INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Modeling an agrifood industrial process using cooperative coevolution Algorithms

Olivier Barrière — Evelyne Lutton — Pierre-Henri Wuillemin — Cédric Baudrit — Mariette Sicard — Bruno Pinaud — Nathalie Perrot

N° 6914

April 2009

Thème BIO

ISSN 0249-6399 ISRN INRIA/RR--6914--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY - ÎLE-DE-FRANCE

Modeling an agrifood industrial process using cooperative coevolution Algorithms

Olivier Barrière , Evelyne Lutton , Pierre-Henri Wuillemin * ,
Cédric Baudrit † , Mariette Sicard † , Bruno Pinaud † , Nathalie Perrot †

Thème BIO — Systèmes biologiques
Équipe-Projet Apis

Rapport de recherche n° 6914 — April 2009 — 48 pages

Abstract: This report presents two experiments related to the modeling of an industrial agrifood process using evolutionary techniques. Experiments have been focussed on a specific problem which is the modeling of a Camembert-cheese ripening process. Two related complex optimisation problems have been considered:

- a deterministic modeling problem, the phase prediction problem, for which a search for a closed form tree expression has been performed using genetic programming (GP),
- a Bayesian network structure estimation problem, considered as a two-stage problem, i.e. searching first for an approximation of an independence model using EA, and then deducing, via a deterministic algorithm, a Bayesian network which represents the equivalence class of the independence model found at the first stage.

In both of these problems, cooperative-coevolution techniques (also called "Parisian" approaches) have been proved successful. These approaches actually allow to represent the searched solution as an aggregation of several individuals (or even as a whole population), as each individual only bears a part of the searched solution. This scheme allows to use the artificial Darwinism principles in a more economic way, and the gain in terms of robustness and efficiency is important.

Key-words: Agrifood, Cheese ripening, Cooperative coevolution, Parisian approach, Genetic Programming, Bayesian Network

* LIP6-CNRS UMR7606, 75016 Paris
† UMR782 Génie et Microbiologie des Procédés Alimentaires. AgroParisTech, INRA, F-78850 Thiverval-Grignon, France

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université
4, rue Jacques Monod, 91893 ORSAY Cedex
Téléphone : +33 1 72 92 59 00

Modélisation d'un processus industriel agroalimentaire par coopération-coévolution

Résumé : Nous présentons ici deux expériences concernant l'usage de techniques évolutionnaires dans le cadre de la modélisation de processus industriels agroalimentaires. L'étude concerne la modélisation d'un processus de maturation de fromage (Camembert). Deux problèmes complexes d'optimisation ont été considérés :

- une modélisation déterministe, le problème de la détermination de la phase, pour lequel on cherche une formule explicite, exprimée sous forme d'arbre, via un algorithme de programmation génétique (GP),
- une modélisation sous forme de réseau Bayésien, que l'on aborde en deux étapes : d'abord une recherche d'une approximation d'un modèle d'indépendance de données par algorithme évolutionnaire (EA), puis une reconstruction (via un algorithme déterministe) d'un réseau Bayésien, représentant de la classe d'équivalence figurée par le modèle d'indépendance trouvé dans la première étape.

Pour chacun de ces deux problèmes, nous prouvons que l'usage d'une technique de coopération-coévolution (ou "Evolution Parisienne") est bénéfique. En effet, cette approche permet de représenter la solution recherchée comme une aggrégation de plusieurs individus (voire même de toute une population), chaque individu ne portant qu'un part de la solution recherchée. Ce schéma permet d'exploiter de façon plus économique les principes du Darwinime artificiel, et les gains en termes de robustesse et d'efficacité sont importants.

Mots-clés : Agroalimentaire, Affinage, Coopération, Coévolution, Approche Parisienne, Programmation Génétique, Réseau Bayésien

Contents

1 Backgrounds

This study is part of the French INCALIN research project[1], whose goal is the modelling of agrifood industrial processes. In such food industries, manufacturing processes consist in successive operations whose underlying mechanisms are sometimes still ill-known, such as the cheese ripening process. The challenge of INCALIN is the understanding of the causal relationships between on the one hand, ingredients and physico-chemical or microbiological characteristics and on the other hand, sensory and nutritional properties. The intriguing question is how micro level properties determine or at least influence those on the macro level? The project aims at reconstructing this puzzle of knowledge as to explain the global behaviour of the system.

1.1 Modeling agrifood industrial processes

Various macroscopic models have been experimented to embed expert knowledge, like expert systems [30, 31, 29], neural networks [33, 43], mechanistic models [1, 49], or dynamic Bayesian networks [5].

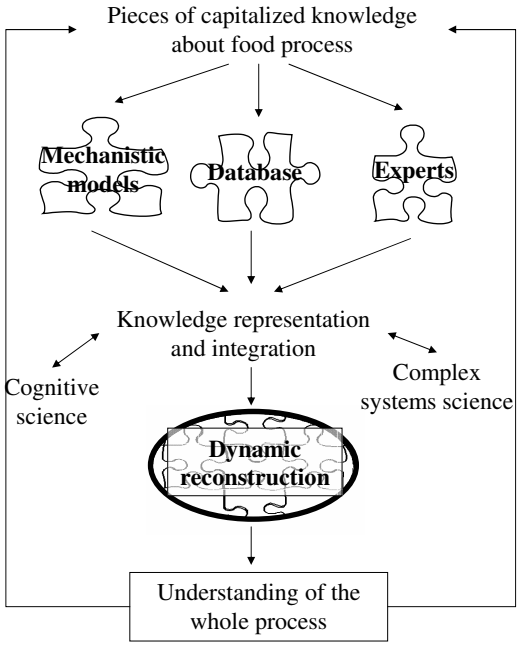


Figure 1: Knowledge integration to represent the dynamics of food processes.

The major problem common to these techniques is related to the sparseness of available data: collecting experimental data is a long and difficult process, and resulting data sets are often uncertain or even erroneous. For example, a complete cheese ripening process last 40 days, and some tests are destructive, i.e a sample cheese is consumed in the analysis. Other measurements require

[1]supported by the ANR-PNRA fund

to grow bacterias in Petri dishes and then to count the number of colonies, which takes a lot of time. Therefore the precision of the resulting model is often limited by the small number of valid experimental data, and parameter estimation procedures have to deal with incomplete, sparse and uncertain data.

1.2 The Camembert-cheese ripening process

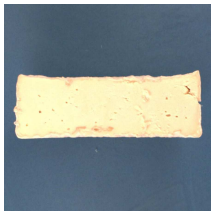
Experimental procedures in laboratories ("model cheeses") use pasteurized milk inoculated with *Kluyveromyces marxianus* (*Km*), *Geotrichum candidum* (*Gc*), *Penicillium camemberti* (*Pc*) and *Brevibacterium auriantiacum* (*Ba*) under aseptic conditions.

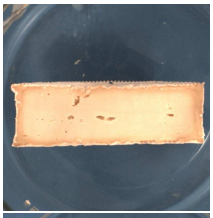
- *K. marxianus* is one of the key flora of Camembert cheese. One of its principal activity is the fermentation of lactose (noted *lo*) [13, 14] (curd de-acidification by lactose consumption). Three dynamics are apparent in the timeline of *K. marxianus* growth [36, 35]. Firstly, there is an exponential growth during about five days that corresponds to a decrease of lactose concentration. Secondly, the concentration of *K. marxianus* remains constant during about fifteen days and thirdly decreases slowly.

- *G. candidum* plays a key role in ripening because it contributes to the development of flavour, taste and aroma of cheeses [2, 8, 37]. One of its principal activities is the consumption of lactate (noted *la*). Three dynamics are apparent in the timeline of *G. candidum* growth [36, 35]. Firstly, there is a latency period during about three days. Secondly, there is an exponential growth that corresponds to a decrease of lactate concentration and thus an increase of pH. Thirdly, the concentration of *G. candidum* remains constant to the end of ripening.

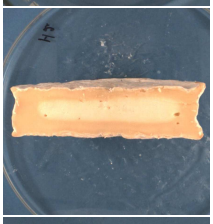
During ripening, these soft-mould cheese behave like an ecosystem (a bioreactor) extremely complex to be modeled as a whole, and where human experts operators have a decisive role. Relationships between microbiological and physicochemical changes depend on environmental conditions (*e.g.* temperature, relative humidity ...) [35] and influence the quality of ripened cheeses [28, 36]. A ripening expert is able to estimate the current state of some of the complex reactions at a macroscopic level through its perceptions (sight, touch, smell and taste). Control decisions are then generally based on these subjective but robust expert measurements. An important information for parameter regulation is the subjective estimation of the current state of the ripening process, discretised in four phases:

- **Phase 1** is characterized by the surface humidity evolution of cheese (drying process). At the beginning, the surface of cheese is very wet and evolves until it presents a rather dry aspect. The cheese is white with an odor of fresh cheese.

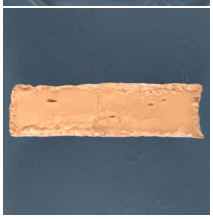




• **Phase 2** begins with the apparition of a *P. camemberti*-coat (*i.e* the white-coat at the surface of cheese), it is characterized by a first change of color and a "mushroom" odor development.



• **Phase 3** is characterized by the thickening of the creamy under-rind. *P. camemberti* cover all the surface of cheeses and the color is light brown.



• **Phase 4** is defined by strong ammoniac odor perception and the dark brown aspect of the rind of cheese.

These four steps are representative of the main evolution of the cheese during ripening. The expert's knowledge is obviously not limited to these four phases, but these phases help to evaluate the whole dynamics of ripening and to detect drift from the standard evolution.

1.3 Modeling human expertise on cheese ripening

A major problem addressed in the INCALIN project is the search for automatic procedures that mimic the way human expert aggregate data through their senses to estimate and regulate the ripening of the cheese. Stochastic optimisation techniques, like evolutionary techniques, have already been proven successful on several agrifood problems [4, 22, 54]. In this report, we explore how genetic programming (GP) and cooperative-coevolution algorithms (CCEA) can be used to capture (learn) expert knowledge. The first part of this report deals with the estimation of the phase using Genetic Programming and the second part goes through the estimation of the structure of a Bayesian network using independence models.

1.3.1 Phase estimation using a deterministic model

A first approach to the problem of phase detection is to search for a convenient formula that estimates the phase at time t, knowing micro-organisms proportions at the same time t, but without a priori knowledge of the phase at the time $t-1$. Genetic programming (GP) is a convenient tool to manage this as an optimisation problem. GP is a specialization of evolutionary algorithms where each individual of a population is a function, represented as a tree structure. Every tree node has an operator function $(+, -, /, *, \ldots)$ and every terminal node has an operand (a constant or a variable). It is then easy to apply mutations and crossover to these trees within a genetic algorithm.

In a first classical GP approach, the phase estimator is search as a single best "monolithic" function. Although it already outperforms other methods, we decided to split the phase estimation into four combined (and simpler) "phase detectors". The structures searched are binary output functions (or binarised functions) that characterize one of the four phases. The population is shared into four classes such that individuals of class k are good at characterizing phase k. Finally, a global solution is made of at least one individual of each class, in order to be able to classify the sample into one of the four phases via a voting scheme. The problem is formulated as a collective task, following cooperative coevolution principles known as the "Parisian approach" (see section 1.4).

This approach is more robust than the classical GP approach because it has almost the same recognition rate but with a lower variance. Moreover, it allows to evolve simpler structure during less generations, and yield results that are easier to interpret. Nevertheless, it often happens that a generation notably improves the global fitness, while the generations that follow are not able to keep it: the global fitness is not a monotonically increasing function. In order to avoid this stagnation phenomenon due to over-specialisation of the best individuals, we experiment a variable sized population Parisian GP strategy, using adaptive deflating and inflating schemes for the population size. The idea is to group individuals with the same characteristics into "clusters" and remove the most useless ones at the end of every generation while periodically adding "fresh blood" to the population (i.e. new random individuals) if a stagnation criterion is fulfilled.

1.3.2 Bayesian network structure estimation

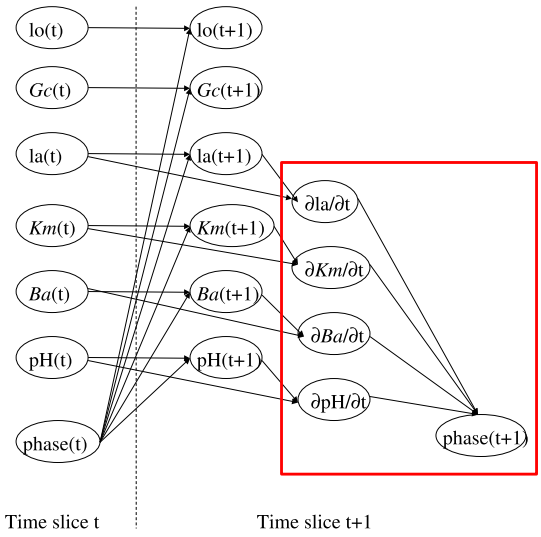


Figure 2: Dynamic Bayesian Network representing the dynamic of variables depending on the observation of ripening phases. The static Bayesian network used for comparison in the sequel is in the right box

Stochastic models can also be used to capture expert knowledge. Bayesian networks are for example a convenient model for this purpose, already used for cheese ripening modeling [5, 47]. In this work actually, a dynamic Bayesian network (figure 2) has been manually built, using human expert knowledge, to represent the macroscopic dynamic of each variable. The phase the network is in at time t plays a determinant role for the prediction of the variables at time $t+1$. Moreover, four relevant variables have been identified, the derivative of pH, la, Km and Ba at time t, allowing to predict phase at time $t+1$. This leads to a computer-based phase estimation to model the way experts aggregate information from their senses.

In order to go further in this direction, a theoretical study has been carried out. The idea is to learn the structure of a Bayesian network by considering an equivalent representation: the independence model (IM). IM represents data dependencies via a set of independence statements (IS). An IS=$(X, Y|S)$ means that the variable X is independent of Y knowing the set of variables S and is evaluated by χ^2 tests. If we consider a set of ISs that evolve according to a Parisian evolutionary scheme, the best of them at each generation can be selected to build a partial IM. We thus get a convenient way to evolve partial IM. The remaining problem, which is to built the structure of the Bayesian network that best fits a given partial IM can be adressed using a deterministic procedure.

Once again, a cooperative evolutionary approach is proven to be beneficial, see section 3.

1.4 Cooperative co-evolution techniques

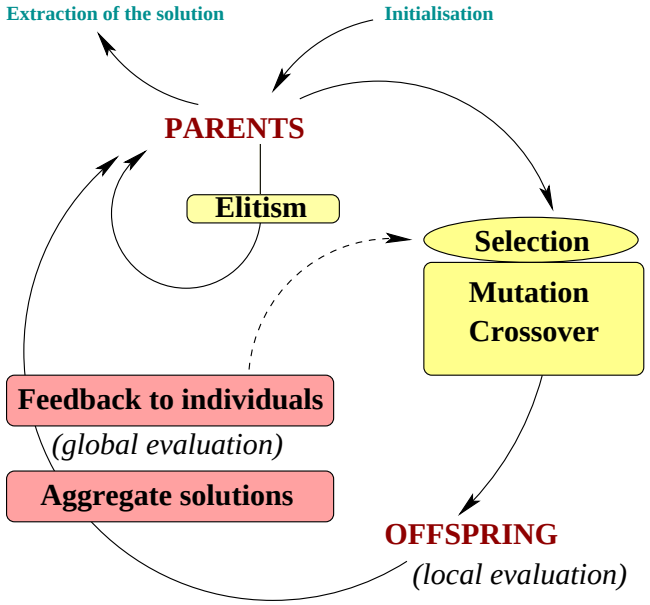


Figure 3: A Parisian EA: a monopopulation cooperative-coevolution

Cooperative coevolution strategies actually rely on a formulation of the problem to be solved as a cooperative task, where individuals collaborate or compete

in order to build a solution. They mimic the ability of natural populations to build solutions via a collective process. Nowadays, these techniques are used with success on various problems [20, 57], including learning problems [7].

The large majority of these approaches deals with a coevolution process that happens between a fixed number of separated populations [45, 9, 48]. We study here a different implementation of cooperative coevolution principles, the so-called Parisian approach [16, 44] described on figure 3, that uses cooperation mechanisms within a *single* population. It is based on a two-level representation of an optimization problem, in the sense that an individual of a Parisian population represents only a part of the solution to the problem. An aggregation of multiple individuals must be built in order to obtain a solution to the problem. In this way, the co-evolution of the whole population (or a major part of it) is favoured instead of the emergence of a single best individual, as in classical evolutionary schemes. The motivation is to make a more efficient use of the genetic search process, and reduce the computational expense. Successful applications of such a scheme usually rely on a lower cost evaluation of the partial solutions (i.e. the individuals of the population), while computing the full evaluation only once at each generation.

2 Phase estimation using GP

The interest of evolutionary optimisation methods for the resolution of complex problems related to agrifood has been proved by various recent publications. For example [4] uses genetic algorithms to identify the smallest discriminant set of variables to be used in certification process for an Italian cheese (validation of origin labels). [22] used GP to select the most significant wavenumbers produced by a Fourier transform infrared spectroscopy measurement device, in order to build a rapid detector of bacterial spoilage on beef. And a recent overview on optimisation tools in food industries [54] mentions works based on evolutionary approaches.

2.1 Phase estimation using a classical GP

A Genetic Programming (GP) approach is used to search for a convenient formula that links the four derivatives of micro-organisms proportions to the phase at each time step t (static model), without *a priori* knowledge of the phase at $t-1$. GP is a specialisation of genetic algorithms where each individual is a function, represented as a tree structure (figure 4). Every tree node is an operator function $(+, -, /, *, \ldots)$ and every terminal node is an operand (a constant or a variable).

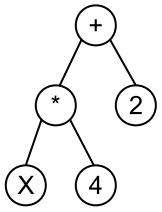


Figure 4: $2X+4$ represented as a tree structure

This problem is a symbolic regression one, however, it has to be noted that the small number of samples and their irregular distribution make it difficult. Results will be compared with the performances of a static Bayesian network, extracted from the DBN of [5], (the part within the box on figure 2), and with a very simple learning algorithms (multilinear prediction, see section 2.2.5).

2.1.1 Search space

The derivatives of four variables will be considered, namely the derivative of pH (acidity), la (lactose proportion), Km and Ba (lactic acid bacteria proportions, see section 1.2), for the estimation of the phase (static problem). The GP will search for a phase estimator $\widehat{Phase}(t)$, i.e. a function defined as follows:

$$\widehat{Phase}(t) = f\left(\frac{\partial pH}{\partial t}, \frac{\partial la}{\partial t}, \frac{\partial Km}{\partial t}, \frac{\partial Ba}{\partial t}\right)$$

The function set is made of arithmetic operators: $\{+, -, *, /, \hat{}, log\}$, with protected / and log, and logical operators $\{if, >, <, =, and, or, xor, not\}$ in order to allow complex estimation formula.

The terminal set is made of the four partial derivatives plus real constants. The constant's values are not limited, but randomly initialised using one of the

following laws $\mathcal{U}$ $[0,1]$, $-\mathcal{U}$ $[0,1]$, $\mathcal{N}$ $(0,1)$, also randomly chosen. ($\mathcal{U}$ is the uniform law, and $\mathcal{N}$ the normal law).

2.1.2 Fitness function

Available data are shared in two sets: learning set and test set, that are randomly chosen within the available data set for each run. The 16 available experiences are thus randomly shared between learning and test sets. The size of the learning set vary from 10 to 15 experiments, while the size of the corresponding test set vary from 6 to 1 experiment (see section 2.2.5).

The fitness function, *to be minimised*, is made of a factor that measures the quality of the fitting on the learning set, plus a "parsimony" penalisation factor in order to minimize the size (the number of nodes, actually) of the evolved structures (to avoid bloat). It is divided by the number of variables involved in the evaluated tree in order to favour structures that embed all four variables of the problem (this is a requirement of biologists ; experiments also show that recognition results are better with this constraint):

$$fitness = \frac{\sum_{learning\_set} \left| f\left(\frac{\partial pH}{\partial t}, \frac{\partial la}{\partial t}, \frac{\partial Km}{\partial t}, \frac{\partial Ba}{\partial t}\right) - Phase(t) \right| + W\#Nodes}{\#Variables + 1}$$

The parameter W has been experimentally tuned, the optimal value ($W = 1$) favours evolution of structures with 30 to 40 nodes.

2.1.3 Genetic operators

A classical tree crossover (exchange of subtrees from a randomly chosen node) has been used with probability p_c (defined per tree), as a means of evolving the structure of the tree. Two types of mutations have been used:

- **a subtree mutation** (mutation of the structure), that randomly rebuilt a new subtree from a randomly chosen node, applied with probability p_{sm} (defined per tree),
- **a point mutation** (mutation of nodes content), applied with probability p_{cm} (also defined per tree) that does not modify the structure, but randomly changes the content of each node of the tree within the set of compatible functions or terminals (arity constraints). The probabilities (defined per node) are detailed in table 1. Real values are considered separately and undergo a real mutation with probability p_{rm} as a multiplicative perturbation according to a χ^2 law of parameter N:

  $$x' = x \frac{\sum_{i=1}^{N} \mathcal{N}(0,1)^2}{N}$$

  p_{rm} and N vary linearly according to generations, from 0.1 to 0.5 for p_{rm}, and from 1 to 1000 for N, in order to start with rather infrequent large radius mutations and finish with more frequent mutations with smaller radius.

Table 1: Probabilities of point mutation operators

From	to	probability
operator	operator	0.1
variable	variable	0.1
variable	constant	0.05
constant	variable	0.05
constant	constant	p_{rm}: 0.1 to 0.5
		N: 1 to 1000

Crossover, subtree and point mutation probabilities vary along evolution according to the adapting scheme[18] available in the GPLAB toolbox[27]. p_c, p_{sm} and pcm are initially fixed to $\frac{1}{3}$, and are updated according statistics of success of the various operators computed on a tuneable window of past generations.

2.2 Phase estimation using a Parisian GP

Instead of searching for a phase estimator as a single monolithic function, phase estimation can actually be split into four combined (and simpler) phase detection trees as shown on figure 5. The structures searched are binary output functions (or binarised functions) that characterize one of the four phases. The population is then split into four classes such that individuals of class k are good at characterizing phase k. Finally, a global solution is made of at least one individual of each class, in order to be able to classify the sample into one of the four previous phases via a voting scheme detailed at the end of this section.

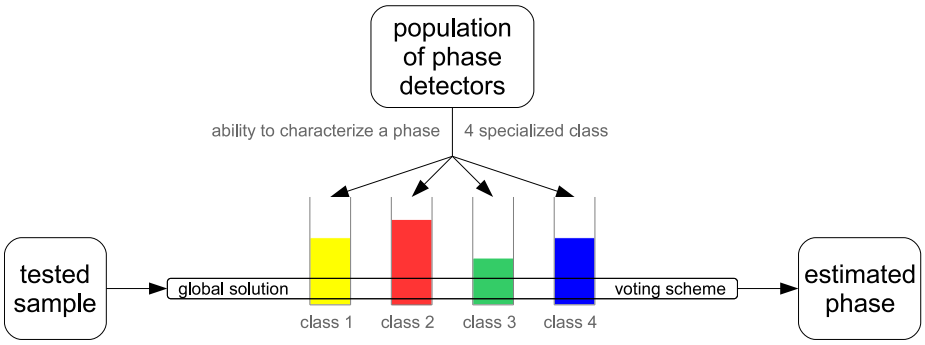


Figure 5: Phase estimation using a Parisian GP. Four classes of phase detectors are defined: individuals of class k are good at characterizing phase k.

2.2.1 Search space

We now search for formulas of type: $I\left(\frac{\partial pH}{\partial t}, \frac{\partial la}{\partial t}, \frac{\partial Km}{\partial t}, \frac{\partial Ba}{\partial t}\right)$ with real outputs mapped to binary outputs, via a sign filtering: $(I() > 0) \rightarrow 1$ and $(I() \leq 0) \rightarrow 0$. The functions (except logical ones) and terminal sets, as well as the genetic operators, are the same as in the global approach above.

Using the available samples of the learning set, four values can be computed, in order to measure the capability of an individual I to characterize each phase:

$$k \in \{1,2,3,4\} \quad F_k(I) = 3 \sum_{i,phase=k} \frac{I(sample(i))}{\#Samples_{phase=k}} - \sum_{i,phase\neq k} \frac{I(sample(i))}{\#Samples_{phase\neq k}}$$

i.e. if I is good for representing phase k, then $F_k(I) > 0$ and $F_{\neq k}(I) < 0$

2.2.2 Local fitness

The local adjusted fitness value, *to be maximised*, is a combination of three factors:

$$AdjFit = \max\{F_1, F_2, F_3, F_4\} \times \frac{\#Ind}{\#IndPhaseMax} \times \frac{NbMaxNodes}{NbNodes}\Big|_{\text{if } NbNodes > NbMaxNodes}$$

The first factor is aimed at characterising if individual I is able to distinguish one of the four phases, the second factor tends to balance the individuals between the four phases ($\#IndPhaseMax$ is the number of individuals representing the phase corresponding to the $argmax$ of the first factor and $\#Ind$ is the total number of different individuals in the population) and the third factor is a parsimony factor in order to avoid large structures. $NbMaxNodes$ has been experimentally tuned, and is currently fixed to 15.

Several fitness measures are actually used to rate individuals, namely the raw fitness $rawfitness$, i.e. the set of four values $\{F_1, F_2, F_3, F_4\}$, that measure the ability of the individual to characterize each phase, the local fitness $localfitness = \max(rawfitness)$ which represents the best characterised phase, and the adjusted fitness $adjfitness = \frac{localfitness}{\mu} \times \frac{\#IndPhaseMax}{\#Ind} \times \frac{\#NodesMax}{\#Nodes} \times bonus^{\alpha}$, which includes sharing, balance, parsimony and global fitness bonus terms.

2.2.3 Sharing distance

The set of measurements $\{F_1, F_2, F_3, F_4\}$ provides a simplified representation in $\mathbb{R}^4$ of the discriminant capabilities of each individual. As the aim of a Parisian evolution is to evolve distinct subpopulations, each being adapted to one of the four subtasks (i.e. characterize one of the four phases), it is natural to use an euclidean distance in this four dimensional phenotype space, as a basis of a simple fitness sharing scheme [19].

2.2.4 Aggregation of partial solutions and global fitness

At each generation, the population is shared in four classes corresponding to the phase each individual characterises the best (i.e. the argmax of $\max\{F_1, F_2, F_3, F_4\}$ for each individual). The 5% best of each class are used via a voting scheme to decide the phase of each tested sample[2] (see figure 5). The global fitness measures the proportion of correctly classified samples on the learning set:

[2]This scheme may also yield a confidence level of the estimation. This measurement is not yet exploited but can be used in future developments of the method.

$$GlobalFit = \frac{\sum_{i=1}^{learning\_set} CorrectEstimations}{\#Samples}$$

The global fitness is then distributed as a multiplicative bonus on the individuals who participated in the vote: $LocalFit' = LocalFit \times (GlobalFit + 0.5)^{\alpha}$.

As $GlobalFit \in [0, 1]$, multiplying by $(GlobalFit + 0.5) > 1$ corresponds to a bonus. The parameter α varies along generations, for the first generations (a third of the total number of generations) $\alpha = 0$ (no bonus), and then α linearly increases from 0.1 to 1, in order to help the population to focus on the four peaks of the search space.

Two sets of indicators are computed at each generation (see section 2.2.5, third line of figure 7):

- the sizes of each class, that show if each phase is equally characterised by the individuals of the population.
- the discrimination capability of each phase, computed on the 5% best individuals of each class as the minimum of:

$$\Delta = \max_{i \in [1,2,3,4]} \{F_i\} - \frac{\sum_{k \neq argmax\{F_i\}} \{F_k\}}{3}$$

2.2.5 Experimental analysis

Available data have been collected from 16 experiments during 40 days each, yielding 575 valid measurements.[3] The derivatives of pH, la, Km and Ba have been averaged and interpolated (spline interpolation) for some missing days. Logarithms of these quantities are considered.

Table 2: Parameters of the GP methods

	GP	**Parisian GP**
Population size	1000	1000
Number of generations	100	50
Function set	arithmetic and logical functions	arithmetic functions only
Sharing	no sharing	$\sigma_{share} = 1$ at the beginning, then linear decrease from 1 to 0.1 $\alpha_{share} = 1$ (constant)

The parameters of both GP methods are detailed in table 2. The code has been developed in Matlab, using the GPLAB toolbox[27]. Comparative results of the four considered methods (multilinear regression, Bayesian network, GP and Parisian GP) are displayed in figure 6, and a typical GP run is analysed in figure 7.

The multilinear regression algorithm used for comparison works as follows: the data are modeled as a linear combination of the four variables:

$$\widehat{Phase}(t) = \beta_1 + \beta_2 \frac{\partial pH}{\partial t} + \beta_3 \frac{\partial la}{\partial t} + \beta_4 \frac{\partial Km}{\partial t} + \beta_5 \frac{\partial Ba}{\partial t}$$

[3] The data samples are relatively balanced except for phase 3, which has a longer duration, thus a larger number of samples: we got 57 representatives of phase 1, 78 of phase 2, 247 of phase 3 and 93 of phase 4.

The 5 coefficients $\{\beta_1, \ldots, \beta_5\}$ are estimated using a simple least square scheme.

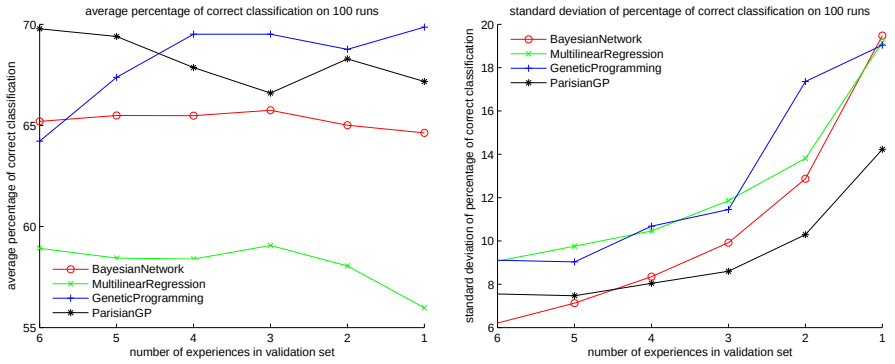


Figure 6: Average (left) and standard-deviation (right) of recognition percentage on 100 runs for the 4 tested methods, the abscissa represent the size of the test-set

Experiments show that both GP outperform multilinear regression and Bayesian network approaches in terms of recognition rates. Additionally the analysis of a typical GP run (figure 7) shows that much simpler structures are evolved: The average size of evolved structures is around 30 nodes for the classical GP approach and between 10 and 15 for the Parisian GP.

It has also to be noted in figure 7 that co-evolution is balanced between the four phases, even if the third phase is the most difficult to characterize (this is in accordance with human experts' judgement, for which this phase is also the most ambiguous to characterize).

The development of a cooperative-coevolution GP scheme (Parisian evolution) seems very attractive, as it allows to evolve simpler structure during less generations, and yield results that are easier to interpret. Moreover, the computation time is almost equivalent between both presented methods (100 generations of a classical GP against 50 generations of a Parisian one), as one "Parisian" generation necessitates more complex operations, all in all). One can expect a more favourable behaviour of the Parisian scheme on more complex issues than the phase prediction problem, as the benefit of splitting the global solutions into smaller components may be higher and may yield computational shortcuts (see for example [16]).

2.3 Variable population size strategies in a Parisian GP

2.3.1 Stagnation problem

First of all, it is necessary to distinguish local and global levels:

- the adjusted fitness is used as a basis for selection, crossover and mutation operators, associated to a first elitism mechanism which keeps in the population the four best individuals *of the current generation* (one per phase) based on the non adjusted fitness.

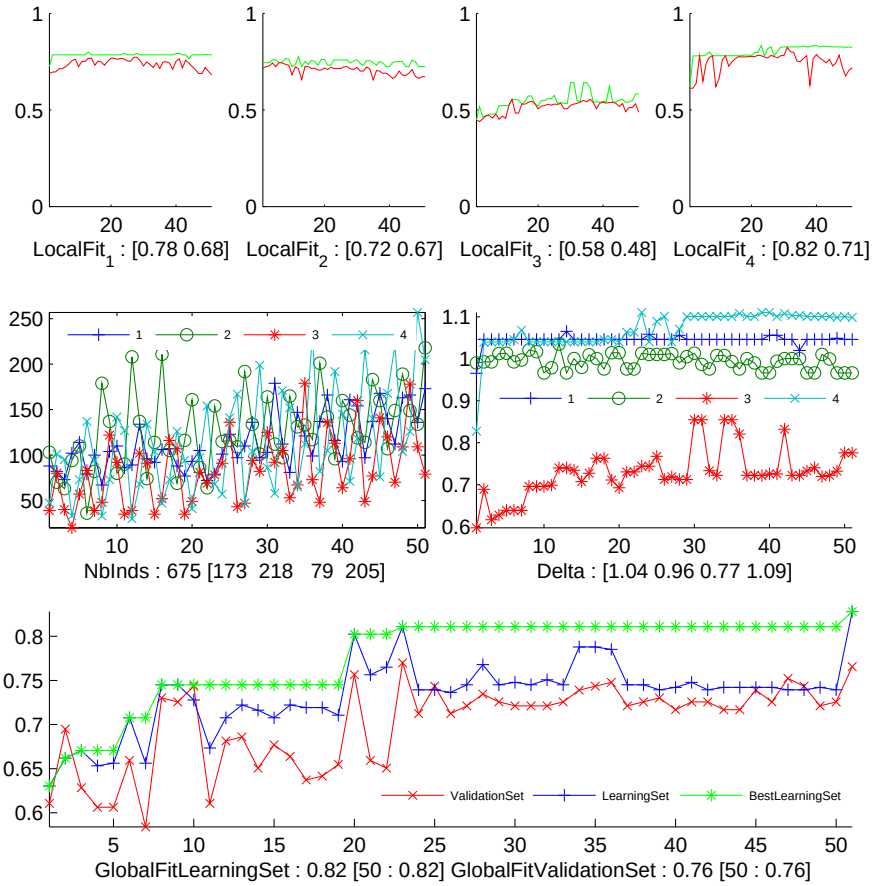


Figure 7: A typical run of the Parisian GP:
- **First line:** the evolution with respect to generation number of the 5% best individuals for each phase: the upper curve of each of the four graphs is for the best individual, the lower curve is for the "worst of 5% best" individuals.
- **Second line left:** the distribution of individuals for each phase: the curves are very irregular but numbers of representatives of each phases are balanced.
- **Second line right:** discrimination indicator, which shows that the third phase is the most difficult to characterize.
- **Third line:** evolution of the recognition rates of learning and test set. The best-so-far recognition rate on learning set is tagged with a star.

- at the end of each generation, the global fitness is computed and reinjected in the population as a bonus, combined with a second elitism mechanism, which keeps the four individuals *of the generation that yielded the best global fitness.*

Despite of local elitism and bonus mechanisms, the global fitness is not a monotonically increasing function. In particular, it often happens that a generation notably improves the global fitness, while the generations that follow are not

able to keep it as one can see on figure 8.

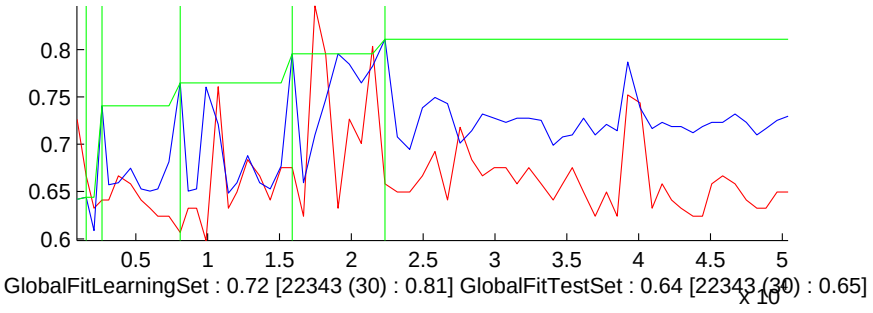


Figure 8: Typical run of a Parisian GP: stagnation of the global fitness

To avoid this undesirable effect, a variable sized population Parisian GP strategy is experimented, using adaptive deflating and inflating schemes for the population size. The idea is to group individuals with the same characteristics into "clusters" and remove the most useless ones at the end of every generation while periodically adding "fresh blood" to the population (i.e. new random individuals) if a stagnation criterion is fulfilled.

Various population sizing and resizing schemes have been studied in the literature for classical evolutionary schemes [38, 21]. It has been clearly stated that adaptive population size allows to build more efficient optimisation algorithms, by dynamically balancing the exploration and exploitation capabilities of the search, the gain in efficiency being measured in terms of number of fitness evaluations.

Common on-line population size adjustment schemes are related to the improvement of the best individual of the population, to the variance of population fitness, or rely on the notion of age and lifetime of individuals. There also exists strategies based on competing subpopulations, for example [52] proposed a scheme based on competing subpopulations: each subpopulation is running a different search strategy, and regularly compete with each other. The size of "good" strategies then increases while "bad" ones decreases, the sum of the sizes of all population being constant.

However, to the best of our knowledge, there exists no work of this type for cooperative-coevolution schemes. The strategy we experiment for mono-population cooperative-coevolution relies on the notion of global fitness improvement, and allows to allocate less local fitness evaluations to obtain a better result *in fine*. Tests have been performed in order to evaluate the improvements due to population deflation, then to population deflation + inflation, in comparison to a constant population size scheme.

2.3.2 Fair play comparison

In order to fairly compare different schemes, results will be indexed with the number of new individuals evaluations instead of the number of generations. As a consequence, for the same cost (i.e the same total number of evaluations) a decreasing size population scheme "uses" more generations.

2.3.3 Redundancy - Diversity's hidden iceberg

Because of the binarised output which only takes into account the sign of the identification function $I()$, several individuals may have the same raw fitness. This is often the case at the end of the evolution, which causes a loss of diversity.

2.3.4 Clustering

Individuals having the same $rawfitness$ are grouped into clusters. Then, inside each cluster, individuals are sorted according to their number of nodes as described in figure 9. The first and best one is the one with the smallest number of nodes.

Input: population of size N
Output: population of size lower or equal to N
foreach *cluster of the population* **do**
  if *size of the cluster greater than to_keep* **then**
    remove the last *to_remove* individuals from the cluster
  else
    keep all individuals from the cluster
  end
end

Algorithm 1: Elimination

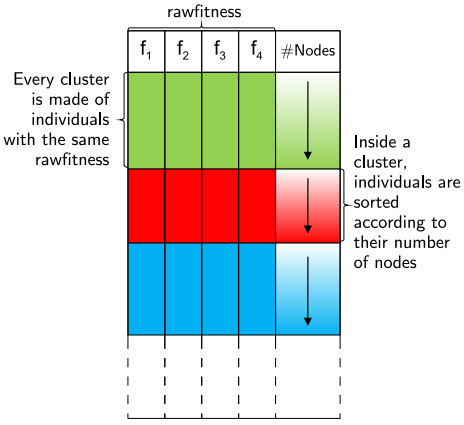


Figure 9: Population clustering

2.3.5 Elimination rules

Useless individuals elimination allows to decrease the population size: an individual is considered as useless if it belongs to a big cluster and has a large number of nodes. The elimination rule depends on two parameters (*to_keep* and *to_remove*), in order to tune the decreasing speed of the population while keeping enough diversity. The elimination procedure is called at the end of each generation, the detailed procedure is given in algorithm 1: if a cluster has less than *to_keep* individuals, they are all kept, and if it has more, only the last *to_remove*, having the largest number of nodes, are removed. Typical values of these parameters are $to\_keep = 7$ and $to\_remove = 1$.

2.3.6 Partial restart scheme

In order to avoid stagnation due to over-specialisation of the best individuals, we propose to periodically add "fresh blood" to the population (i.e. new random

individuals) if a stagnation criterion is fulfilled. The corresponding algorithm uses one parameter denoted *to_insert*, typically set to a lower value than *to_keep*, see algorithm 2.

Input: population of size N
Output: population of size between N and N_{max}
creation of a fresh population of $N_{max} - N$ individuals randomly created
foreach *individual of the fresh population* **do**
  if *size of cluster in which the individual fits lower than to_insert* **then**
    insert the individual into the corresponding cluster of the old population
  end
end

Algorithm 2: Partial restart

In this way, if a cluster of the old population is empty or has not enough elements according to a stricter rule than during the elimination process, it gets new elements. Moreover, the size of the subpopulation to be included being $N_{max} - N$, the final population is insured to be between N and N_{max}.

2.3.7 Criterion of stagnation

If the last improvement of the global fitness is older (in terms of generations) than *stagnation_threshold*, then the partial restart is triggered.

2.3.8 Deflation-inflation scheme

It is made of the following steps (see figure 10):

- **mutations and crossover** yield a temporary population $tmppop$
- **local fitness** is computed on the temporary population: $localfitness(tmppop)$
- **adjusted fitness** is computed via sharing: $sharing(pop + tmppop)$
- **selection** of the N best individuals: $pop = survival(pop + tmppop)$
- **elimination** of the useless individuals with algorithm 1: $pop = elimination(pop)$
- **global fitness** computation of the global fitness of the population: $globalfitness(pop)$
- **partial restart** if a stagnation criterion is met, using algorithm 2: $pop = restart(pop)$

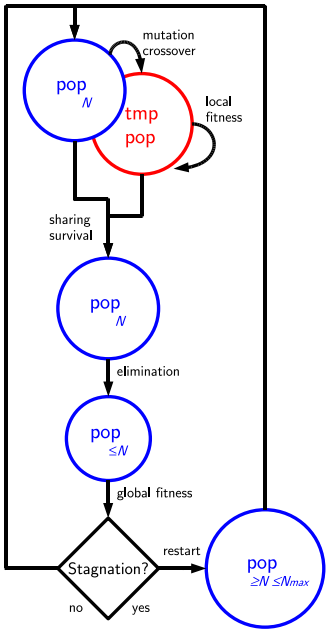


Figure 10: Deflation-inflation scheme

2.3.9 Typical runs

Before introducing a complete statistical analysis, here is a preview of typical runs of each scheme, namely "fixed size" on figure 11, "deflating only" on picture 12, and "inflating+deflating" on picture 13.

When the size of the population is fixed, the total number of individuals is of course constant (here equals to 1000) but one can see that inside this population, the number of representatives of each class is quite balanced, and the number of unique individuals is also quite stable. But the drawback is that the global fitness is very irregular, and gets improvements only at the beginning of the evaluations and then stagnates.

With the deflating-only scheme, the population is slowly decreasing because we eliminate useless individuals. One can notice that the number of unique individuals gets close to the total number of individuals at the end of the evaluations. Nevertheless, there are still only few improvements of the global fitness and a stagnation is quickly reached.

On the contrary, with the deflating+inflating scheme, there are much more improvements of the global fitness. The final recognition rate on the learning set is better than with the two other schemes. As far as the size of the population is concerned, one can observe the cycles of deflations and partial restart. The population is still quite balanced between the four classes, and the number of unique individuals is also quite stable.

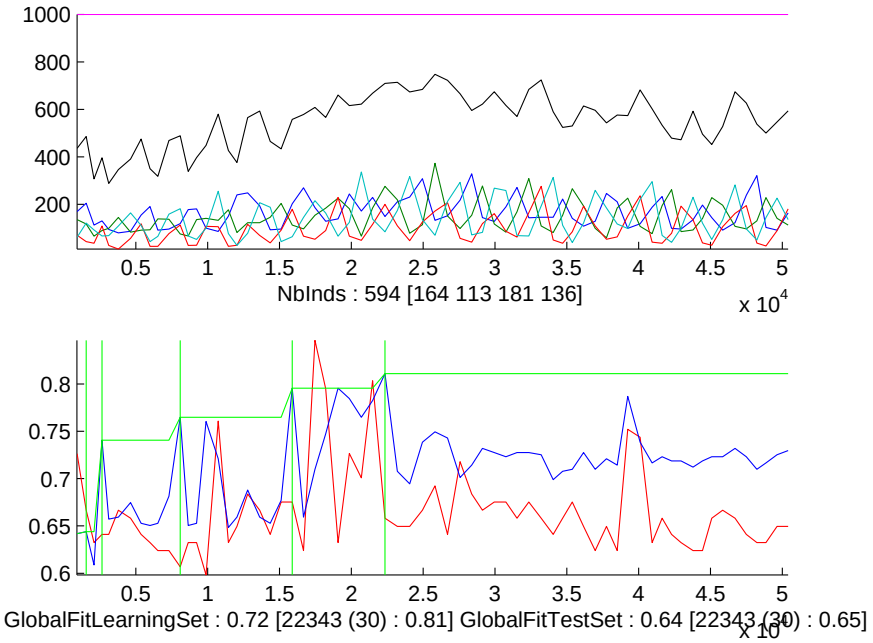


Figure 11: Typical run of a Parisian GP (Fixed size scheme). Top: size of the population and number of unique individuals in each class. Bottom: percentage of correct classification on learning set and test set.

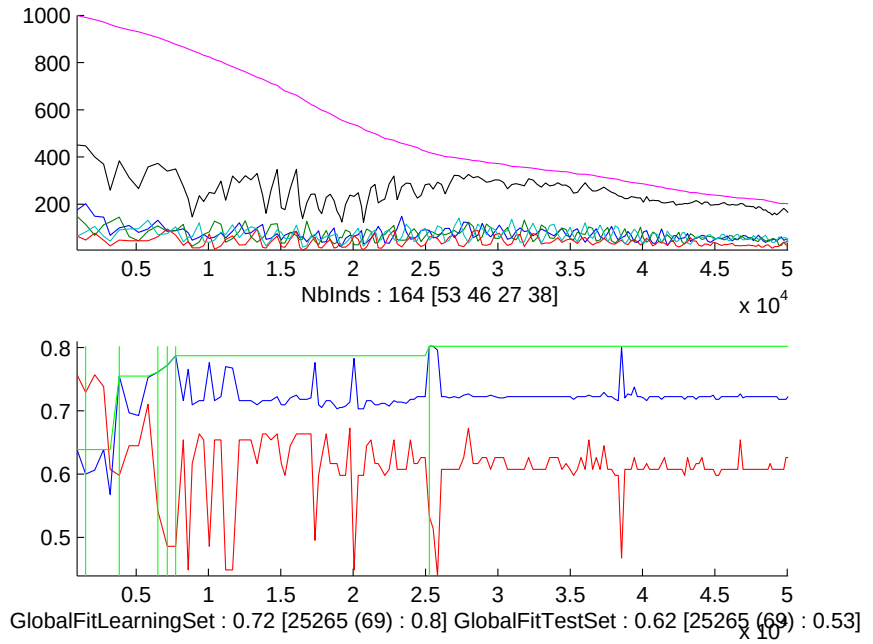


Figure 12: Typical run of a Parisian GP (Deflating only scheme). Top: size of the population and number of unique individuals in each class. Bottom: percentage of correct classification on learning set and test set.

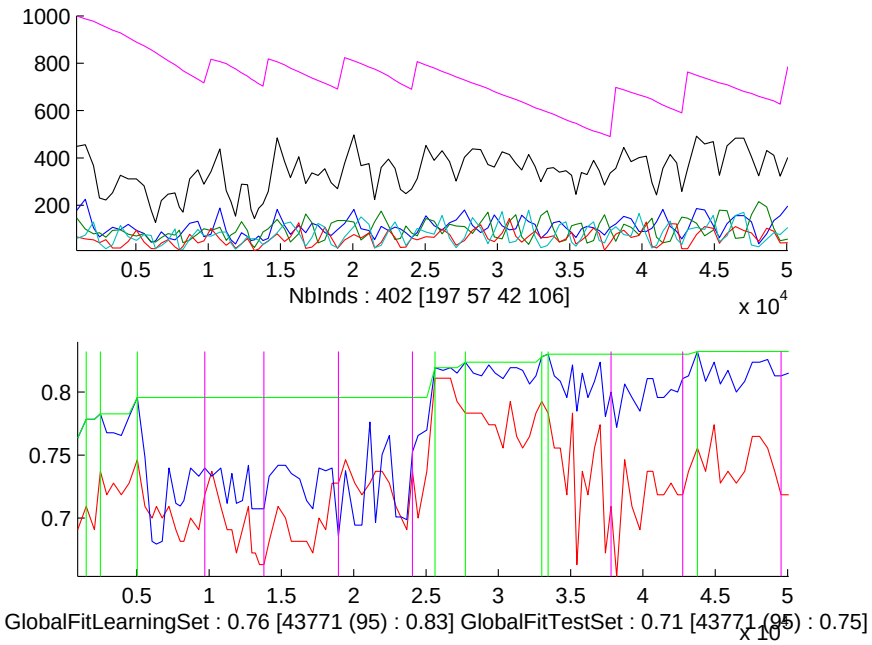


Figure 13: Typical run of a Parisian GP (Deflating+Inflating scheme). Top: size of the population and number of unique individuals in each class. Bottom: percentage of correct classification on learning set and test set.

2.3.10 Experiments

A statistical comparison between the three schemes ("fixed size", "deflating only", and "inflating+deflating") has been performed, based on 100 runs. For each run, we share the 16 experiments into a learning set, made of 10 to 13 randomly chosen experiments, and a test set, made of the rest of the experiments. The three strategies are tested on the same sets during 50000 evaluations and their parameters are detailed in table 3.

	Fixed size	**Deflating-only**	**Deflating-inflating**
Population size	1000	1000, then decreasing	1000, then decreasing and increasing
Clustering parameters	none	$to\_keep = 7$ $to\_remove = 1$	$to\_keep = 7$ $to\_remove = 1$ $to\_insert = 3$
Number of evaluations	50000		
Sharing	$\sigma_{share} = 1$ on the first third of evaluations then linear decrease from 1 to 0.1 $\alpha_{share} = 1$ (constant)		

Table 3: Parameters of the three strategies.

2.3.11 Results

Medians, means and standard deviations have been computed for the percentage of correct classifications on the test and learning sets (see figure 14). Number of evaluations and number of generations to reach the best individual, as well as total number of generations for 50000 evaluations are presented in figure 14 and table 4.

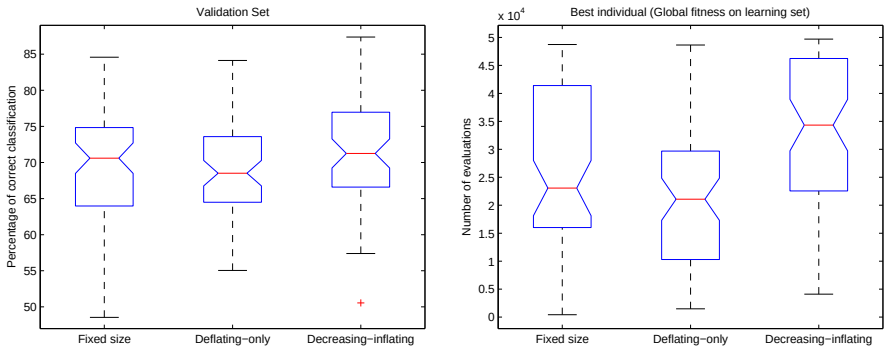


Figure 14: Percentage of correct classification of the best individual on learning set (left) and number of evaluations needed to reach it (right), statistics are made on 100 runs.

Using the fixed sized population as a reference for comparisons, one observes on table 4 that the deflating-inflating scheme allows to gain almost +2% on the test set, whereas the deflating-only scheme reaches almost the same score. The same conclusions can be drawn on the learning set. More precisely, on figure 14

	Fixed size			Deflating-only			Deflating-inflating		
	med	mean	std	med	mean	std	med	mean	std
Correct classification (test set)	70.59	68.93	8.48	68.51	68.69	7.32	71.24	70.96	7.95
Correct classification (learning set)	79.49	79.39	2.75	79.17	78.76	3.26	80.33	80.09	3.27
Number of evaluations (best)	23065	25866	14612	21073	20827	12727	34324	33130	13637
Number of generations (best)	39	41.10	22.9	46	68.3	94.8	70	70.1	32.7
Number of generations (total)	74	75.26	7.77	269	356.3	240.7	98	100.9	12.52

Table 4: Experimental results of the three strategies

it is to be noticed again that the classification on test set is better on average with the deflating-inflating scheme, but also that it has a narrowed range of values, i.e it fails less often.

As far as the number of evaluations is considered, one notices on table 4 and figure 14 that decreasing the size of the population and then increasing it enables to reduce the stagnation effect (the best individual is reached far later). This stagnation effect is more visible with the deflating-only scheme, due to the fact that decreasing the size of the population also decreases its diversity.

2.3.12 Analysis of variance

A one-way ANOVA has been used for comparing the means of the various test-samples[4]. It returns the p-value for the null hypothesis, that is "the two sets are samples of the same mean." We compare strategies two by two, first fixed versus deflating-only, then fixed versus deflating-inflating, and finally deflating-only versus deflating-inflating. Results are given in table 5

	Fixed size VS Deflating-only	Fixed size VS Deflating-inflating	Deflating-only VS Deflating-inflating
Correct classification on the test set	0.8602	0.1627	0.0930
Correct classification on the learning set	0.2331	0.1921	0.0219

Table 5: P-values

A large p-value (close to 1) corresponds to a high probability of having two samples of the same mean. This is the case for the classification on the test set for the fixed size and deflating-only schemes. While deflating-only and deflating-inflating have much lower p-values, meaning that there is a significant statistical difference.

2.4 Conclusion

This first attempt to manage varying population sizes within a Parisian GP scheme show the effectiveness of the population deflation-inflation scheme in

[4]This test supposes that the distributions of the samples are Gaussian, which is obviously not the case here. In the absence of additional hypotheses, the p-value however provides a quite good measurement of the similarities of samples distributions.

terms of computational gain and quality of results on a real problem. The deflating scheme allows to obtain the same result as the fixed-size population strategy, but using less fitness evaluations. The deflating-inflating strategy improves the quality of results for the same number of fitness evaluations as the fixed-size strategy.

In general the development of a monopopulation cooperative-coevolution GP scheme is very attractive as it allows to evolve simpler structures during less generations, and yield results that are usually easier to interpret. However, as one "Parisian" generation necessitates more complex operations, one must carefully consider the global gain of such a procedure (in terms of fitness evaluation or even global computation time). The implementation of a population deflating-inflating scheme is another way to spare computational power, as it allows to avoid redundancy while regularly renewing population diversity.

More generally, the deflation-inflation scheme has two major characteristics: a clusterisation-based redundancy pruning and a selective inflation, which tries to maintain limited-size clusters with low complexity individuals. These two concurrent mechanisms tends to better maintain low complexity individuals as well as genetic diversity. These characteristics may actually be transposed to classical GP or EAs, in particular to limit GP-bloat effects.

3 Bayesian Network Structure estimation using CCEAs

Bayesian networks structure learning is a NP-Hard problem [12], which has applications in many domains, as soon as we try 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, like cheese ripening [5].

Evolutionary techniques have been used to solve the Bayesian network structure learning problem, and were facing crucial problems like: Bayesian network representation (an individual being a whole structure like in [34], or a sub-structures like in [42]), and fitness function choice [42]. Various strategies have been used, based on evolutionary programming [55], immune algorithms [32], multi-objective strategies [51], lamarkian evolution [56] or hybrid evolution [58].

In this work, we propose 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. Furthermore, it is suited to a cooperative coevolution scheme, which allows to obtain computationally efficient algorithms.

Some notions related to Bayesian networks and independence models are recalled in section 3.2. Then, section 3.3 sketches the components of the evolutionary algorithm that is used to solve the first step of IMPEA. The second step of the algorithm is detailed in section 3.7. Experiments are described and analysed in section 3.8, before concluding in section 3.9.

3.1 Background on probability concepts

The joint distribution of X and Y is the distribution of the intersection of the random variables X and Y, that is, of both random variables X and Y occurring together. The *joint probability* of X and Y is written $P(X, Y)$. The *conditional probability* is the probability of some random variable X, given the occurrence of some other random variable Y and is written $P(X|Y)$.

To say that two random variables are *statistically independent* intuitively means that the occurrence of one random variable makes it neither more nor less probable that the other occurs. If two random variables X and Y are independent, then the conditional probability of X given Y is the same as the unconditional probability of X, that is $P(X) = P(X|Y)$.

Two random variables X and Y are said to be *conditionally independent* given a third random variable Z if knowing Z gives no more information about X once one knows Y. Specifically, $P(X|Z) = P(X|Y, Z)$. In such a case we say that X and Y are conditionally independent given Z and write it $X \perp\!\!\!\perp Y \mid Z$.

3.2 Bayesian networks

A Bayesian Network (BN) is a "graph-based model of a joint multivariate probability distribution that captures properties of conditional independence between variables" [25]. On the one hand, it is a graphical representation of the joint

probability distribution and on the other hand, it encodes independences between variables. For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases (i.e. inference).

Formally, a Bayesian networks is a directed acyclic graph (DAG) whose nodes represent variables, and whose missing edges encode conditional independences between the variables. This graph, represented on figure 15, is called the structure of the network and the nodes containing probabilistic information are called the parameters of the network.

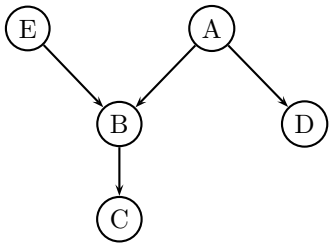


Figure 15: Directed Acyclic Graph

The set of parent nodes of a node X_i is denoted by $pa(X_i)$. In a Bayesian network, the joint probability distribution of the node values can be written as the product of the local probability distribution of each node and its parents:

$$P(X_1, X_2, \ldots, X_n) \quad = \quad \prod_{i=1}^{n} P(X_i | pa(X_i))$$

3.2.1 Uses of Bayesian networks

Using a Bayesian network can save considerable amounts of memory, if the dependencies in the joint distribution are sparse. For example, a naive way of storing the conditional probabilities of 10 binary variables as a table requires storage space for $2^{10} = 1024$ values. If the local distributions of no variable depends on more than 3 parent variables, the Bayesian network representation only needs to store at most $10 * 2^3 = 80$ values. One advantage of Bayesian networks is that it is intuitively easier for a human to understand (a sparse set of) direct dependencies and local distributions than complete joint distribution.

Lastly, more than just a computing tool, Bayesian networks can be used to represent causal relationships and appear to be powerful graphical models of causality.

3.2.2 Parameter and structure learning

The Bayesian network learning problem has two branches: the *parameter* learning problem (i.e., to find the probability tables of each node) and the *structure* learning problem (i.e., to find the graph of the network), following the decomposition of the two constitutive parts of a Bayesian network: its structure and its parameters.

There already exists algorithms specially suited to the parameter learning problem, like expectation-maximization (EM) that is used for finding maximum likelihood estimates of parameters.

Learning the structure is a more challenging problem because the number of possible Bayesian network structures (NS) grows superexponentially with the number of nodes [50]. For example, $NS(5) = 29281$ and $NS(10) = 4.2 \times 10^{18}$. A direct approach is intractable for more than 7 or 8 nodes, it is thus necessary to use heuristics in the search space.

In a comparative study by O. Francois and P. Leray [23], authors identified some currently used structure learning algorithms, namely *PC* [53] or *IC/IC** [46] (causality search using statistical tests to evaluate conditional independence), *BN Power Constructor (BNPC)* [10] (also uses conditional independence tests) and other methods based on scoring criterion, such as *Minimal weight spanning tree (MWST)* [15] (intelligent weighting of the edges and application of the well-known algorithms for the problem of the minimal weight tree), *K2* [17] (maximisation of $P(G|D)$ using Bayes and a topological order on the nodes), *Greedy search* [11] (finding the best neighbour and iterate) or *SEM* [24] (extension of the EM meta-algorithm to the structure learning problem). However that may be, the problem of learning an optimal Bayesian network from a given dataset is NP-hard [12].

3.2.3 The PC algorithm

PC, the reference causal discovery algorithm, was introduced by Sprites, Glymour and Scheines in 1993 [53]. A similar algorithm, IC, was proposed simultaneously by Pearl and Verma [46]. 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. Here are the first detailed steps of this algorithm:

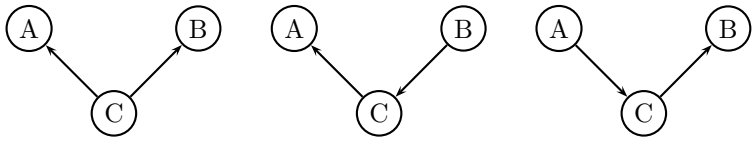
- Step 0: Start with a complete undirected graph G
- Step 1: Test all conditional independences of order 0 (i.e $x \perp\!\!\!\perp y \mid \emptyset$ where x and y are two distinct nodes of G). If $x \perp\!\!\!\perp y$ then remove the edge $x - y$.
- Step 2: Test all conditional independences of order 1 (i.e $x \perp\!\!\!\perp y \mid z$ where x, y, and z are three distinct nodes of G). If $x \perp\!\!\!\perp y \mid z$ then remove the edge $x - y$.
- step 3: Test all conditional independences of order 2 (i.e $x \perp\!\!\!\perp y \mid \{z_1, z_2\}$ where x, y, z_1 and z_2 are four distinct nodes of G). If $x \perp\!\!\!\perp y \mid \{z_1, z_2\}$ then remove the edge $x - y$.
- ...
- Step k: Test all conditional independences of order k (i.e $x \perp\!\!\!\perp y \mid \{z_1, z_2, \ldots, z_k\}$ where $x, y, z_1, z_2, \ldots, z_k$ are $k + 2$ distinct nodes of G). If $x \perp\!\!\!\perp y \mid \{z_1, z_2, \ldots, z_k\}$ then remove the edge between $x - y$.

- Next steps take particular care to detect some structures called *V-structures* (see section 3.2.4) and recursively detect orientation of the remaining edges.

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.

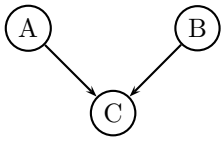
3.2.4 Independence models

As we have seen, a Bayesian network represents a factorization of a joint probability distribution, but there can be many possible factorizations representing the same joint probability distribution. Two structures are said to be *Markov equivalent* if they represent the same joint probability distribution.



$P(A|C)P(B|C)P(C)$ $P(A|C)P(B)P(C|B)$ $P(A)P(B|C)P(C|A)$
$A \perp\!\!\!\perp B \mid C$ $A \perp\!\!\!\perp B \mid C$ $A \perp\!\!\!\perp B \mid C$

These tree structures encode the same independence statement $A \perp\!\!\!\perp B \mid C$.



$P(A)P(B)P(C|A,B)$
A is NOT independent of B knowing C

This last structure, called *V-structure* (or *collider*), is not Markov equivalent to the three first ones.

In this work, we do not work directly on Bayesian networks but on a more general model called *Independence Model* (IM), which can be seen as the underlying model of Bayesian networks and 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.

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, which corresponds to a cooperative coevolution scheme.

3.3 Evolution of an Independence Model

As in the first part of the report, we use the implementation of cooperative coevolution called *Parisian approach*, described at section 1.4. However, in a pure Parisian scheme, the evaluation of the whole population through the computation of the global fitness is done at each generation and redistributed as a bonus to the individuals who participated in this aggregation. Here, we will only compute the global evaluation at the end, and thus don't use any feedback to the population. This approach has already been used with success for example in real-time evolutionary algorithms, such as the *flies* algorithm [39].

IMPEA is a Parisian Evolutionary Algorithm that consists in two steps. 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 construct the structure of the network.

3.3.1 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, as represented on figure 16.

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 $p < \alpha$.

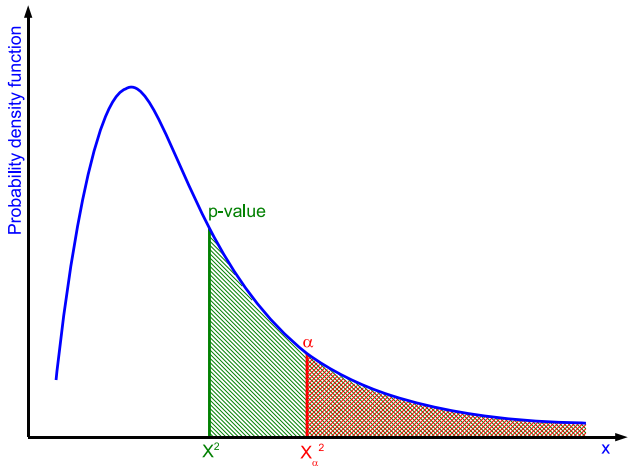


Figure 16: Chi-square test of independence

The reader has to keep in mind that rejecting H_0 allows one to conclude that the two variable are dependent, but not rejecting H_0 means that one cannot conclude that these two variable are dependent (which is not exactly the same as claiming that they are independent). 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 tails vanishes so quickly that individuals with poor p-values are often rounded to 0, making then indistinguishable.

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. Here is the adjusted local fitness[5]:

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

3.3.2 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

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

(see figure 17), 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.

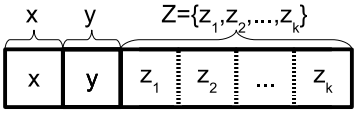


Figure 17: Representation of $\langle x, y|Z\rangle$

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 an 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 18).

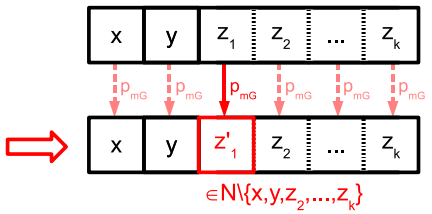


Figure 18: Genome content mutation

- 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 19, 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 consist in a simple swapping mechanism between x, y and Z. Two individuals $\langle x, y|Z\rangle$ and $\langle x', y'|Z'\rangle$ can exchange x or y with

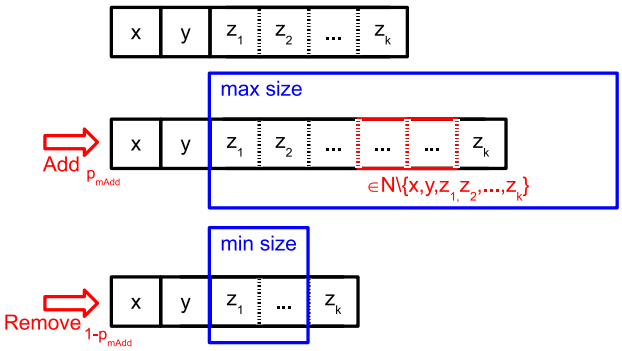


Figure 19: Add/remove mutation

probability p_{cXY} and Z with probability p_{cZ} (see figure 20). 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.

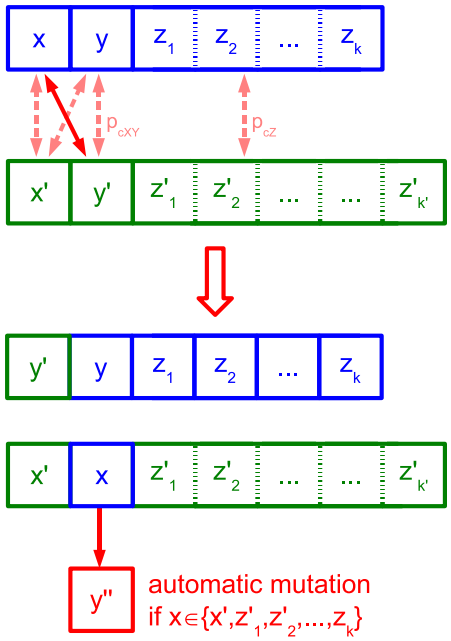


Figure 20: Robust crossover

3.4 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 [19] and is based on the fact that fitness is considered as a shared resource, i.e 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 subpopulations whose size is proportional to the height of the peak they are colonising [26]. 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 [40]. This scheme allows to equilibrate the subpopulations within peaks, whatever their height.

3.5 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 immortals individuals count as neighbours and thus penalize this region, encouraging the exploration of other zones.

3.5.1 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 6 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. Others parameters, like the minimum or maximum number of nodes in Z, are arrays indexed by the number of generations, allowing these parameter to follow a profile of 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 probability of adding nodes in Z along generations	$0.25 \dots 0.75$
MinNodes (#Gen)	Array of minimal number of nodes in Z along generations	$0 \dots 2$
MaxNodes (#Gen)	Array of maximal number 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 6: Parameters of IMPEA. Values are chosen within their typical range depending on the size of the network and the desired computation time.

3.7 Bayesian network structure estimation

The last step of IMPEA consist 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.

3.7.1 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.

3.7.2 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 21.

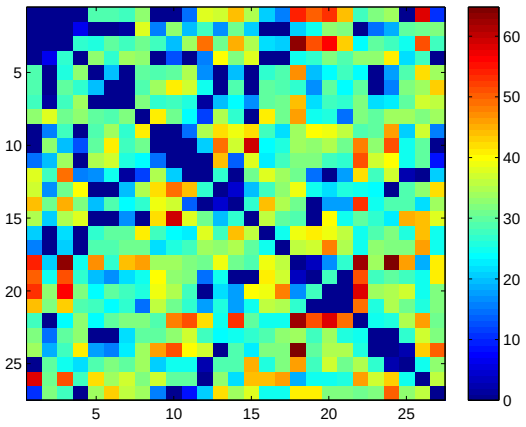


Figure 21: Accumulated adjacency matrix of a network with 27 nodes (from Insurance network).

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.

3.8 Experiments and results

3.8.1 Test case: 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 22. 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.

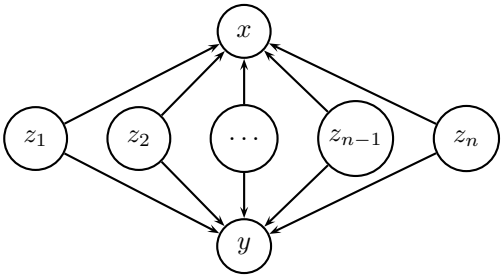


Figure 22: A n-comb network

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* [41] 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 23, 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 an unique big one.

The correct structure (figure 24) is found after 40 (out of 50) generations.

The figure 25 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.

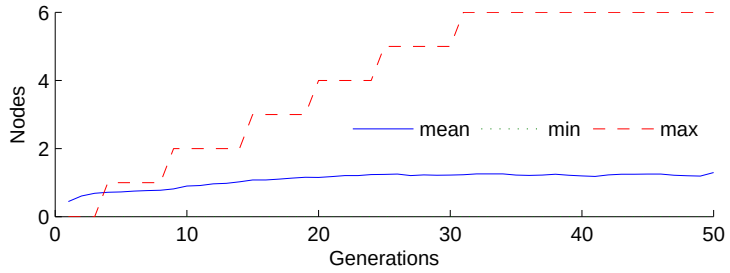


Figure 23: Evolution of Minimal, Maximal and Average number of nodes in Z along generations

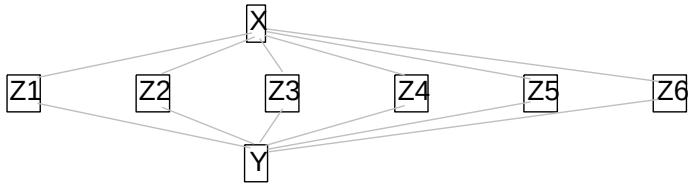


Figure 24: Final evolved structure for the comb network

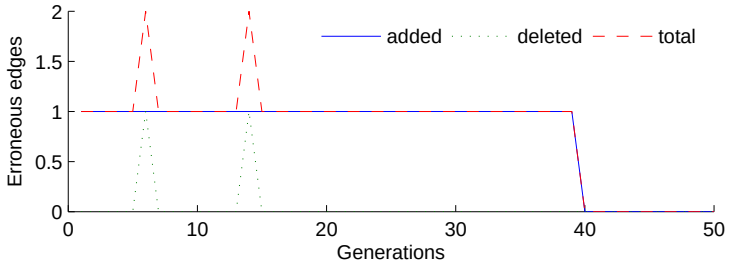


Figure 25: Evolution of the number of erroneous edges of the structure along generations

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.
- 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 7 and 8):
  - How many edges were found? Among these, how many were erroneous? (added or deleted)
  - Did the algorithm remove the edge $x - y$?
- PC is tuned with a fan-in k equal to 3 and P-IMPEA is tuned with 50 generation of 50 individuals in order to take the same computational time as PC. They both share the same significance level α.

The actual network contains 8 edges and 6 nodes. Therefore, the number of possible alternative 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 row of table 7) 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.

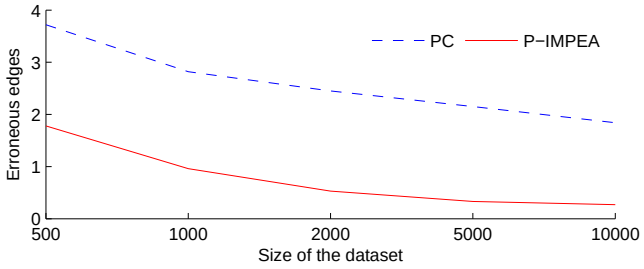


Figure 26: Number of erroneous edges (added+deleted) for PC and P-IMPEA, depending on the size of the dataset

Regarding the global performance, the figure 26 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.

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 7: Averaged results of PC algorithm after 100 runs

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 whereas it is greater than 9 for the PC algorithm since it cannot remove the $x - y$ edge.

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 8: Averaged results of P-IMPEA algorithm after 100 runs

3.8.2 Classical benchmark: the Insurance Bayesian network

Insurance [6] is a network for evaluating car insurance risks. The Insurance Bayesian network contains 27 variables and 52 arcs (figure 27). 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 28 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.

Concerning the evolution of the number of erroneous edges, represented on figure 29, 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 9). When asking for 52 edges, the actual number of edges in the Insurance network, it makes 14 errors (7 additions and 7 deletions).

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 9: Number of detected edges for all algorithms

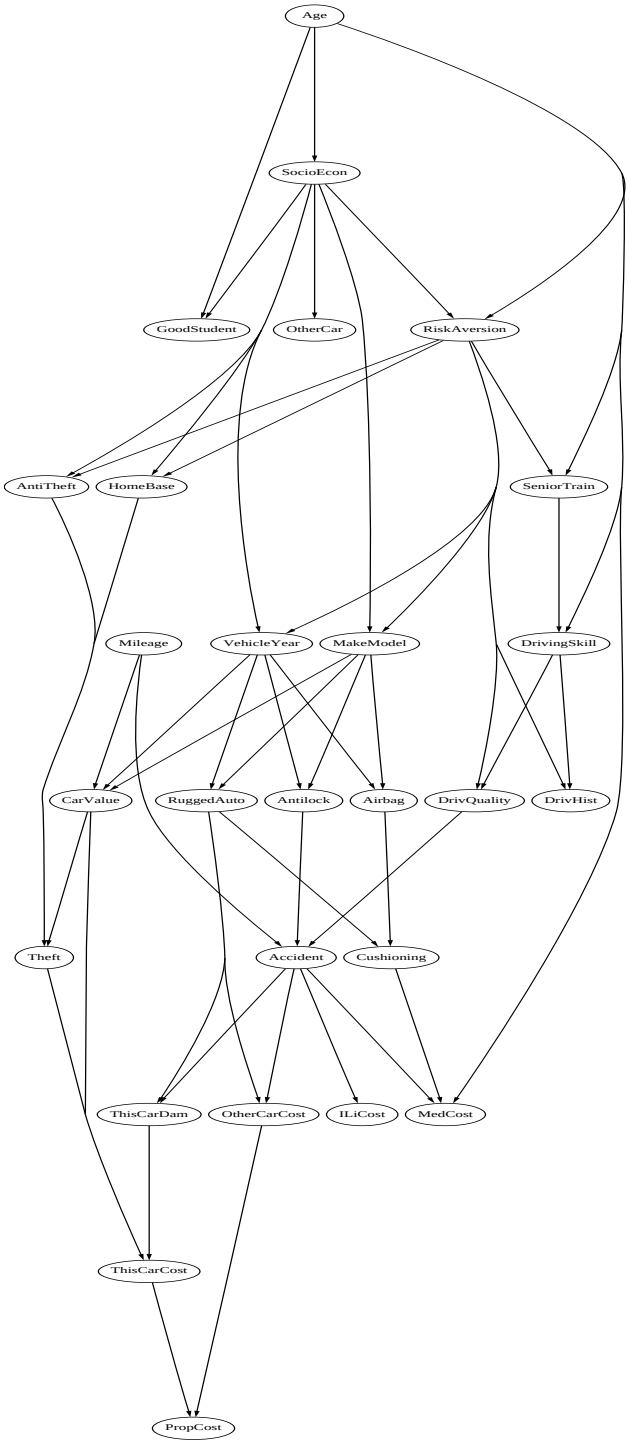


Figure 27: The Insurance Bayesian Network

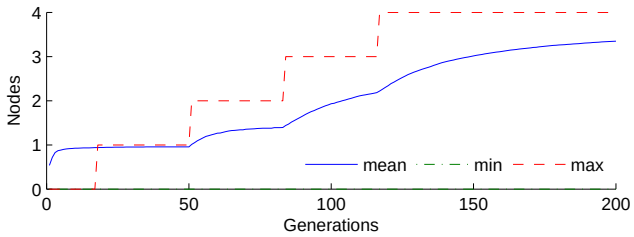


Figure 28: Evolution of Minimal, Maximal and Average number of nodes in Z along generations

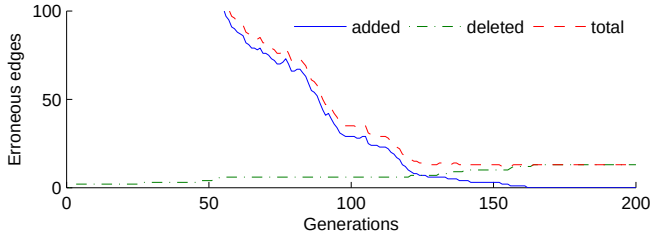
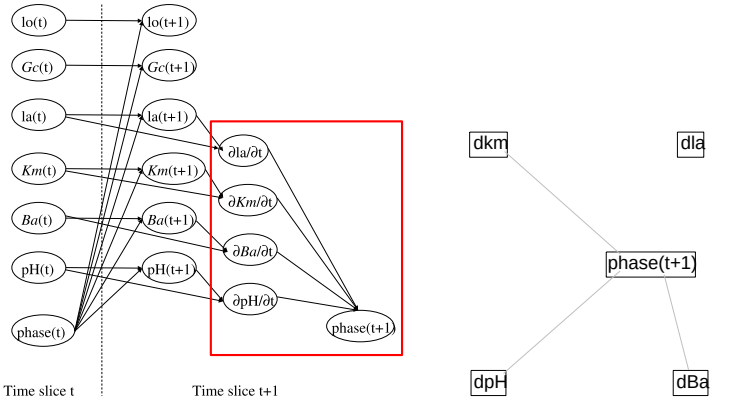


Figure 29: Evolution of the number of erroneous edges of the structure along generations

3.8.3 Real Dataset: Cheese ripening data from the INCALIN project

The last step is to test our algorithm on real data. Our aim is to compare the result of IMPEA with a part of the dynamic Baysian network, already described at section 2, built with human expertise in the scope of the INCALIN project. We are interested in the part of the network that predicts the current phase knowing the derivatives of some bacteria proportions. We used the same data as in the first part of the report (see section 2.2.5), made of the derivatives of pH, la, Km and Ba and estimation of the current phase done by an expert.

After 10 generations of 25 individuals each, P-IMPEA converges to a network whose structure is almost the same as the one proposed by the expert. As one can see on the right of figure 30, no extra edge is added, but the edge between the derivative of la and the phase is missing.



(a) Dynamic Bayesian Network proposed by cheese ripening experts.

(b) Results of P-IMPEA.

Figure 30: Comparison between the model proposed by experts and the network found by IMPEA on a real dataset from the INCALIN project.

3.9 Conclusion

In this work 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 3.8 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 to capture a complex dependency, even on a small network. Additionally it is to be noticed that IMPEA better resists to a current problem of real life data, that is the insufficient number of available samples.

Future work on this topic will be devoted to statistical tests on large benchmarks and on a real industrial agrifood application, where we will have to consider incomplete data.

References

[1] M. Aldarf, F. Fourcade, A. Amrane and Y. Prigent. Substrate and metabolite diffusion within model medium for soft cheese in relation to growth of Penicillium camembertii. *J. Ind. Microbiol. Biotechnol.*, vol. 33, 685–692, 2006.

[2] K. Arfi, F. Amrita, H.E. Spinnler and P. Bonnarme. Catabolism of volatile sulfur compounds precursors by Brevibacterium linens and Geotrichum

candidum, two microorganisms of the cheese ecosystem. *J. Biotechnol.*, vol. 105, Issue 3, 6,p p245-253, 2003.

[3] 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.

[4] D. Barile, J.D. Coisson, M. Arlorio and M. Rinaldi. Identification of production area of Ossolano Italian cheese with chemometric complex aproach, *Food Control*, Volume 17, Issue 3, March 2006, Pages 197-206.

[5] 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*.

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

[7] J. Bongard and H. Lipson. Active Coevolutionary Learning of Deterministic Finite Automata, Journal of Machine Learning Research 6, pp 1651-1678, 2005.

[8] R. Boutrou and M. Guguen. Interests in Geotrichum candidum for cheese technology. *Int. J. Food Microbiol.*, 102, 1-20, 2005.

[9] A. Bucci and J. B. Pollack. On identifying global optima in cooperative coevolution, GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary, Washington DC, USA, 2005.

[10] 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.

[11] 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.

[12] 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.

[13] C. Choisy, M.J. Desmazeaud, J.C. Gripon, G. Lamberet and J. Lenoir. La biochimie de l'affinage, p. 86-105. In *Le fromage*. A. Eck and J.C. Gillis, ed Tec Doc Lavoisier, Paris, 1997.

[14] C. Choisy, M.J. Desmazeaud, M. Gueguen, J. Lenoir, J.L. Schmidt and C. Tourneur. Les phénomènes microbiens, p. 377-446. In *Le fromage*. A. Eck and J.C. Gillis, ed Tec Doc Lavoisier, Paris, France, 1997

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

[16] P. Collet, E. Lutton, F. Raynal and M. Schoenauer. Polar IFS + Parisian Genetic Programming = Efficient IFS Inverse Problem Solving, *In Genetic Programming and Evolvable Machines Journal*, Volume 1, Issue 4, pp. 339-361, October, 2000.

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

[18] L. Davis. Adapting Operators Probabilities in Genetic Algorithms. 3rd ICGA conference, 1989, Morgan-Kaufmann pp 61-69.

[19] 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.

[20] E.D. De Jong, K.O. Stanley and R.P. Wiegand. Introductory tutorial on coevolution, ECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation, London, United Kingdom, 2007.

[21] A.E. Eiben, E. Marchiori and V.A. Valkò. Evolutionary algorithms with on-the-fly population size adjustment, Parallel Problem Solving from Nature PPSN VIII, LNCS 3242, 2004

[22] D.I. Ellis, D. Broadhurst and R. Goodacre. Rapid and quantitative detection of the microbial spoilage of beef by Fourier transform infrared spectroscopy and machine learning, *Analytica Chimica Acta*, Volume 514, Issue 2, 1 July 2004, Pages 193-201

[23] 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.

[24] 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.

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

[26] 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.

[27] S. Silva. GPLAB A Genetic Programming Toolbox for MATLAB, `http://gplab.sourceforge.net/`

[28] A. Gripon. Mould-ripened cheeses. In *Cheese: Chemistry, Physics and Microbiology*, 2, 111-136, 1993 (Ed. PF Fox). London, United Kingdom: Chapman & Hall.

[29] I. Ioannou, G. Mauris, G. Trystram and N. Perrot. Back-propagation of imprecision in a cheese ripening fuzzy model based on human sensory evaluations. *Fuzzy Sets And Systems*, vol. 157, 1179–1187, 2006.

[30] I. Ioannou, N. Perrot, C. Curt, G. Mauris and G. Trystram. Development of a control system using the fuzzy set theory applied to a browning process - a fuzzy symbolic approach for the measurement of product browning: development of a diagnosis model - part I. *Journal Of Food Engineering*, vol. 64, 497–506, 2004.

[31] I. Ioannou, N. Perrot, G. Mauris and G. Trystram. Development of a control system using the fuzzy set theory applied to a browning process - towards a control system of the browning process combining a diagnosis model and a decision model - part II. *J. .Food Eng.*, (64), 507-514, 2004.

[32] H. Jia, D. Liu, and P. Yu. Learning dynamic bayesian network with immune evolutionary algorithm. 2005.

[33] S.A Jimenez-Marquez, J. Thibault and C. Lacroix. Prediction of moisture in cheese of commercial production using neural networks. *Int. Dairy J.*, vol. 15, 1156–1174, 2005.

[34] 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.

[35] M.N. Leclercq-Perlat, D. Picque, H. Riahi and G. Corrieu. Microbiological and Biochemical Aspects of Camembert-type Cheeses Depend on Atmospheric Composition in the Ripening Chamber. *J. Dairy Sci*, (89), pp. 3260-3273, 2006..

[36] M.N. Leclercq-Perlat, F. Buono, D. Lambert, E. Latrille, H.E. Spinnler and G. Corrieu. Controlled production of Camembert-type cheeses. Part I: Microbiological and physicochemical evolutions. *J. Dairy Res.*, (71), pp. 346-354, 2004.

[37] J. Lenoir. The surface flora and its role in the ripening of cheese. *Int Dairy Fed Bull*, 171:3-20, 1984.

[38] F. G. Lobo. A review of adaptive population sizing schemes in genetic algorithms, In Proceedings of the 2005 Workshop on Parameter Setting in Genetic and Evolutionary Algorithms (PSGEA 2005), part of GECCO 2005.

[39] 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.

[40] 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.

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

[42] 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.

[43] H.X. Ni and S. Gunasekaran. Food quality prediction with neural networks. *Food Technology*, vol. 52, 60–65, 1998.

[44] G. Ochoa, E. Lutton and E. Burke. Cooperative Royal Road Functions, *In Evolution Artificielle*, Tours, France, October 29-31, 2007.

[45] L. Panait, S. Luke and J. F. Harrison. Archive-based cooperative coevolutionary algorithms, GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation, Seattle, Washington, USA, 2006.

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

[47] B. Pinaud, C. Baudrit, M. Sicard, P-H. Wuillemin and N. Perrot. Validation et enrichissement interactifs d'un apprentissage automatique des paramètres d'un réseau bayésien dynamique appliqué aux procédés alimentaires, *Journées Francophone sur les Réseaux Bayésiens*, Lyon, France (2008).

[48] E. Popovici and K. De Jong. The Effects of Interaction Frequency on the Optimization Performance of Cooperative Coevolution. In Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2006.

[49] M.H. Riahi, I.C. Trelea, M.N. Leclercq-Perlat, D. Picque and G. Corrieu. Model for changes in weight and dry matter during the ripening of a smear soft cheese under controlled temperature and relative humidity. *International Dairy Journal*, 17, 946–953, 2007.

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

[51] B. J. Ross and E. Zuviria. Evolving dynamic bayesian networks with multiobjective genetic algorithms. *Applied Intelligence*, 26, 2007.

[52] D. Schlierkamp-voosen and H. Mühlenbein. Adaptation of population sizes by competing subpopulation, In Proceedings of the 1996 IEEE Conference on Evolutionary Computation, Piscataway.

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

[54] C.D. Tarantilis and C.T. Kiranoudis. Operational research and food logistics, *Journal of Food Engineering*, Volume 70, Issue 3, October 2005, Pages 253-255

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

[56] 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.

[57] R.P. Wiegand and M.A. Potter. Robustness in cooperative coevolution, GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation, Seattle, Washington, USA, 2006.

[58] 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.

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 Orsay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399