEARNING BAYESIAN STRUCTURES FROM DATA

MAREK KOZŁOWSKI[1] AND SŁAWOMIR T. WIERZCHOŃ[2,3]

[1] *Faculty of Mathematics and Information Sciences,*
Warsaw University of Technology,
Plac Politechniki 1, 00-661 Warsaw, Poland
kozlowsm@mini.pw.edu.pl

[2] *Institute of Computer Science, Polish Academy of Sciences,*
Ordona 21, 01-267 Warsaw, Poland
stw@ipipan.waw.pl

[3] *Department of Computer Science, Technical University of Bialystok,*
Wiejska 45a, 15-333 Bialystok, Poland

(Received 27 April 2002)

Abstract: In this paper we report an evolutionary approach to learning Bayesian networks from data. We explain reasons, which advocate such a non-deterministic approach. We analyze weaknesses of previous works and come to conclusion that we should operate in the search space native for the problem *i.e.* in the space of directed acyclic graphs instead of standard space of binary strings. This requires adaptation of evolutionary methodology into very specific needs. We propose quite new data representation and implementation of generalized genetic operators and then we present an efficient algorithm capable of learning complex networks without additional assumptions. We discuss results obtained with this algorithm. The approach presented in this paper can be extended with the possibility to absorb some suggestions from experts or obtained by means of data preprocessing.

Keywords: Bayesian networks, structure learning, evolutionary algorithm, discrete optimization

1. Introduction

As stated by Jordan in [1]: *Graphical models are a marriage between probability theory and graph theory. They provide a natural tool for dealing with two problems that occur throughout applied mathematics and engineering – uncertainty and complexity – and in particular they are playing an increasingly important role in the design and analysis of machine learning algorithms. Fundamental to the idea of a graphical model is the notion of modularity – a complex system is built by combining simpler parts. Probability theory provides the glue whereby the parts are combined, ensuring that the system as a whole is consistent and providing ways to interface models to data. The graph theoretic side of graphical models provides both an intuitively appealing interface*

by which humans can model high-interacting sets of variables, as well as data structure that lends itself naturally to the design of efficient general purpose algorithms.

Generally graphical models are graphs in which nodes represent random variables and the arcs specify the (in-)dependence assumptions that must hold between the random variables. Particularly, the lack of arcs represents conditional independence assumptions. Graphical models divide into Markov Random Fields, or undirected graphical models (consult *e.g.* [2] for details) and Bayesian Networks (BN in short) or directed graphical models (consult *e.g.* [3]). While undirected models are popular with the physics and computer vision communities, the directed models are more popular with the artificial intelligence and statistics communities.

Formally speaking a BN is a triple $(X,E,\{P_i\}_{i=1,\ldots,n})$ where $G=(X,E)$ is a directed acyclic graph (DAG for brevity) spanned over the set of n nodes, and P_i, $i=1,\ldots,n$, stands for a conditional probability table over the set $x_i \cup \pi_i$ of variables (π_i is the set of parents of i^{th} node in the graph G). In other words P_i is the conditional probability $P(x_i|\pi_i)$; if a node has no parents, *i.e.* $\pi_i = \oslash$ then P_i is the marginal probability $P(x_i)$. The joint probability distribution, or jpd, over the set of variables $X=\{x_1,\ldots,x_n\}$ can be concisely represented as the product of P_i tables:

$$P(x_1,\ldots,x_n)=\prod_{i=1}^{n} P(x_i|\pi_i). \tag{1}$$

We say that the jpd factorizes according to the above equation.

BNs are attractive for at least two reasons. A directed arc from x_i to x_j has a causal interpretation "x_i causes x_j" or "x_j is an effect of x_i". Hence the set π_i is often interpreted as the set of immediate causes of x_i. Thus, although the formal definition of a BN is based on conditional independence – expressed by the Equation (1) – in practice a BN is typically constructed using notions of cause and effect. Such information can be used as a guide to construct the graph structure. In addition, directed models can encode deterministic relationships, and are easier to learn (*i.e.* fit to data).

The triple $(X,E,\{P_i\}_{i=1,\ldots,n})$, or in fact the tuple $(G,\{P_i\}_{i=1,\ldots,n})$ expresses the dichotomy between qualitative information – represented by the DAG G – and quantitative information represented by the set of parameters P_i, $i=1,\ldots,n$. The latter, much simpler, task can be solved by using standard estimation techniques (consult *e.g.* [4] for details). The first task, of acquiring graphical structure, is the subject of this paper. Equivalently, this task relies upon identification of the set of immediate causes for each variable. Knowing such sets we construct a BN for a given set of variables by drawing arcs from cause variables to their immediate effects. In almost all cases, doing so results in a BN whose conditional independence implications are accurate. In the sequel we will assume that a problem domain is just the set of variables $X=\{x_1,\ldots,x_n\}$ and each x_i is a discrete random variable (for BNs with continuous variables see [5]).

The paper is organized as follows. In Section 2 we shortly summarize previous research on learning Bayesian networks. A "standard" genetic approach to the problem and some of its disadvantages are presented in Section 3. To improve these disadvantages a new evolutionary methodology is proposed in Section 4. In Section 5

we present and discuss the results obtained with our algorithm. Finally, in Section 6, we summarize the paper, propose some of the possible further extensions of the proposed approach and define open problems for further research.

2. Learning Bayesian networks

A naive approach to identify the graphical structure G is simply to ask a domain expert. When the number of nodes, n, increases, this task becomes more and more unreliable. The cardinality of the space of all possible structures (search space) is given by the formula (consult [6]):

$$f(n)=\sum_{i=1}^{n}(-1)^{i+1}\binom{n}{i}2^{i(n-i)}f(n-i),\quad f(0)=1,\quad f(1)=1.$$

For instance $f(3)=25$ and $f(5)=29\,280$. This number can be dramatically reduced if an ordering among the nodes is assumed. In this case the cardinality of the search space reduces to $2^{n(n-1)/2}$. However, identification of such an ordering is not a trivial task. Thus, in general leaning BN from the data sets is an NP-complete problem (consult *e.g.* [7]).

Most of the literature on acquiring BNs structures from data is concerned with what statisticians call model selection. This approach relies upon selecting a "good" model from among all possible models, and use it as if it were the correct model. Of course, to be able to select such a model we need a scoring criterion. Now, prior knowledge, a database and a set of graph structures are taken and the degree to which a graphical structure fits the prior knowledge and data is computed according to the assumed criterion.

A natural scoring criterion for a graphical structure seems to be the (relative) posterior probability of that structure given the database – consult Heckerman's tutorial [4] for details. A successful illustration of such an approach is the CH metrics introduced in [8]:

$$\mathrm{CH}(\mathit{struct})=-\log P(\mathit{data}|\mathit{struct})=\sum_{i=1}^{n}\left(\sum_{j=1}^{q_i}\left(\sum_{k=1}^{r_i}\log(N_{ijk})-\sum_{k=r_i}^{N_{ij}+r_i-1}\log(k)\right)\right),\quad(2)$$

where n – number of variables, q_i – number of unique instances (values) of the parents π_i of i^{th} variable, r_i – number of possible values of i^{th} variable, N_{ijk} – number of cased in which the variable i takes k^{th} value and π_j takes its j^{th} value, $N_{ij}=\sum_k N_{ijk}$.

A nice feature of this criterion is its separability, *i.e.* it can be written as $\mathrm{CH}(\mathit{struct})=\sum_{i=1,\ldots,n}s(x_i|\pi_i)$ where $s(x_i|\pi_i)$ is only a function of x_i and its parents. This means that $\mathrm{CH}(\mathit{struct})$ can be computed locally (*e.g.* in parallel).

Unfortunately, because of large cardinality of the search space, this criterion cannot be applied immediately. To simplify the problem, the authors proposed a greedy search algorithm, called K2, that uses "cognitive ordering" over the set of variables.

A review of other approaches to learning BNs can be found in [9, 4]. Intriguingly, the problem of network identification can be treated also as a discrete optimization problem. Thus, given a scoring criterion, we will try to adapt non-deterministic

evolutionary techniques (derivative of the genetics algorithms) to solve the problem. We have decided to choose such an approach for the following reasons:

- evolutionary techniques proved to be very efficient in solving many NP-hard discrete optimization problems,
- deterministic methods for NP-hard problems can suffer for local optima,
- using evolutionary techniques we can avoid specific additional assumptions that are required for many other approaches,
- we think of adaptation of the experts' knowledge if any is given,
- non-deterministic algorithm can generate acceptable but different structures during successive runs; we can choose the best one or think of some kind of "averaging" over them.

In our study it is supposed that a number of "standard" assumptions (*cf.* [4] or [10]) holds:

(a1) the database D is a multinomial sample from some network structure S with the set of parameters $\{P_i\}$,

(a2) for each network, the parameters associated with one node are independent of the parameters associated with other nodes (global independence), and the parameters associated within a node given one instance of its parents are independent of the parameters of that node given other instances of its parent nodes,

(a3) if a node has the same parents in two distinct networks then the distribution of the parameters associated with this node is identical in both networks (parameters modularity),

(a4) each case is complete,

(a5) the distribution of the parameters associated with each node is Dirichlet.

For the future reference let us introduce a technical assumption:

(a6) the cardinality of parents sets π_i, $i=1,\ldots,n$ does not exceed a fixed number m.

3. Genetic algorithms applied for learning BBNs

Our research has been inspired by the study of Larrañaga and his co-workers [11] in which the authors use standard genetic algorithm (GA) for learning BNs with additional assumption that $m=4$.

Since standard GA operates on binary strings (consult [12] for details), a BN consisting of n nodes is represented as a string of length n^2 resulting from the concatenation of the rows of the incidence matrix. The (i,j) entry of such a matrix equals 1 if j is a parent of i and zero otherwise. Thus a BN from Figure 1 is represented by the string 000100110.

As a scoring metric the authors applied CH criterion (2).

The algorithm proposed in [11] uses simple ranking selection, standard mutation (alternation of the bits of an individual with probability p_M) and a standard, single-point crossover with probability p_C.

Unfortunately an offspring may not satisfy the assumption (a6) or may not be a DAG. So a *reduction* operator choosing the best m nodes from the parent set of every variable and a *repair* operator converting illegal structures into DAGs by

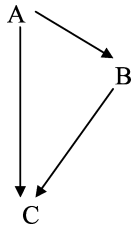

Figure 1. Sample Bayesian network structure

dropping random edges must be used afterwards. The authors point out that reduction should be done by choosing the best subset of parent nodes instead of dropping them randomly. Conversion into DAGs is more complex – the cycles may have common edges and it is difficult to find an efficient algorithm for such operation. So the edges are thrown away according to a random value.

However, the algorithm equipped with the two additional operators behaves rather poorly, *i.e.* it slowly converges to rather local optima in the case of complex graphical structures. It seems that such behavior of the algorithm is caused mainly by rather artificial transformation of the original search space (*i.e.* the space of all DAGs with n nodes) into n^2-Hamming space (*i.e.* collection of binary strings of length n^2). We think that the structures of BNs cannot be represented by binary or other strings, or operators on such strings cannot lead out to better structures because:

- Those will often result in cyclic structures. Repair operators tend to increase the "noise" (they change structures as drastically as the genetic operators). If we want to preserve convergence, we need to use very small values for probabilities of genetic operators or very "conservative" selection strategies. The authors propose a so-called ELITIST criterion: an existing population is extended with offspring (the size of the population is doubled) and then it is reduced to the initial size by dropping the worst individuals.
- Crossing over such strings depends on the ordering of variables (their parent sets) in the individual; especially it should be distinct if we use single-point crossover. Schemata Theorem (consult [12]) does not work in this case.
- Mutation alternates every bit with some probability p_M. For n variables, every variable has $(n-1)$ possible parents. Unfortunately we are interested only in graphs with nodes having at most m parents. This implies that in every row of the incidence matrix there are at most m bits equal to 1 and at least $(n-m)$ bits equal to 0. For quite big values of n we have "sparse" (containing mostly 0s) individuals. So, during mutation we generally add some new arcs to the graph and then delete some of the arcs in the offspring by the reduction operator. Then we convert the structure into DAG by randomly dropping some arcs. Hence we can think of mutation as extending some parent sets and then reducing them, firstly deterministically (by reducing their sizes) then non-deterministically (by converting a new structure into DAG). Furthermore, individuals tend to converge to $m*n$ arcs because mutation increases the number of edges. How about networks with many nodes and few arcs?
- We have carried out some experiments with a program written according to indications of the authors of the algorithm with the same parameters and with

the same database. Then we ran some experiments with the same parameters but without crossover. The results obtained were the same in both cases. This confirms that the algorithm in fact works as described above.

4. Genesis of a new algorithm

It is clear that a different approach should be used: instead of adapting the problem to the requirements of GAs let us try to adapt the very idea of this methodology to our needs. Let us reconsider the nature of genetic operators first.

Mutation should slightly disorder the structure – in other words – when mutating we expect that an individual will be replaced with one of its neighbors (according to some definition of neighborhood). That is the mutation operator is responsible for exploration of the close neighborhood of the individuals in a population. Simple modification of parents of single variable can considerably affect the value of the CH metric. So mutation that usually changes parents of at most one variable is preferred. It requires shorter representation of individuals or smaller value of p_M. Smaller value of p_M results in that changing more than one parent of the variable at once gets almost impossible. Shorter representation seems to have no such disadvantages.

Crossover should produce new solutions based on given structures and thus explore new areas of the search space. It is desired that offsprings have some properties or components of both parents (individuals producing offspring). In our problem "component" means a parent set π_j for given variable x_j. Because of the restrictions (only DAGs are allowed), the hypothesis of building blocks (a corollary from Holland's Schemata Theorem) doesn't work in our case, at least in the simple, binary form. The crossover should not take into account any ordering given *a priori*. That is if we encode DAG in the form of some linear representation (a string), we decide to order somehow the variables – this ordering should not impact crossover. That condition is not fulfilled by the crossover proposed in the standard algorithm described above.

Avoiding creation of cyclic structures during both mutation and crossover is strongly preferable. For faster implementation we also want those operators to run as parallel as possible for distinct variables of the individual. In addition, we can think of such implementations of those operators that are able to absorb some suggestions from experts, *i.e.* to explore more carefully some areas of the search space without precocious convergence of the population to solutions in those areas.

Selection should result in preferring some areas of the search space by throwing away worse and increasing the number of better individuals. To carefully explore the search space the population cannot converge too quickly to one region of the space, which usually causes convergence to local minimum.

If we have an assumption: no more than $m = 4$ parents for any variable, we can use arrays $n \times 4$ instead of $n \times n$. For any variable we keep only numbers (names) of its parents. If there are fewer than 4 parents we fulfill rows with NULL symbols. We will treat those NULLs in the same way as "normal" parents.

With this new representation it is possible to modify parents of each node by random exchange of some fields from the row corresponding to that node by some allowed new values computed separately in every iteration. We can infer which variables are allowed and which are not, we can consider some parents more desired

and some less for any variable. To do so we need some (partial) ordering of the nodes of a DAG; precise definition of this ordering is given in Subsection 4.1. So we are able to suggest some fragments of structures. Finally, we can create those sets of possible parents with avoidance of cycles in mind.

Similarly, we can introduce new and "safe" (*i.e.* resulting in acyclic structures) crossover operator. It is implemented as follows. Given two randomly chosen parent structures, choose randomly a node in the first parent. For every parent of this node we leave its predecessors untouched, while parents of remaining nodes are replaced by the parents from the second structure. Let us note that such an operator: (a) does not produce cyclic structures, and (b) uses the natural precedence ordering instead of the abstract nodes ordering forced by the binary representation of DAGs. Figure 2 illustrates the operator.

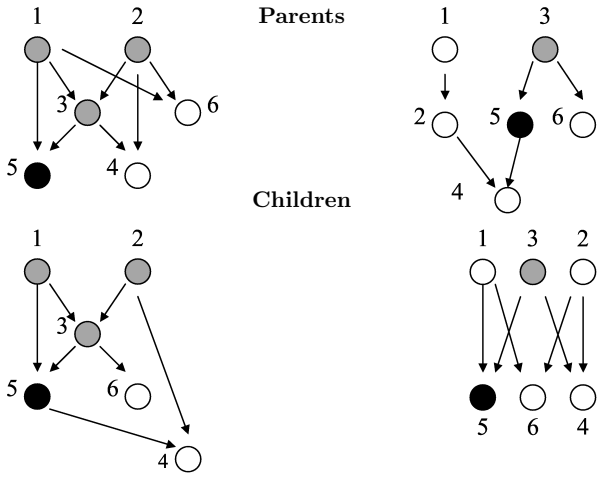

Figure 2. Crossover operator. The variable no. 5 (marked black) has been selected. Its predecessors in both parent individuals are marked dark grey. Those variables (their parents) become untouched, while for other variables (the white ones) we take whole parent sets from the other individual

It is also worth saying that if we cache components of the CH criterion for every variable we do not need to compute it again after crossover. This makes that operator extremely fast.

4.1. A Partial Order in a DAG

Let an i^{th} node be a parent of the node j in a DAG. It is obvious that:

1. the longest incoming path is shorter for the i^{th} node than for the j^{th} node,
2. the longest outgoing path is longer for the i^{th} node than for the j^{th} node.

This means that it is possible to partition nodes of any DAG into disjoint subsets (layers) in two ways: considering the longest paths incoming or outgoing from those nodes. Let us denote labels of those layers according to the length of those paths. Let nodes with no parents always belong to the 0 layer *i.e.* for the ordering

according to incoming paths – the layer number is equal to the longest path incoming to the node, while for the "outgoing ordering" it is equal to:

(the longest path in the DAG – the longest path outgoing from the node)

Figure 3 shows that those two partitions can result in different subsets for the same DAG.

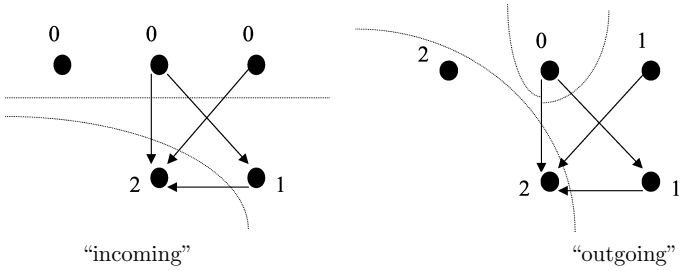

Figure 3. Nodes of the graph divided into "layers" according to "incoming" and "outgoing" ordering

The main difference can be observed for those nodes that are separated (have no edges).

Let us notice that if for any node in layer l ($l>0$) we add a parent belonging to the "previous" layer $l'<l$, it won't result in a cycle. This means that both of those divisions are some partial orderings of the nodes. Thus our new mutation operator can be described as follows: nodes allowed to be parents of any node i are simply nodes belonging to "previous" layers (including some number of NULL symbols).

4.2. The algorithm

The pseudocode given below summarizes our remarks on a new approach to learning BNs.

0. Set $i=0$.
1. We generate randomly *popSize* individuals (0^{th} population).
2. Selection. Create an auxiliary population $(i+1)^{\text{th}}$ by copying some individuals form the i^{th} population.
3. Crossover (performed with probability p_C):
   (a) choose (randomly) a pair of parent structures,
   (b) select randomly a node, say n,
   (c) do not change parent nodes of n and nodes being predecessors of n; replace parents of the remaining nodes of the first (second) structure by the parents of these nodes in the second (first) structure,
   (d) apply "avoid local minimum" operator after crossover, *i.e.* select (with probability p_E) two nodes and exchange them (*e.g.* 1^{st} node becomes 5^{th} node and 5^{th} node becomes 1^{st} node); this operator resembles somehow the "shaking" procedure of stochastic annealing. Every individual is mutated then, we calculate adequate component of the Cooper-Herskovits formula.
4. Mutation:
   (a) calculate layers using one (randomly selected) division criterion; next steps are executed for every node separately (it means concurrently),

(b) create a list, *i.e.* select all nodes belonging to lower layers than a given node and add some NULLs (we suggest that half of that list should contain NULLs – the number of edges won't change too much),

(c) for every parent of the node (including NULL parents) we generate a random value – if it's lower than p_M we exchange that parent with randomly selected node (or NULL) from the list created in the previous step (4b).

5. Calculate adequate component of the Cooper-Herskovits formula.
6. Go to step 2.

5. Results

To verify the algorithm from Section 4.2 a number of experiments have been performed. All experiments were run on the database generated form the ALARM network introduced by Beinlich in [13] and presented in Figure 4. It is a complex structure consisting of 37 variables, very hard to be learned. That database was chosen for comparative reasons, since it is a very popular model for testing algorithms working on BNs. It was also used by Larrañaga *et al.* for testing their genetic approach. For the proper structure of the network the value of CH criterion should be equal to 14 407[1] (the less the better). For a random structure it is equal to $\sim 23\,000$. Structures with the value below 14 440 are quite acceptable.

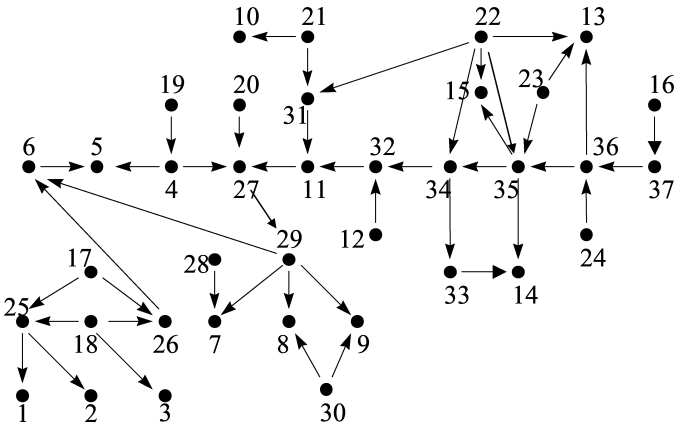

Figure 4. Structure of the ALARM network

The values of the parameters p_M, p_C, and p_E have been selected experimentally. Our algorithm usually finds best results before 3000 iterations (see Figure 5), sometimes those are a little improved later so we agreed on the "safe" value 5000 iterations.

The algorithm is sensitive to probabilities of evolutionary operators, especially the probability of mutation. Good values for p_C and p_E vary from 0.3 to 0.5 without significant decrease of the results obtained. We found the best results for the p_M equal to 0.002–0.003, for the value of 0.004 those were significantly worse. Fortunately

1. We operate on the dataset generated by ourselves and that value is slightly different to the analogous value for the Larrañaga's dataset which was 14 412

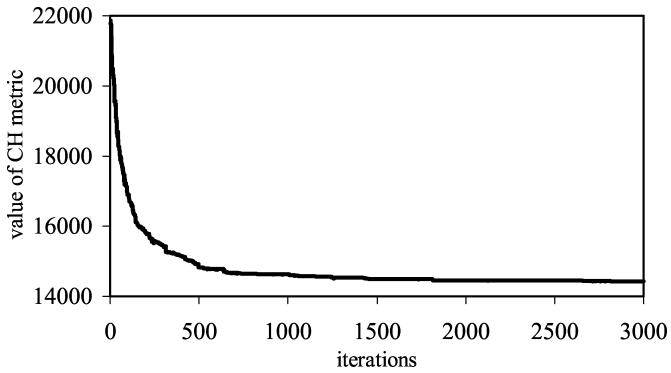

Figure 5. Best values of fitting function in successive iterations

it is quite easy to set this parameter to the right value by gradually decrementing it (starting with or example 0.05) and observing the results obtained in successive runs. These observations confirm the error-threshold phenomena described in the evolutionary biology literature, *cf. e.g.* [14] or [15]. The error threshold is a critical (and rather low) mutation rate at which the population evolutionary dynamics changes radically. There exists a phase transition between an "ordered" (selection-dominated) regime and "disordered" (mutation-dominated) one. Mutation rates above this critical value cause a loss of the genetic information gained so far.

We have run tests on different population sizes. Populations smaller than 40 individuals tend to be a little "unstable", which means that sometimes considerably worse or better results happen. Populations of the size 80 result in comparable but sub-optimal structures (*cf.* Table 1). Populations twice bigger give much better solutions in the sense of the value of the CH criterion (*cf.* Table 2). Again this observation is in accordance with evolutionary biology. In Eigen's quasispecies theory [14], it has been observed that in the case of small population size, the evolution process is essentially stochastic, while for very large population size the evolution becomes a deterministic process which can be described in terms of the ordinary differential equations. When the population size is infinite then: (1) the evolution process always converges, and (2) the final population is a quasispecie, that is the distribution of the sequences in the neighborhood of the master sequence (optimal solution).

Table 1. Results obtained with the population of 80 individuals, where:

min absolute – minimal value found
generation – generation in which the minimal value was found
min last – minimal value of the last population
avg. last – average value of the last population

$PopSize = 80$, $p_C = 0.4$, $p_M = 0.003$, $p_E = 0.4$				
run	1	2	3	4
min absolute	14426	14418	14416	14424
generation	2826	2563	1756	2127
min last	14426	14418	14417	14424
avg. last	14548	14538	14580	14525

Table 2. Results obtained with the population of 160 individuals

$PopSize = 160$, $p_C = 0.4$, $p_M = 0.003$, $p_E = 0.4$				
run	1	2	3	4
min absolute	14404	14408	14414	14416
generation	3990	4848	2853	661
min last	14404	14408	14417	14416
avg. last	14535	14505	14520	14517

This does not mean that the corresponding structures are closer to the desired ones in the sense of the Hamming distance or in the sense of correct parent sets. This is because changing one arc causes changing some arcs in the neighborhood – the criterion selected prefers structures that can better factorize the joint distribution function (and then allow proper knowledge propagation) instead of searching for as many as possible "cause-effort" relations. So we can observe some "regions" of disordered arcs. For example in Figure 7 such region are variables 4, 19, 20, 27, and in both underlying structures nodes 17 and 24.

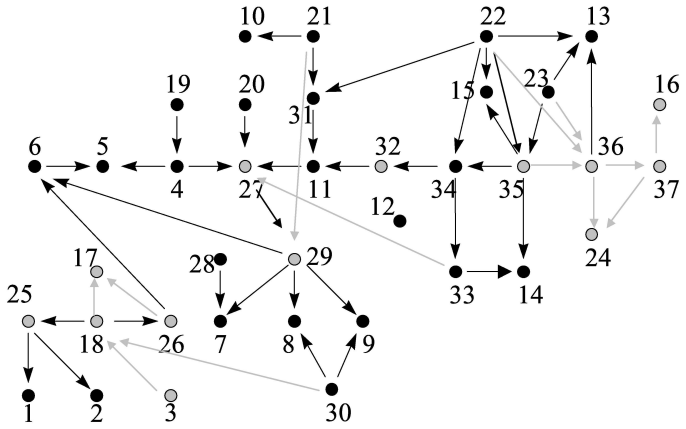

Figure 6. Sample structure found (CH = 14408)

The same erroneous arcs in both structures: $18 \rightarrow 17$, $26 \rightarrow 17$, $36 \rightarrow 24$, $37 \rightarrow 24$, $33 \rightarrow 27$ can also suggest that a larger dataset should be used.

For the population size of 200 our algorithm 3 times of 4 found structures with CH = 14407, two of them were exactly the structures we were searching for. For the population of 80 our algorithm works about 20–30 minutes on the Celeron400/256 MB RAM. For larger populations – proportionally longer.

6. Conclusions and further research

In this paper we have proposed a new approach to learning Bayesian networks. We adapted evolutionary methodology to specific requirements of the problem being solved. We showed fast and efficient implementation of genetic operators on DAG structures. Thanks to this we are able to learn complex Bayesian network structures without difficult to be fulfilled assumptions, for example about linear ordering of

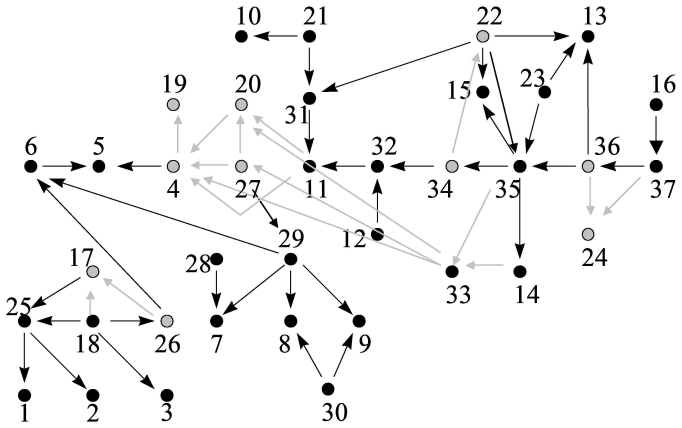

Figure 7. Sample structure found (CH = 14 425)

variables. In this way we proposed how to simply dynamically divide the set of variables into 'classes' and thus define another, natural (non-linear) ordering.

The mutation proposed by us is not only efficient and simple but thanks to it we can further develop our approach toward absorption of experts' suggestions. In our algorithm the very basic part of the individual is now not an arc (or its absence) between two nodes, but a parent set of given variable. Our operators work on such parent sets. During mutation we slightly change parent sets of distinct variables. According to suggestions of experts or any preprocessing (based on, for example, computation of described by Pearl and Rebane weights of arcs) we can prepare some "preferences". Some parents (if available) can be more probable, some - less. Given such suggestions we can go further and implement mutation as changing a whole parent set according to the suggestions mentioned above and nodes availability (according to division into layers). We plan to concentrate on that and on the possibilities of such "suggesting" in our further research.

References

[1] Jordan M I (Ed.) 1998 *Learning in Graphical Models*, The MIT Press, London
[2] Lauritzen S L 1996 *Graphical Models*, Oxford University Press
[3] Pearl J 1986 *Artificial Intelligence* **29** 241
[4] Heckerman D in [1] pp. 301–354
[5] Cowell R G, David A P, Lauritzen S L and Spegelhalter D J 1999 *Probabilistic Networks and Expert Systems*, Springer-Verlag, New York
[6] Robinson R W 1997 *Lecture Notes in Mathematics 622: Combinatorial Mathematics V* (Little C H C, Ed.), Springer-Verlag, pp. 28–43
[7] Chickering D M, Geiger D and Heckerman D 1994 *Technical Report* MSR-TR-94-17, Redmond WA: Microsoft Research
[8] Cooper G F and Herskovits E 1992 *Machine Learning* **9** 309
[9] Buntine W 1996 *IEEE Trans. on Knowledge and Data Engineering* **8** 195
[10] Geiger D and Heckerman D 1995 *Uncertainty in Artificial Intelligence* 11 (Besnard P and Hanks S, Eds.), Morgan Kaufmann, pp. 196–207
[11] Larrañaga P, Poza M, Yurramendi Y, Murga R H and Kuijpers C M H 1996 *IEEE Trans. on Pattern Analysis and Machine Intelligence* **18** 912
[12] Goldberg D E 1989 *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, Mass.: Addison-Wesley

[13] Beinlich I A, Suermondt H J, Chavez R M and Cooper G F 1989 *Proc. 2nd European Conf. on Artificial Intelligence in Medicine*, London, Springer-Verlag, pp. 247–256
[14] Eigen M 1971 *Naturwissenschaften* **58** 465
[15] Peliti L *Introduction to the Statistical Theory of Darwinian Evolution*, arXiv:cond-mat/9712027