Independence Model estimation using Artificial Evolution

Olivier Barrière, Evelyne Lutton, Pierre-Henri Wuillemin

Independence Model estimation using Artificial Evolution

Estimation de modèle d'indépendance par évolution artificielle

Olivier Barrière*, Evelyne Lutton**,
Pierre-Henri Wuillemin***

* Université de Montréal, Faculté de Pharmacie,
2940 chemin de Polytechnique, H3T 1J4 Montréal
olivier.barriere@umontreal.ca
** INRIA - Saclay-Ile-de-France,
AVIZ team, Bt 490, Université Paris-Sud,
91405 Orsay Cedex, France.
Evelyne.Lutton@inria.fr
*** LIP6-CNRS UMR7606,
75016 Paris, France
Pierre-Henri.Wuillemin@lip6.fr

Abstract:

In this paper, we consider a Bayesian network structure estimation problem as a two step problem based on an independence model representation. We first perform an evolutionary search for an approximation of an independence model. A deterministic algorithm is then used to deduce a Bayesian network which represents the equivalence class of the independence model. This paper is a shortened version of a paper that has been published in a genetic algorithms conference [2].

Résumé :

Nous abordons dans cet article le problème de l'estimation de la structure d'un réseau Bayésien en deux étapes, à partir d'un représentation sous forme de modèle d'indépendance. Nous considérons d'abord le problème de la recherche d'un modèle d'indépendance approché à l'aide d'un algorithme évolutionnaire. Un algorithme déterministe est ensuite employé pour déduire un représentant du modèle d'indépendance. Cet article est une version condensée d'une précédente publication présentée dans une conférence sur les algorithmes génétiques [2].

1 Introduction

Bayesian networks structure learning is a NP-Hard problem [7], which has applications in many domains, as soon as one tries to analyse a large set of samples in terms of statistical dependence or causal relationship. In agrifood industries for example, the analysis of experimental data using Bayesian networks helps to gather technical expert knowledge and know-how on complex processes [3]. In a comparative study by O. Francois and P. Leray [11], authors present various approaches used to cope with the problem of structure learning: *PC* [24] or *IC/IC** [21] (causality search using statistical tests to evaluate conditional independence), *BN Power Constructor (BNPC)* [5] (also uses conditional independence tests) and other methods based on scoring criterion, such as *Minimal weight spanning tree (MWST)* [8] (intelligent weighting of the edges and application of the well-known algorithms for the problem of the minimal weight tree), *K2* [9] (maximisation of $P(G|D)$ using Bayes and a topological order on the nodes), *Greedy search* [6] (finding the best neighbour and iterate) or *SEM* [12] (extension of the EM meta-algorithm to the structure learning problem).

Evolutionary techniques were also used to solve the Bayesian network structure learning problem, but were facing crucial problems like BN representation (an individual being a whole structure like in [16], or a sub-structures like in [20]), or fitness function choice [20]. Various strategies were used, based on evolutionary programming [25], immune algorithms [15], multi-objective strategies [23], lamarkian evolution [26] or hybrid evolution [27].

We propose here to use an alternate representation, independence models, in order to solve the Bayesian network structure learning in two steps. Independence model learning is still a combinatorial problem, but it is easier to embed within an evolutionary algorithm.

The paper is organised as follows. Section 2 gives some basics about Independence Models and describes the PC estimation algorithm that is used for comparison in this paper. Then, section 3 sketches the components of the evolutionary algorithm that is used in the first step of the algorithm (named IMPEA). The second step of IMPEA is detailed in section 4. Experimental analysis has been developed on two test-cases (section 5). Conclusions and further works are sketched in section 6.

2 Independence models

In this work, we consider *Independence Models* (IMs), which can be seen as underlying models of Bayesian networks. They are defined as follows:

- Let N be a non-empty set of variables, then $T(N)$ denotes the collection of all triplets $\langle X, Y|Z\rangle$ of disjoint subsets of N, $X \neq \emptyset$ and $Y \neq \emptyset$. The class of elementary triplets $E(N)$ consists of $\langle x, y|Z\rangle \in T(N)$, where $x, y \in N$ are distinct and $Z \subset N \setminus \{x, y\}$.
- Let P be a joint probability distribution over N and $\langle X, Y|Z\rangle \in T(N)$. $\langle X, Y|Z\rangle$ is called an *independence statement* (IS) if X is conditionally independent of Y given Z with respect to P (i.e., $X \perp\!\!\!\perp Y \mid Z$)
- An independence model (IM) is a subset of $T(N)$: each probability distribution P defines an IM, namely, the model $\{\langle X, Y|Z\rangle \in T(N)\ ;\ X \perp\!\!\!\perp Y \mid Z\}$, called the *independence model* induced by P.

To summarize, an independence model is the set of all the independence statements, that is the set of all $\langle X, Y|Z\rangle$ satisfied by P, and different Markov-equivalent Bayesian networks induce the same independence model. By following the paths in a Bayesian network, it is possible (even though it can be combinatorial) to find a part of its independence model using algorithms based on directional separation (d-separation) or moralization criteria. Reciprocally, an independence model is a guide to produce the structure of a Bayesian network.

PC, the reference causal discovery algorithm, was introduced by Sprites, Glymour and Scheines in 1993 [24]. A similar algorithm, IC, was proposed simultaneously by Pearl and Verma [21]. It is based on chi-square tests to evaluate the conditional independence between two nodes. It is then possible to rebuild the structure of the network from the set of discovered conditional independences. PC algorithm actually starts from a fully connected network and every time a conditional independence is detected, the corresponding edge is removed.

The complexity of this algorithm depends on N, the size of the network and k, the upper bound on the fan-in, and is equal to $O(N^k)$. In practice, this implies that the value of k must remain very small when dealing with big networks.

Consequently, as the problem of finding an independence model can be turned to an optimisation problem, we investigate here the use of an evolutionary algorithm. More precisely, we build an algorithm that let a population of triplets $\langle X, Y|Z\rangle$ evolve until the whole population comes near to the independence model.

3 Evolutionary estimation of an Independence Model

Our algorithm (Independence Model Parisian Evolutionary Algorithm - IMPEA) is a *Parisian* cooperative co-evolution. This scheme corresponds to a rather recent way of fomulating the resolution of problems in the EA community [1]. It has already been used with success for example in real-time evolutionary algorithms, such as the *flies* algorithm [17].

IMPEA is actually a two steps algorithm. First, it generates a subset of the independence model of a Bayesian network from data by evolving elementary triplets $\langle x, y|Z\rangle$, where x and y are two distinct nodes and Z is a subset of the other ones, possibly empty. Then, it uses the independence statements that it found at the first step to build the structure of a representative network.

3.1 Search space and local fitness

Individuals are elementary triplets $\langle x, y|Z\rangle$. Each individual is evaluated through a chi-square test of independence which tests the null hypothesis H_0: "The nodes x and y are independent given Z". The chi-square statistic χ^2 is calculated by finding the difference between each observed O_i and theoretical E_i frequencies for each of the n possible outcomes, squaring them, dividing each by the theoretical frequency, and taking the sum of the results: $\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$. The chi-square statistic can then be used to calculate a *p-value* p by comparing the value of the statistic χ^2 to a chi-square distribution with $n-1$ degrees of freedom.

p represents the probability to make a mistake if the null hypothesis is not accepted. It is then compared to a significance level α (0.05 is often chosen as a cut-off for significance) and finally the independence is rejected if $p < \alpha$. Given that the higher the p-value, the stronger the independence, p seems to be a good candidate to represent the local fitness (which measure the quality of individuals). Nevertheless, this fitness suffers from two drawbacks:

- When dealing with small datasets, individuals with long constraining set Z tends to have good p-values only because dataset is too small to get enough samples to test efficiently the statement $x \perp\!\!\!\perp y \mid Z$.
- Due to the exponential behaviour of the chi-square distribution, its tail vanishes so quickly that individuals with poor p-values are often rounded to 0, making then indistinguishable.

[1] The searched solution is embedded in several individuals, or even in the whole population, instead of being embedded in a single individual. This allow to design in some cases very efficient evolutionary algorithms.

First, p has to be adjusted in order to promote independence statements with small Z. This is achieved by setting up a parsimony term as a positive multiplicative malus $parcim(\#Z)$ which decrease with $\#Z$, the number of nodes in Z. Then, when $p < \alpha$ we replace the exponential tail with something that tends to zero slower. This modification of the fitness landscape allows to avoid *plateaus* which would prevent the genetic algorithm to travel all over the search space[2]:

$$AdjLocalFitness = \begin{cases} p \times parcim(\#Z) & if\ p \geq \alpha \\ \alpha \times parcim(\#Z) \times \frac{X_\alpha^2}{X^2} & if\ p < \alpha \end{cases}$$

3.2 Genetic operators

The genome of an individual, being $\langle x, y|Z\rangle$ where x and y are simple nodes and Z is a set of nodes is straightforward: It consists in an array of three cells (see figure 1), the first one containing the index of the node x, the second cell containing the index of y and the last one is the array of the indexes of the nodes in Z.

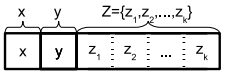

Figure 1: Representation of $\langle x, y|Z\rangle$

This coding implies specific genetic operators because of the constraints resting upon a chromosome: there must not be doubles appearing when doing mutations or crossovers. A quick-and-dirty solution would have been to first apply classical genetic operators and then apply a *repair operator* a posteriori. Instead, we propose wise operators (which do not create doubles), namely two types of mutations and a robust crossover.

- Genome content mutation

  This mutation operator involves a probability p_{mG} that an arbitrary node will be changed from its original state. In order to avoid the creation of doubles, this node can be muted into any nodes in N except the other nodes of the individual, but including itself (see figure 2).

- Add/remove mutation

  The previous mutation randomly modifies the content of the individuals, but does not modify the length of the constraining set Z. We introduce a new mutation operator called *add/remove mutation*, represented on figure 3, that allows to randomly add or remove nodes in Z. If this type of mutation is selected, with probability P_{mAR}, then new random nodes are either added with a probability P_{mAdd} or removed with $1-P_{mAdd}$. These probabilities can vary along generations. Moreover, the minimal and the maximal number of nodes allowed in Z can evolve as well along generations, allowing to tune the growth of Z.

- Crossover

  The crossover consists in a simple swapping mechanism on x, y and Z. Two individuals $\langle x, y|Z\rangle$ and $\langle x', y'|Z'\rangle$ can exchange x or y with probability p_{cXY} and Z with probability p_{cZ} (see figure 4). When a crossover occurs, only one swapping among $x \leftrightarrow x'$, $y \leftrightarrow y'$, $x \leftrightarrow y'$, $y \leftrightarrow x'$ and $Z \leftrightarrow Z'$ is selected via a wheel mechanism which implies that $4p_{cXY} + p_{cZ} = 1$. If the exchange is impossible, then the problematic nodes are automatically muted in order to keep clear of doubles.

[2] *Note:* This can be viewed as an "Ockham's Razor" argument.

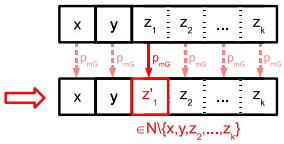

Figure 2: Genome content mutation

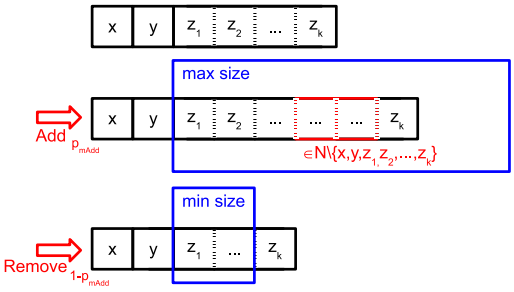

Figure 3: Add/remove mutation

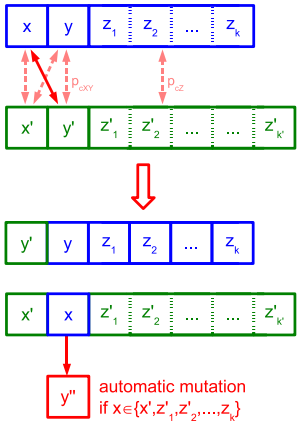

Figure 4: Robust crossover

3.3 Sharing

So as not to converge to a single optimum, but enable the genetic algorithm to identify multiple optima, we use a sharing mechanism that maintains diversity within the population by creating *ecological niches*. The complete scheme is described in [10] and is based on the fact that fitness is considered as a shared resource, that is to say that individuals having too many neighbours are penalized. Thus we need a way to compute the distance between individuals so that we can count the number of neighbours of a given individual. A simple Hamming distance was chosen: two elementary triplets $\langle x, y|Z\rangle$ and $\langle x', y'|Z'\rangle$ are said to be neighbours if they test the same two nodes (i.e., $\{x, y\} = \{x', y'\}$), whatever Z. Finally, dividing the fitness of each individual by the number of its neighbours would result in sharing the population into sub-populations whose size is proportional to the height of the peak they are colonising [14]. Instead, we take into account the relative importance of an individual with respect to its neighbourhood, and the fitness of each individual is divided by the sum of the fitnesses of its neighbours [18]. This scheme allows one to equilibrate the sub-populations within peaks, whatever their height.

3.4 Immortal archive and embossing points

Recall that the aim of IMPEA is to construct a subset of the independence model, and thus the more independence statements we get, the better. Using a classical Parisian Evolutionary Algorithm scheme would allow to evolve a number of independence statements equal to the population size. In order to be able to evolve larger independence statements sets, IMPEA implements an *immortal archive* that gather the best individuals found so far. An individual $\langle x, y|Z\rangle$ can become immortal if any of the following rules applies:

- Its p-value is equal to 1 (or numerically greater than $1 - \epsilon$, where ϵ is the precision of the computer)
- Its p-value is greater than the significance level and $Z = \emptyset$
- Its p-value is greater than the significance level and $\langle x, y|\emptyset\rangle$ is already immortal

This archive serves two purposes: the most obvious one is that at the end of the generations, not only we get all the individuals of the current population but also all the immortal individuals, which can make a huge difference. But this archive also plays a very important role as *embossing points*: when computing the sharing coefficient, immortal individuals that are not in the current population are added to the neighbours counting. Therefore a region of the search space that has already been explored but that has disappeared from the current population is *marked as explored* since immortal individuals count as neighbours and thus penalize this region, encouraging the exploration of other zones.

3.5 Clustering and partial restart

Despite the sharing mechanism, we observed experimentally that some individuals became over-represented within the population. Therefore, we add a mechanism to reduce this undesirable effect: if an individual has too many redundant representatives then the surplus is eliminated and new random individuals are generated to replace the old ones.

3.6 Description of the main parameters

The table 1 describes the main parameters of IMPEA and their typical values or range of values, in order of appearance in the paper. Some of these parameters are scalars, like the number of individuals, and are constant along the whole evolution process. Other parameters, like the minimum or maximum number of nodes in Z, are arrays indexed by the number of generations, allowing these parameters to follow a given path during evolution.

Name	Description	Typical value
MaxGens	Number of generations	$50 \dots 200$
Ninds	Number of individuals	$50 \dots 500$
Alpha	Significance level of the χ^2 test	$0.01 \dots 0.25$
Parcim (#Z)	Array of parsimony coefficient (decreases with the length of Z)	$0.5 \dots 1$
PmG	Probability of genome content mutation	$0.1/(2 + \#Z)$
PmAR	Probability of adding or removing nodes in Z	$0.2 \dots 0.5$
PmAdd (#Gen)	Array of probabilities of adding nodes in Z along generations	$0.25 \dots 0.75$
MinNodes (#Gen)	Array of minimal numbers of nodes in Z along generations	$0 \dots 2$
MaxNodes (#Gen)	Array of maximal numbers of nodes in Z along generations	$0 \dots 6$
Pc	Probability of crossover	0.7
PcXY	Probability of swapping x and y	$1/6$
PcZ	Probability of swapping Z	$1/3$
Epsilon	Numerical precision	10^{-5}
MaxRedundant	Maximal number of redundant individuals in the population	$1 \dots 5$

Table 1: Parameters of IMPEA. Values are chosen within their typical range depending on the size of the network and on the desired computation time.

4 Bayesian network structure estimation

The last step of IMPEA consists in reconstructing the structure of the Bayesian network. This is achieved by aggregating all the immortal individuals and only the *good ones* of the final population. An individual $\langle x, y|Z\rangle$ is said to be *good* if its p-value allows not to reject the null hypothesis $x \perp\!\!\!\perp y \mid Z$. There are two strategies in IMPEA: a pure one, called *P-IMPEA*, which consists in strictly enforcing independence statements and an constrained one, called

C-IMPEA, which adds a constraint on the number of desired edges.

Pure conditional independence

Then, as in PC, P-IMPEA starts from a fully connected graph, and for each individual of the aggregated population, it applies the rule "$x \perp\!\!\!\perp y \mid Z \Rightarrow$ no edge between x and y" to remove edges whose nodes belong to an independence statement. Finally, the remaining edges (which have not been eliminated) constitute the undirected structure of the network.

Constrained edges estimation

C-IMPEA needs an additional parameter which is the desired number of edges in the final structure. It proceeds by accumulation: it starts from an empty adjacency matrix and for each $\langle x, y|Z\rangle$ individual of the aggregated population, it adds its fitness to the entry (x, y). An example of a matrix obtained this way is shown on figure 12.

At the end of this process, if an entry (at the intersection of a row and a column) is still equal to zero, then it means that there was no independence statement with this pair of nodes in the aggregated population. Thus, these entries exactly correspond to the strict application of the conditional independences. If an entry has a low sum, then it is an entry for which IMPEA found only a few independence statements (and/or independence statements with low fitness) and thus there is a high expectancy of having an edge between its nodes. Therefore, to add more edges in the final structure (up to the desired number of edges), we just have to select edges with the lowest values and construct the corresponding network.

This approach seems to be more robust since it allows some "errors" in the chi-square tests, but strictly speaking, if an independence statement is discovered, there cannot be any edge between the two nodes.

5 Experiments and results

5.1 Test case 1: comb network

To evaluate the efficiency of IMPEA, we forge a test-network which looks like a *comb*. A n-comb network has $n + 2$ nodes: x, y, and $z_1, z_2, \ldots, z_n$, as one can see on figure 5. The Conditional Probability Tables (CPT) are filled in with a uniform law. It can be seen as a kind of classifier: given the input $z_1, z_2, \ldots, z_n$, it classifies the output as x or y. For example, it could be a classifier that accepts a person's salary details, age, marital status, home address and credit history and classifies the person as acceptable/unacceptable to receive a new credit card or loan.

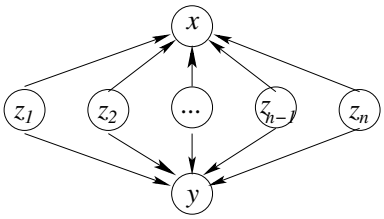

Figure 5: A n-comb network

The interest of such a network is that its independence model can be generated (using

semi-graphoid rules) from the following independence statements:

$$\forall i, j \text{ such as } i \neq j, z_i \perp\!\!\!\perp z_j$$
$$x \perp\!\!\!\perp y \mid \{z_1, z_2, \ldots, z_n\}$$

Thus it has only one complex independence statement and a lot of simple (short) ones. In particular, the only way to remove the edge between x and y using statistical chi-square tests is to test the triplet $\langle x, y \mid \{z_1, z_2, \ldots, z_n\}\rangle$. This cannot be achieved by the PC algorithm as soon as $k < n$ (and in practice, k is limited to 3 due to combinatorial complexity).

Typical run: We choose to test P-IMPEA with a simple 6-comb network. It has been implemented using an open source toolbox, the *Bayes Net Toolbox for Matlab* [19] available at `http://bnt.sourceforge.net/`. We draw our inspiration from PC and initialise the population with individuals with an empty constraining set and let it grow along generations up to 6 nodes, in order to find the independence statement $x \perp\!\!\!\perp y \mid \{z_1, \ldots, z_6\}$. As shown on figure 6, the minimal number of nodes allowed in Z is always 0, and the maximal number is increasing on the first two third of the generations and is kept constant to 6 on the last ones. The average number of nodes in the current population is also slowly rising up but remains rather small since in this example, there are a lot of small *easy to find* independence statements and only a single big one.

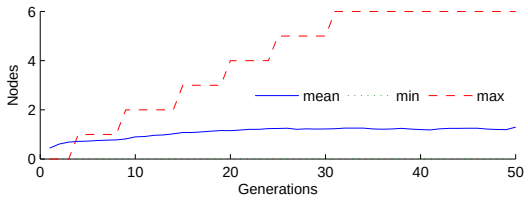

Figure 6: Evolution of Minimal, Maximal and Average number of nodes in Z along generations

The correct structure (figure 7) is found after 40 (out of 50) generations.

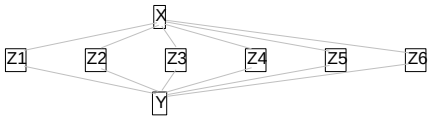

Figure 7: Final evolved structure for the comb network

The figure 8 represents the evolution of the number of errors along generations. The current evolved structure is compared with the actual structure: an *added* edge is an edge present in the evolved structure but not in the actual comb network, and a *deleted* edge is an edge that has been wrongly removed. The total number of errors is the sum of added and deleted edges. Note that even if the number of errors of the discovered edges is extracted at each generation, it is by no means used by IMPEA or reinjected in the population because this information is only relevant in that particular test-case where the Bayesian network that generated the dataset is known.

Statistical results: The previous example gives an idea of the behaviour of P-IMPEA, but to compare it fairly with PC we must compare them not only over multiple runs but also with respect to the size of the dataset. So we set up the following experimental protocol:

- A 4-comb network is created and we use the same Bayesian network (structure and CPT) throughout the whole experiment.

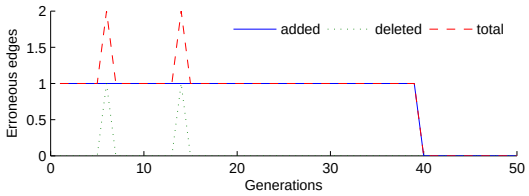

Figure 8: Evolution of the number of erroneous edges of the structure along generations

- We chose representative sizes for the dataset: $\{500, 1000, 2000, 5000, 10000\}$, and for each size, we generate the corresponding number of cases from the comb network.
- We run 100 times both PC and P-IMPEA, and extract relevant information (see tables 2 and 3):
  - How many edges were found? Among these, how many were erroneous? (added or deleted)
  - What is the percentage of runs in which the $x - y$ edge is removed?
- PC is tuned with a fan-in k equal to 3 and P-IMPEA is tuned with 50 generations of 50 individuals in order to take the same computational time as PC. 50 generations are more than enough to converge to a solution due to the small size of the problem. Both algorithms share the same significance level α.

The actual network contains 8 edges and 6 nodes. Therefore, the number of possible alternatives is $2^6 = 64$ and if we roughly want to have 30 samples per possibility, we would need approximatively $64 * 30 \approx 2000$ samples. That explains why performances of the chi-square test are very poor with only 500 and 1000 cases in the dataset. Indeed, when the size of the dataset is too small, PC removes the $x - y$ edge (see the last column of table 2) while it does not even test $\langle x, y \mid \{z_1, z_2, z_3, z_4\}\rangle$ because it is limited by k to 3 nodes in Z. This could appear like an unfair comparison but the value of k is limited to 3 by default in most implementations of the PC algorithm.

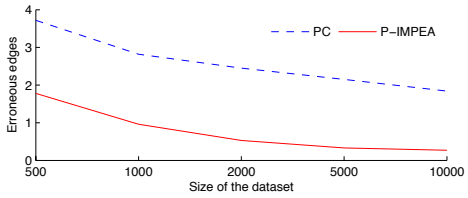

Figure 9: Number of erroneous edges (added+deleted) for PC and P-IMPEA, depending on the size of the dataset

Regarding the global performance, the figure 9 puts up the average number of erroneous nodes (either *added* or *deleted*) of both algorithms. As one can expect, the number of errors decreases with the size of the dataset, and it is clear that P-IMPEA clearly outperforms PC in every case.

Finally, if one has a look to the average number of discovered edges, it is almost equal to 8 (which is the actual number of edges in the 4-comb structure) for P-IMPEA (table 3) whereas it is greater than 9 for the PC algorithm since it can't remove the $x-y$ edge (table 2).

Cases	Edges	Added	Removed	Errors	x-y?
500	5.04 ± 0.85	0.38 ± 0.50	3.34 ± 0.78	3.72 ± 1.01	97%
1000	6.50 ± 1.24	0.66 ± 0.71	2.16 ± 1.01	2.82 ± 1.23	83%
2000	8.09 ± 1.18	1.27 ± 0.80	1.18 ± 0.68	2.45 ± 0.91	39%
5000	9.71 ± 0.74	1.93 ± 0.57	0.22 ± 0.46	2.15 ± 0.73	0%
10000	9.84 ± 0.58	1.84 ± 0.58	0 ± 0	1.84 ± 0.58	0%

Table 2: Averaged results of PC algorithm after 100 runs

Cases	Edges	Added	Removed	Errors	x-y?
500	6.64 ± 0.79	0.05 ± 0.21	1.73 ± 1.90	1.78 ± 1.94	100%
1000	7.32 ± 0.91	0.18 ± 0.50	0.78 ± 1.01	0.96 ± 1.24	100%
2000	8.87 ± 1.04	0.24 ± 0.51	0.29 ± 0.60	0.53 ± 0.82	97%
5000	8.29 ± 0.32	0.30 ± 0.59	0.03 ± 0.17	0.33 ± 0.63	90%
10000	8.27 ± 0.31	0.27 ± 0.54	0 ± 0	0.27 ± 0.54	89%

Table 3: Averaged results of P-IMPEA algorithm after 100 runs

5.2 Test case 2: the Insurance Bayesian Network

Insurance [4] is a network for evaluating car insurance risks. The Insurance Bayesian network contains 27 variables and 52 arcs. We use in our experiments a database containing 50000 cases generated from the network.

Once again, we start from a population with small Z and let it increase up to 4 nodes. The figure 10 illustrates this growth: the average size of the number of nodes in Z of the current population follows the orders given by the minimum and the maximum values.

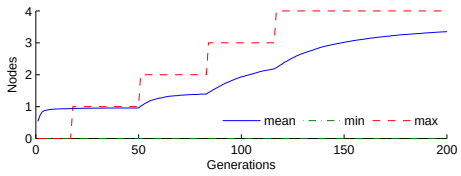

Figure 10: Evolution of Minimal, Maximal and Average number of nodes in Z along generations

Concerning the evolution of the number of erroneous edges, represented on figure 11, it quickly decreases during the first half of the generation (the completely connected graph has more than 700 edges) and then stagnates. At the end, P-IMPEA finds 39 edges out of 52 among which there is no added edge, but 13 which are wrongly removed. It is slightly better than *PC* which also wrongly removes 13 edges, but which adds one superfluous one.

The best results are obtained with C-IMPEA and a desired number of edges equal to 47. Then, only 9 errors are made (see table 4). When asking for 52 edges, the actual number of edges in the Insurance network, it makes 14 errors (7 additions and 7 deletions).

5.3 Analysis

We compared performances on the basis of undirected graphs produced by both algorithms. The edge directions estimation has not been yet programmed in IMPEA, this will be done in future developments, using a low combinatorial strategy similar to PC. Comparisons between both algorithms do not actually depend on this step.

The two experiments of section 5 prove that IMPEA favourably compares to PC, actually, besides the fact that IMPEA relies on a convenient problem encoding, PC performs a deterministic and systematic search while IMPEA uses evolutionary mechanisms to prune computational efforts and to concentrate on promising parts of the search space. The limitation of PC according to problem size is obvious in the first test (Comb network): PC is unable

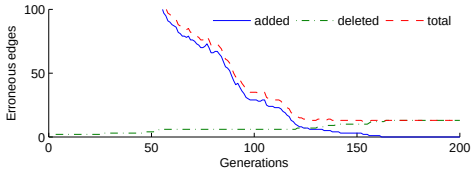

Figure 11: Evolution of the number of erroneous edges of the structure along generations

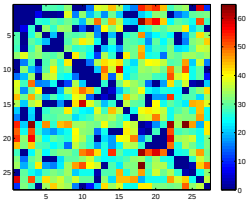

Figure 12: Accumulated adjacency matrix of a network with 27 nodes (from Insurance network).

Algorithm	Edges	Added	Removed	Errors
PC	40	1	13	14
P-IMPEA	39	0	13	13
C-IMPEA	47	2	7	9
C-IMPEA	52	7	7	14

Table 4: Number of detected edges for all algorithms

to capture a complex dependency, even on a small network. Additional tests performed on real data collected during a cheese ripening process [1] proves that IMPEA better resists when using sparse data (insufficient number of available samples).

6 Conclusion

IMPEA has allowed to overcome some known drawbacks of the classical artificial evolution approach to Bayesian Network structure learning (in other words, to find an efficient representation of a direct acyclic graph). We have shown that the scheme we propose is particularly adapted to an alternate representation of Bayesian Networks: Independence Models.

The major difficulty, which is to build a Bayesian Network representative at each generation has been overcome for the moment by a scheme that only built a global solution at the end of the evolution (second step of IMPEA). Future work on this topic will be focused on an improvement of the global fitness management within IMPEA. The major improvement of IMPEA is actually that it only performs difficult combinatorial computations when local mechanisms have pushed the population toward "interesting" area of the search space, thus avoiding to make complex global computations on obviously "bad" solutions. In this sense, CCEAs take into account a priori information to avoid computational waste, in other words, complex computations in unfavourable areas of the search space.

References

[1] O. Barrière, E. Lutton, C. Baudrit, M. Sicard, B. Pinaud and N. Perrot. Modeling human expertise on a cheese ripening industrial process using GP. 10th International Conference on Parallel Problem Solving From Nature, PPSN 2008, Dortmund, Germany, September 13-17, 2008.

[2] O. Barrière, E. Lutton, and P-H Wuillemin. Bayesian Network Structure learning using Cooperative Coevolution. Genetic and Evolutionary Computation Conference (GECCO 2009).

[3] C. Baudrit, P-H. Wuillemin, M. Sicard and N. Perrot. A Dynamic Bayesian Network to represent a ripening process of a soft mould cheese. *submitted*.

[4] J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997.

[5] J. Cheng, D. A. Bell, and W. Liu. Learning belief networks from data: An information theory based approach. In *Sixth ACM International Conference on Information and Knowledge Management*, pages 325–331, 1997.

[6] D. M. Chickering and C. Boutilier. Learning equivalence classes of Bayesian-Network structures. In *Journal of Machine Learning Research*, pages 150–157. Morgan Kaufmann, 1996.

[7] D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian Networks is np-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

[8] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[9] G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 09(4):309–347, 1992.

[10] K. Deb and D.E. Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the third Conference on Genetic Algorithms*, pages 42–50, 1989.

[11] O. Francois and P. Leray. Etude comparative d'algorithmes d'apprentissage de structure dans les réseaux bayésiens. In *Rencontres des Jeunes Chercheurs en IA*, 2003.

[12] N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In *14th International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann, 1997.

[13] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian Network to analyze expression data. *J. Computational Biology*, 7:601–620, 2000.

[14] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Second International Conference on Genetic Algorithms and their application*, pages 41–49. Lawrence Erlbaum Associates, Inc., 1987.

[15] H. Jia, D. Liu, and P. Yu. Learning dynamic Bayesian Network with immune evolutionary algorithm. 2005.

[16] P. Larra naga and M. Poza. Structure learning of Bayesian Networks by genetic algorithms: A performance analysis of control parameters. *IEEE Journal on Pattern Analysis and Machine Intelligence*, 18(9):912–926, 1996.

[17] J. Louchet, M. Guyon, M. J. Lesot, and A. M. Boumaza. Dynamic flies: a new pattern recognition tool applied to stereo sequence processing. *Pattern Recognition Letters*, 23(1-3):335–345, 2002.

[18] E. Lutton and P. Martinez. A genetic algorithm with sharing for the detection of 2d geometric primitives in images. In *AE '95: Selected Papers from the European conference on Artificial Evolution*, pages 287–303. Springer-Verlag, 1995.

[19] K. Murphy. The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*, 33(2):1024–1034, 2001.

[20] J. W. Myers, K. B. Laskey, and K. A. DeJong. Learning Bayesian Networks from incomplete data using evolutionary algorithms. In *Genetic and Evolutionary Computation Conference*, volume 1, pages 458–465. Morgan Kaufmann, 1999.

[21] J. Pearl and T. Verma. A theory of inferred causation. In *Second International Conference on the Principles of Knowledge Representation and Reasoning*, 1991.

[22] R. W. Robinson. Counting unlabeled acyclic digraphs. *Combinatorial mathematics V*, 622:28–43, 1977.

[23] B. J. Ross and E. Zuviria. Evolving dynamic Bayesian Networks with multi-objective genetic algorithms. *Applied Intelligence*, 26, 2007.

[24] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. The MIT Press, second edition, 2001.

[25] A. Tucker and X. Liu. Extending evolutionary programming methods to the learning of dynamic Bayesian Networks. In *GECCO '99*, 1999.

[26] S. C. Wang and S. P. Li. Learning Bayesian Networks by lamarckian genetic algorithm and its application to yeast cell-cycle gene network reconstruction from time-series microarray data. 3141/2004:49–62.

[27] M. L. Wong and K. S. Leung. An efficient data mining method for learning Bayesian networks using an evolutionary algorithm-based hybrid approach. 8:378–404, 2004.