Cooperative coevolution for agrifood process modeling

Olivier Barrière, Evelyne Lutton,
Pierre-Henri Wuillemin,
Cédric Baudrit, Mariette Sicard and Nathalie Perrot

Abstract On the contrary to classical schemes of evolutionary optimisations algorithms, single population Cooperative Co-evolution techniques (CCEAs, also called "Parisian" approaches) make it possible to represent the evolved solution as an aggregation of several individuals (or even as a whole population). In other words, each individual represents only a part of the solution. This scheme allows simulating the principles of Darwinian evolution in a more economic way, which results in gain in robustness and efficiency. The counterpart however is a more complex design phase. In this chapter, we detail the design of efficient CCEAs schemes on two applications related to the modeling of an industrial agri-food process. The experiments correspond to complex optimisations encountered in the modeling of a Camembert-cheese ripening process. Two problems are considered:

- A deterministic modeling problem, phase prediction, for which a search for a closed form tree expression is performed using genetic programming (GP).
- A Bayesian network structure estimation problem. The novelty of the proposed approach is based on the use of a two step process based on an intermediate representation called *independence model*. The search for an independence model is formulated as a complex optimisation problem, for which the CCEA scheme is particularly well suited. A Bayesian network is finally deduced using a deterministic algorithm, as a representative of the equivalence class figured by the independence model.

Olivier Barrière and Evelyne Lutton
INRIA Saclay - Ile-de-France, Parc Orsay Université, 4, rue Jacques Monod, 91893 Orsay Cedex, France, e-mail: olivier.barriere@inria.fr,evelyne.lutton@inria.fr

Pierre-Henri Wuillemin
LIP6-CNRS UMR7606, 75016 Paris, France

Cédric Baudrit, Mariette Sicard and Nathalie Perrot
UMR782 Génie et Microbiologie des Procédés Alimentaires. AgroParisTech, INRA, F-78850 Thiverval-Grignon, France

1 Introduction

Cooperative Co-evolution strategies rely on a formulation of the problem as a cooperative task, where individuals collaborate in order to build a solution.

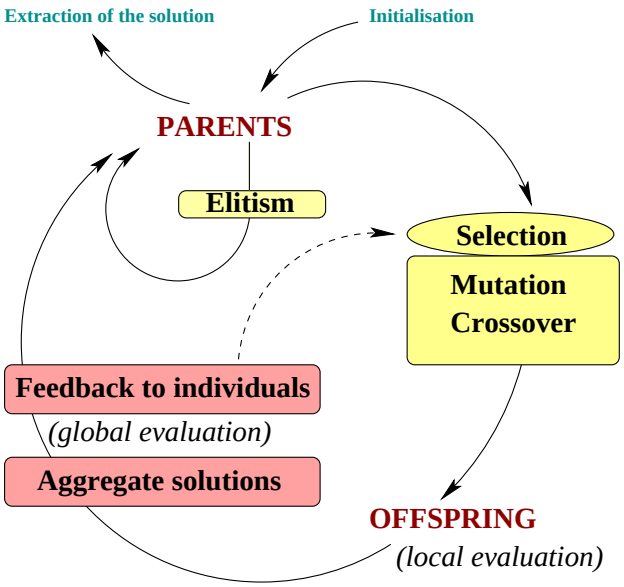


Fig. 1 A Parisian EA: a single population cooperative co-evolution

The large majority of these approaches deals with a co-evolution process that happens between a fixed number of separated populations. The idea is to co-evolve various species that only interact via the evaluation process. [30, 29] were the first to propose this technique, to co-evolve job-shop schedules using a parallel distributed algorithm. [53] then popularize the idea of cooperative co-evolution as an optimisation tool. It is applicable as soon as a decomposition of the problem into subcomponents can be identified. Each component then corresponds to a subpopulation that evolves simultaneously but in isolation to the other subpopulations. Individuals of a subpopulation are evaluated by aggregation with individuals of other subpopulations. Multi-species cooperative co-evolution has been applied to various problems [43, 55, 54, 22, 36, 66], including learning problems [8], and some theoretical analyses have been recently proposed, see [48, 10, 52], or [65] for an analysis considering a relationship between cooperative co-evolution and evolutionary game theory.

In this work, a different implementation of cooperative co-evolution, the so-called Parisian approach [17, 47] is used. It is derived from the classifier systems model proposed by [28]. Shown on Figure 1, this approach uses cooperation mechanisms within a *single* population. On the contrary to the previous model, interactions between sub-species are not limited to the evaluation step, but can also happen

via genetic operators. An individual of a Parisian population, that represents only a part of the solution to the problem, can be evaluated at two levels :

- locally, using an independent evaluation (the “local” fitness), if some criteria can be designed to evaluate partial solutions (for instance, validity conditions),
- globally at each generation, via an aggregation process that builds a solution to the problem to be solved. Individuals are then rewarded via a bonus distribution.

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 within a population, and reduce the computational expense. Successful applications of such a scheme usually rely on a lower cost evaluation of the partial solutions (the individuals of the population), while computing the full evaluation only once at each generation.

The single population approach allows more interaction between subproblems, but in order to avoid trivial solutions (all individuals are the same), diversity preservation becomes a very important mechanism, to favour the evolution of subspecies, that progressively become independent from each other. At least in its early stage, a Parisian approach relies more on “exploration” mechanisms than “exploitation”. Experimental tuning have proven that these two components are balanced in a different manner in classical and Parisian approaches, and that fitness sharing is an important component of Parisian scheme, that ensures an efficient convergence behaviour.

Additionally, we will see in the examples developed in this chapter, that Parisian schemes necessitate a more complex design phase. We actually need to split a problem into interdependent subproblems involving components of the same nature, which is not always possible. Questions regarding the relative efficience of different CCEA approaches, including for instance the single versus multiple population issue are very important, but still open, see for instance [62] for a first attempt in this direction.

This chapter is focussed on the design step, and presents how Parisian approaches have been developed on two examples provided by the agri-food community. The chapter is organised as follows. Section 2 describes the industrial process under study, cheese ripening, and the problems related to expertise modeling in this context. The two examples are then developed:

- section 3 deals with phase estimation using Genetic Programming : for comparison purpose, a classical GP approach is first developed, then a Parisian approach,
- section 4 addresses the problem of evolving the structure of a Bayesian network, with an encoding based on independence models.

Conclusions and future work are given in section 5.

2 Modeling agri-food industrial processes

This study is part of the French INCALIN research project[1]. The goal of this research project was to model agri-food industrial processes. In such food industries, manufacturing processes consist of successive operations whose underlying mechanisms are still unknown, such as the cheese ripening process. INCALIN was concerned with the understanding of the causal relationships between 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 influence those on the macro level. The project aimed to explain the global behaviour of such systems.

Various macroscopic models have embedded expert knowledge, including expert systems [32, 33, 31], neural networks [35, 46], mechanistic models [1, 56], and dynamic Bayesian networks [6].

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 not accurate or even erroneous. For example, a complete cheese ripening process lasts 40 days, and some tests are destructive, that is to say that a cheese sample is consumed during each analysis. Other measurements require the growing of bacterias in Petri dishes and then counting the number of colonies, which is very time consuming. Therefore the precision of the resulting model is often limited by the small number of valid experimental data. Also, parameter estimation procedures have to deal with incomplete, sparse and uncertain data.

2.1 The Camembert-cheese ripening process

"Model cheeses" are produced in laboratories using pasteurized milk inoculated with *Kluyveromyces marxianus* (*Km*), *Geotrichum candidum* (*Gc*), *Penicillium camemberti* (*Pc*) and *Brevibacterium auriantiacum* (*Ba*) under aseptic conditions.

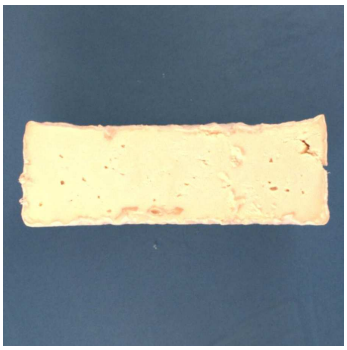
- *K. marxianus* is one of the key flora of Camembert cheese. One of its principal activity is the fermentation of lactose (noted *lo*) [14, 15] (curd de-acidification by lactose consumption). Three dynamics are apparent in the timeline of *K. marxianus* growth [38, 39]. Firstly, there is an exponential growth during about five days that corresponds to a decrease of lactose concentration. Secondly, the concentration of *K. marxianus* remains constant for about fifteen days and then decreases slowly.
- *G. candidum* plays a key role in ripening because it contributes to the development of flavour, taste and aroma of cheeses [2, 9, 40]. One of its principal activi-

[1] "Cognitive and Viability methods for food quality control" (translation from french), supported by the French ANR-PNRA fund.

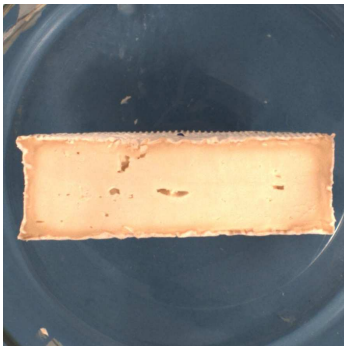
ties is the consumption of lactate (noted *la*). Three dynamics are apparent in the timeline of *G. candidum* growth [38, 39]. First, there is a latency period of about three days. Second, there is an exponential growth that corresponds to a decrease of lactate concentration and thus an increase of pH. Third, the concentration of *G. candidum* remains constant to the end of ripening.

During ripening, soft-mould cheese behave like an ecosystem (a bio-reactor), which is extremely complex to model as a whole. In such a process, human experts operators have a decisive role. Relationships between microbiological and physicochemical changes depend on environmental conditions (temperature, relative humidity ...) [39] and influence the quality of ripened cheeses [27, 38]. A ripening expert is capable of estimating the current state of some complex reactions at a macroscopic level through its perceptions (for example, sight, touch, smell and taste). Control decisions are then generally based on subjective but robust expert measurements. An important factor of parameter regulation is the subjective estimation of the current state of the ripening process. This process is split into four phases:

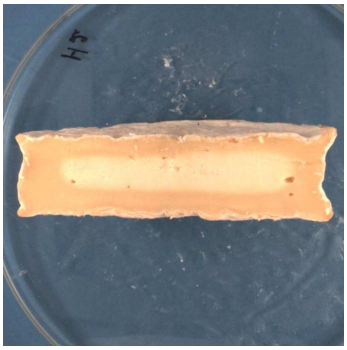
- Phase 1 is characterized by the surface humidity evolution of cheese (drying process). At the beginning, the surface of cheese is very wet and evolves until it is rather dry. The cheese is white with an odor of fresh cheese.



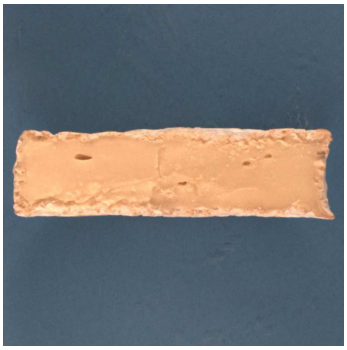
- Phase 2 begins with the apparition of a *P. camemberti*-coat (the white-coat at the surface of cheese). It is characterised by a first change of color and a “mushroom” odor development.



- 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 phases are representative of cheese ripening. The expert's knowledge is obviously not limited to these four phases, but a correct identification of phases helps to evaluate the dynamics of ripening and to detect drift from the standard evolution.

2.2 Modeling expertise on cheese ripening

A major problem, which was addressed in the INCALIN project, is the search for automatic procedures that mimic the way a human aggregates data through his senses, to estimate and regulate the ripening of the cheese.

Stochastic optimisation techniques, like evolutionary techniques, have already been proven successful on several agri-food problems. The interest of evolutionary optimisation methods for the resolution of complex problems related to agri-food is demonstrated by various recent publications. For example, [4] used genetic algorithms to identify the smallest discriminant set of variables to be used in certification process for an Italian cheese (validation of origin labels). [21] 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 of beef. A recent overview on optimisation tools in food industries [61] discusses works based on evolutionary approaches.

We investigate here the use of cooperative co-evolution schemes (CCEAs) in the context of cheese ripening, for the modeling of expert knowledge. The next part (section 3) of this chapter deals with a first problem, which is phase estimation using Genetic Programming, under the form of a simple deterministic model (closed formula). Experimental as well as expert analysis made evident a simple relationship between four derivatives and the phase. A simple scheme they use in practice is based on a multilinear regression model. We will see below that a classical GP approach, that optimises a closed formula, i.e. a non-linear dependency, already improves the recognition rates, and that a Parisian scheme provides similar regognition rates with simpler structures, while keeping good recognition rates when the learning set is small.

The second part of this chapter (section 4) deals with a more sophisticated stochastic model of dependencies: Bayesian Network. The difficult point is now to address the problem of structure learning for a Bayesian Network (BN). Classical approaches of evolutionary computation are usually blocked by the problem of finding an efficient representation of a whole Bayesian Network. We will see that the Parisian scheme allows addressing this issue in an elegant way. In order to validate the method and compare it to the best approaches of the domain, we used classical BN benchmarks before testing it on the cheese ripening data, for which no “ground truth” model exist.

3 Phase estimation using GP

In previous work on cheese ripening modeling [6, 51], a dynamic Bayesian network (Figure 2) has been built, using human expert knowledge, to represent the macroscopic dynamic of each variable. The phase of the network at time t plays a determinant role for the prediction of the variables at time $t+1$. Moreover, four relevant variables have been identified by biologists, the derivative of *pH*, *la* (lactate), *Km* (Kluyveromyces marxianus) and *Ba* (Brevibacterium auriantiacum) at time t, allowing phase prediction at time $t+1$. This relates to a way in which experts aggregate information from their senses.

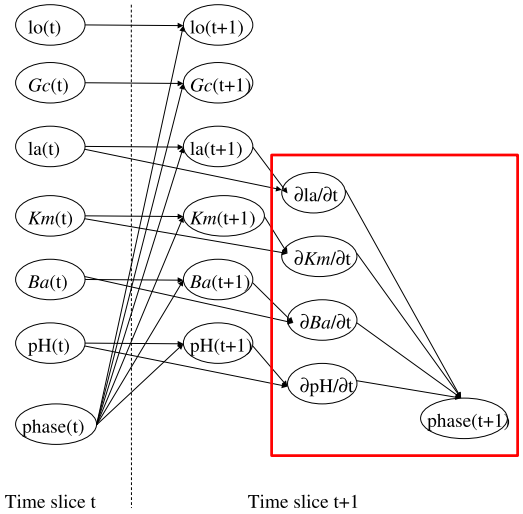


Fig. 2 Dynamic Bayesian Network representing dynamic variables based on the observation of ripening phases. The static Bayesian network used for comparison is in the right hand side box

3.1 Phase estimation using a classical GP

A Genetic Programming (GP[2]) 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 $t-1$.

[2] GP is a type of EA where each individual figures a function, represented as a tree structure. Every tree node is an operator function $(+, -, /, *, \ldots)$ and every terminal node is an operand (a constant or a variable).

When available, a functional representation of dependencies between variables is interesting (for prediction purpose for example). This problem is a symbolic regression one, however the small number of samples and their irregular distribution makes it difficult. In such a case, probabilistic dependencies (like Bayesian networks) seems usually to be more adapted, but are facing the same difficulty (robust estimation when data are sparse). A first question that could be adressed is thus to know which type of representation is more robust when data are sparse.

Results of GP estimation are compared in the sequel with the performances of a static Bayesian network, extracted from the DBN of [6], (the part within the box in Figure 2), and with a simple learning algorithm (multilinear prediction, see section 3.2.6), that was used by biologists in a first approach.

3.1.1 Overview of the classical GP algorithm

The classical GP algorithm consists first of an initialisation step where an initial population is randomly generated and then of a main loop where the reproduction (mutation and crossovers) and selection mechanism (ranking) are applied. The pseudo code of such an algorithm is given as follows:

Input: Maximum number of evaluations
Output: Single best individual
Creation of a random initial population
while *Maximum number of evaluations not reached* **do**
  Create a temporary population *tmppop* using selection, mutations and crossover
  Compute the fitness of the new temporary population *tmppop*
  Select the best individuals of the current population $pop + tmppop$
end
Select the best individual of the final population

Algorithm 1: Classical GP algorithm

3.1.2 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 2.1), for the estimation of the phase (static problem). The GP will search for a phase estimator $\widehat{Phase}(t)$. That is, a function defined as follows (equation 1):

$$\widehat{Phase(t)} = f\left(\frac{\partial pH}{\partial t}, \frac{\partial la}{\partial t}, \frac{\partial Km}{\partial t}, \frac{\partial Ba}{\partial t}\right) \quad (1)$$

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

The terminal set is made of the four partial derivatives plus real constants. The constant's values are not limited and randomly initialised using one of the following laws $\mathscr{U}\ [0,1]$, $-\mathscr{U}\ [0,1]$, $\mathscr{N}\ (0,1)$, randomly chosen. ($\mathscr{U}$ is the uniform law, and $\mathscr{N}$ the normal law).

3.1.3 Fitness function

Available data are separated in two sets: learning set and test set. Each is randomly chosen within the available data set for each run. The 16 available experiments are randomly split between learning and test sets. The size of the learning set varies from 10 to 15 experiments, while the size of the corresponding test set vary from 6 to 1 experiments (see section 3.2.6).

The fitness function (equation 2), *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, measured as the number of nodes (*#Nodes* in equation 2), of the evolved structures. The aim of this factor is to avoid bloat. It is divided by the number of variables (*#Variables* in equation 2) involved in the evaluated tree in order to favour structures that embed all four variables of the problem. Human experts use four classes to quantify the behaviour of the ripening process, and industrial processes are organised accordingly. Another type of classification (i.e. more or less classes) would have a strong impact on industrial devices. We choose to remain consistent with this expert approach. This is important in future developments where interfaces with human experts will be built. Experiments also show that recognition results are better with this constraint.

$$fitness = \frac{\sum_{learning\_set} \left| f\left(\frac{\partial pH}{\partial t}, \frac{\partial la}{\partial t}, \frac{\partial Km}{\partial t}, \frac{\partial Ba}{\partial t}\right) - Phase(t) \right| + W\#Nodes}{\#Variables + 1} \quad (2)$$

The parameter W has been experimentally tuned. A large number of combinations were tested and it turned out that $W = 1$ is the optimal value in terms of algorithmic performance which favours evolution of structures with roughly 30 to 40 nodes. Bigger structures are so penalised that they are excluded from the population during the selection process.

3.1.4 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:

- **Subtree mutation** (mutation of the structure), that randomly rebuilds a new subtree from a randomly chosen node, applied with probability p_{sm} (defined per tree),
- **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. The probabilities (defined per node) are detailed in Table *I*. 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: $x' = x\frac{\sum_{i=1}^{N}\mathscr{N}(0,1)^2}{N}$ p_{rm} and N vary linearly according to generations, from 0.1 (first generation) to 0.5 (last generation) for p_{rm}, and from 1 to 1000 for N. This allows starting 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 [19] available in the GPLAB toolbox [59]. p_c, p_{sm} and p_{cm} 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.

3.2 Phase estimation using a Parisian GP

Instead of searching for a phase estimator as a single monolithic function, phase estimation can be split into four combined (and simpler) phase detection trees as shown in Figure 3. The structures searched are binary output functions (or binarised functions) that characterise one of the four phases. The individuals are split into four classes such that individuals of class k are good at characterising phase k. Finally,

a global solution is made of the 5% best (at least one) individuals 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).

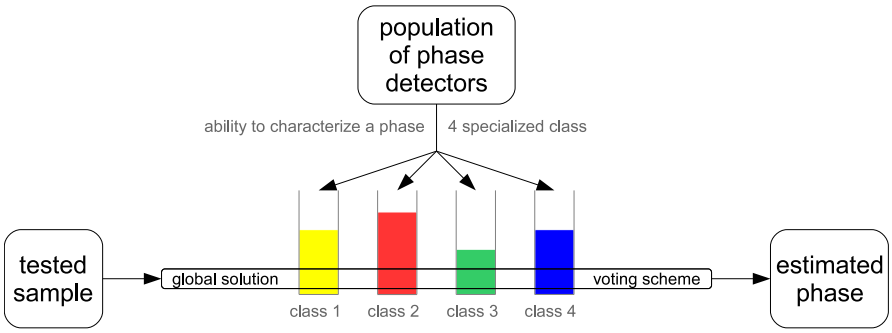


Fig. 3 Phase estimation using a Parisian GP. Four classes of phase detectors are defined: individuals of class k are good at characterising phase k.

3.2.1 Overview of the Parisian GP algorithm

Unlike the classical GP algorithm, the output of Parisian GP algorithm is not a single individual but a part of the population. The main loop of the Parisian GP algorithm consists in first applying reproduction and selection mechanism, and then aggregating the current individuals in order to build a potential solution to the problem. The following pseudo code illustrates the principles of a Parisian GP:

Input: `Maximum number of evaluations`
Output: `Aggregation of individuals`
`Creation of a random initial population`
while *Maximum number of evaluations not reached* **do**
  `Create a temporary population` $tmppop$ `using selection, mutations and crossover`
  `Compute the localfitness of the new temporary population` $tmppop$
  `Compute the adjustedfitness of the current population` $pop+tmppop$ `via sharing`
  `Select the best individuals of the current population` $pop+tmppop$
  `Compute the globalfitness of the selected population by aggregating the best individuals`
end

Algorithm 2: Parisian GP algorithm

3.2.2 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 real values can be computed, in order to measure the capability of an individual I to characterise one phase (equation 3):

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

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

3.2.3 Local fitness

The local adjusted fitness value, *to be maximised*, starts with a combination of three factors (equation 4):

$$\max\{F_1,F_2,F_3,F_4\} \times \frac{\#Ind}{\#IndPhaseMax} \times \frac{NbMaxNodes}{NbNodes}\Big|_{\text{if } NbNodes>NbMaxNodes} \tag{4}$$

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. The parameter *#IndPhaseMax* is the number of individuals representing the phase corresponding to the *argmax* of the first factor. The parameter *#Ind* is the total number of different individuals in the population. The third factor is a parsimony factor for avoiding large structures. *NbMaxNodes* has been experimentally tuned, and is currently fixed to 15, so that evolved structures got enough nodes to characterise the problem, but not too many, to avoid bloat effect. 15 represented a good tradeoff between accuracy and performance.

However, this is not the final formula of the local adjusted fitness. The two following subsection add two more factors, a penalising factor (μ) for individuals with too many neighbours (diversity preservation via a sharing scheme) and a bonus factor $bonus^{\alpha}$ for the best individuals.

3.2.4 Sharing distance

The set of measurements $\{F_1,F_2,F_3,F_4\}$, that measures the ability of an individual to characterise each phase, provides a simplified representation of the search space

in $\mathbb{R}^4$. As the aim of a Parisian evolution is to evolve distinct sub-populations, each being adapted to one of the four subtasks (to characterise 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 as stated in [20].

3.2.5 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 (the argmax of $\max\{F_1,F_2,F_3,F_4\}$ for each individual). In other words, the population is split into four sub-populations (one for each class) within the population. The 5% best of each class are used via a voting scheme to decide the phase of each tested sample (see Figure 3). The global fitness measures the proportion of correctly classified samples on the learning set (equation 5):

$$GlobalFit = \frac{\sum_{learning\_set} CorrectEstimations}{\#Samples} \tag{5}$$

The global fitness is then distributed to individuals who participated in the vote according to the following formula: $LocalFit' = LocalFit \times (GlobalFit + 0.5)^{\alpha}$.

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.

Several fitness measures are used to rate individuals. Namely

- the raw fitness $rawfitness$, which is the set of four values $\{F_1,F_2,F_3,F_4\}$, that measure the ability of the individual to characterise 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.

Two sets of indicators are computed at each generation (see Figure 5):

- The sizes of each class, that show if each phase is equally characterised by the individuals of the population.
- The discrimination capability for each phase, computed on the 5% best individuals of each class as the minimum of: $\Delta = \max_{i\in[1,2,3,4]}\{F_i\} - \frac{\sum_{k\neq argmax\{F_i\}}\{F_k\}}{3}$

The higher the value of Δ, the better the phase is characterised.

3.2.6 Experimental analysis

Available data were collected from 16 experiments during 40 days for each experiment, yielding 575 valid measurements. The data samples are relatively balanced except for phase 3, which has a longer duration, thus a larger number of samples: we have 57 representatives of phase 1, 78 of phase 2, 247 of phase 3 and 93 of phase 4. The derivatives of *pH*, *la*, *Km* and *Ba* were averaged and interpolated (spline interpolation) for some "missing" days. Indeed, due to difficulty to collect experimental data, a few values were missing. Finally, 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 *II*. The code has been developed in Matlab, using the GPLAB toolbox [59]. Comparative results of the four considered methods (multilinear regression, Bayesian network, GP and Parisian GP) are displayed in Figure 4, and a typical GP run is analysed in Figure 5.

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

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

The 5 coefficients $\{\beta_1, \ldots, \beta_5\}$ are estimated using a simple least square scheme. This model was included in the comparison because it was the model previously used by the biologists in the INCALIN project.

Experiments show that GP outperforms both multilinear regression and Bayesian network approaches in terms of recognition rates. Additionally the analysis of a typical Parisian GP run shows that it evolves much simpler structures than the classical GP. 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 5 that co-evolution is balanced between the four phases. The third phase is the most difficult to characterise. This is in accordance with human experts' judgement, for which this phase is also the most ambiguous to

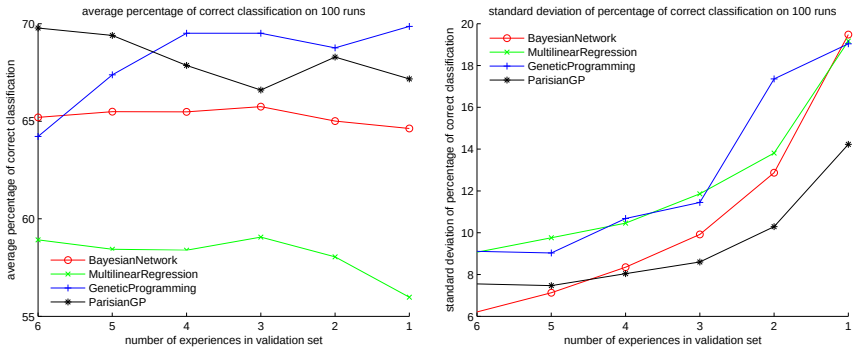
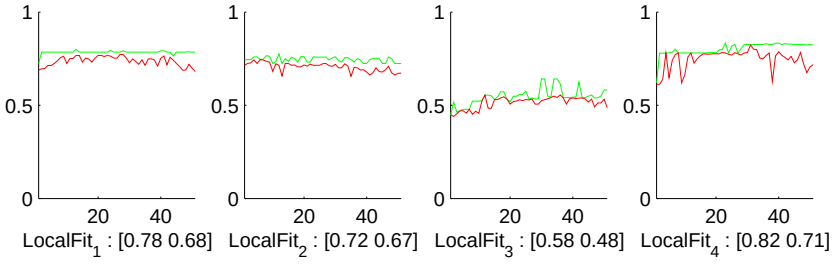


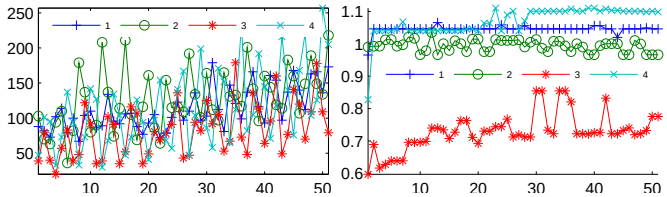
Fig. 4 Average (left) and standard-deviation (right) of recognition percentage on 100 runs for the 4 tested methods, the abscissa represent the size of the test-set

characterise.

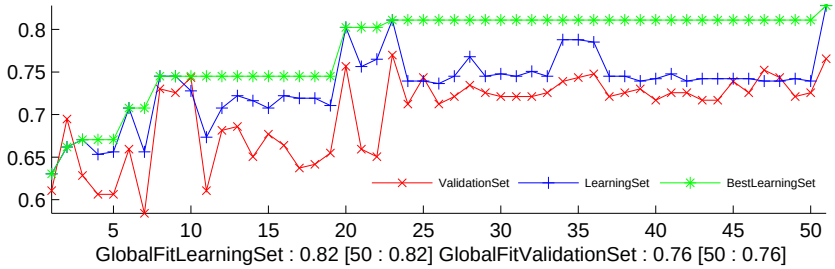
The development of a cooperative co-evolution GP scheme (Parisian evolution) seems very attractive, as it allows the evolution of simpler structure in less generations, and yield results that are easier to interpret. Moreover, the computation time is almost equivalent to both presented methods (100 generations for a classical GP against 50 generations for a Parisian one), as one “Parisian” generation necessitates more complex operations, all in all). One can expect a more favourable behaviour for 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 [17]).



(a) 5% best individuals.



(b) Distribution of individuals / Discrimination indicator.



(c) Recognition rates.

Fig. 5 A typical run of the Parisian GP:
- (a): 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.
- (b) left: the distribution of individuals for each phase: the curves are very irregular but numbers of representatives of each phases are balanced.
- (b) right: discrimination indicator Δ, which shows that the third phase is the most difficult to characterise.
- (c): evolution of the recognition rates of learning and test set. The best-so-far recognition rate on learning set is tagged with a star.

4 Bayesian Network Structure learning using CCEAs

Bayesian networks structure learning is a NP-Hard problem [13], which has applications in many domains, as soon as one tries to analyse a large set of samples in terms of statistical dependence or causal relationship. In agri-food industries for example, the analysis of experimental data using Bayesian networks helps to gather technical expert knowledge and know-how on complex processes [6].

Evolutionary techniques were 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 [37], or a sub-structures like in [45]),
- Fitness function choice like in [45].

Various strategies were used, based on evolutionary programming [3], immune algorithms [34], multi-objective strategies [58], lamarkian evolution [64] or hybrid evolution [67].

We propose here to use an alternate representation, independence models, in order to solve the Bayesian network structure learning in two steps. Independence model learning is still a combinatorial problem, but it is easier to embed within an evolutionary algorithm. Furthermore, it is suited to a cooperative co-evolution scheme, which allows obtaining computationally efficient algorithms.

4.1 Recall of some probability notions

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 Y gives no more information about X once one knows Z. 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 $X \perp\!\!\!\perp Y \mid Z$.

4.2 Bayesian networks

A Bayesian Network (BN) is a "graph-based model of a joint multivariate probability distribution that captures properties of conditional independence between random variables" as defined by [25]. On the one hand, it is a graphical representation of the joint probability distribution and on the other hand, it encodes probabilistic independences between variables. For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms (resp. diseases), the network can be used to compute the probabilities of the presence of various diseases (resp. symptoms). These computations are called probabilistc inference.

Formally, a Bayesian network is represented by a directed acyclic graph (DAG) whose nodes are random variables, and whose missing edges encode conditional independences between the variables.

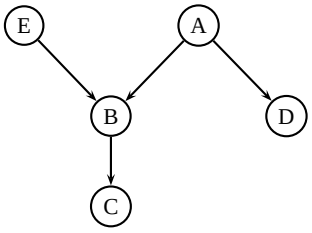


Fig. 6 Directed Acyclic Graph

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 random variables can be written using the graph structure as the product of the conditional probability distributions of each node given its parents:

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

For instance, the joint distribution represented as a Bayesian network in Figure 6 can be written : $P(A,B,C,D,E) = P(A) \cdot P(B|E,A) \cdot P(C|B) \cdot P(D|A) \cdot P(E)$.

The very graph is called the structure of the Bayesian network and the values of conditional probabilities (e.g. $P(A = 0)$) for each node are called the parameters of the network.

4.2.1 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. The graph structure of a Bayesian network also allows to dramatically speed up the probabilistic inference in Bayesian network (i.e. the computation of $P(X_i|X_j)$).

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.

4.2.2 Parameter and structure learning

The Bayesian network learning problem has two branches: the *parameter* learning problem (in other words, how to find the probability tables of each node) and the *structure* learning problem (in other words, how to find the graph representing the Bayesian 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-maximisation (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 [57]. 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 [23], authors identified some currently used structure learning algorithms, namely *PC* [60] or *IC/IC** [50] (causality search using statistical tests to evaluate conditional independence), *BN Power Constructor (BNPC)* [11] (also uses conditional independence tests) and other methods based on scoring criterion, such as *Minimal weight spanning tree (MWST)* [16] (intelligent weighting of the edges and application of the well-known algorithms for the problem of the minimal weight tree), *K2* [18] (maximisation of $P(G|D)$ using Bayes and a topological order on the nodes), *Greedy search* [12] (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 [13].

4.2.3 The PC algorithm

PC, the reference causal discovery algorithm, was introduced by [60]. A similar algorithm, IC, was proposed simultaneously by [50]. 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 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 4.2.4) and recursively detect orientation of the remaining edges.

The first stage is learning associations between variables for constructing an undirected structure. This requires a number of conditional independence test growing exponentially with the number of nodes. This complexity is reduced to polynomial complexity by fixing the maximal number of parents a node can have. It is of the order of N^k, where N is the size of the network and k is the upper bound on the fan-in. This implies that the value of k must remain small when dealing with big networks. In practice, k is often limited to 3. This value will be used in the sequel.

4.2.4 Independence models

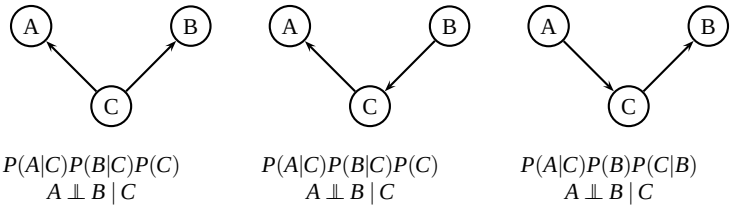
In this work, we do not work directly on Bayesian networks but on a more general model called *Independence Model* (IM), which can be seen as the underlying model of Bayesian networks and defined as follows:

- 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*[3] induced by P.

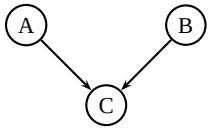
As we have seen, a Bayesian network represents a factorisation of a joint probability distribution, but there can be many possible structures that represents the same probability distribution.

For instance, the tree structures in Figure 7 encode the same independence statement $A \perp\!\!\!\perp B \mid C$. However, the structure in Figure 8, called *V-structure* (or *collider*), is not Markov equivalent to the three first ones.



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

Fig. 7 Markov equivalent structures



$P(A)P(B)P(C|A,B)$
NOT($A \perp\!\!\!\perp B \mid C$) but $A \perp\!\!\!\perp B$

Fig. 8 V-structure

Two structures are said to be *Markov equivalent* if they represent the same Independence Model. Particularly, an algorithm to learn the structure of a Bayesian network can not choose between two markov-equivalent structures.

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

[3] For more details about Independence Models and their properties, see [49].

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 co-evolution scheme.

4.3 Evolution of an Independence Model

As in section 3, our algorithm (Independence Model Parisian Evolutionary Algorithm - IMPEA) is a *Parisian* cooperative co-evolution. However, in a pure Parisian scheme (Figure 1), a multi-individuals evaluation (global fitness computation) is done at each generation and redistributed as a bonus to the individuals who participated in the aggregation. Here, IMPEA only computes the global evaluation at the end of the evolution, and thus do not use any feedback mechanism. This approach, which is an extreme case of the Parisian CCEA, has already been used with success for example in real-time evolutionary algorithms, such as the *flies* algorithm [41].

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

4.3.1 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 9.

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 $p < \alpha$. 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

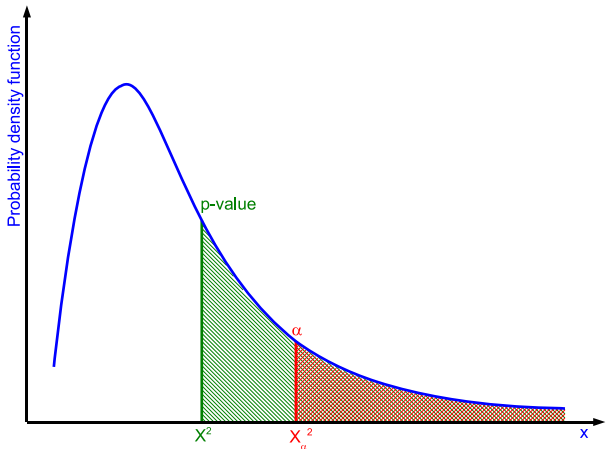


Fig. 9 Chi-square test of independence

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 measures 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 avoiding *plateaus* which would prevent the genetic algorithm to travel all over the search space. Here is the adjusted local fitness[4]:

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

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

4.3.2 Genetic operators

The genome of an individual, being $\langle x, y|Z\rangle$ where x and y are simple nodes and Z is a set of nodes is straightforward: It consists in an array of three cells (see Figure 10), 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.

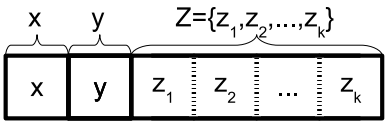


Fig. 10 Representation of $\langle x, y|Z\rangle$

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 11).

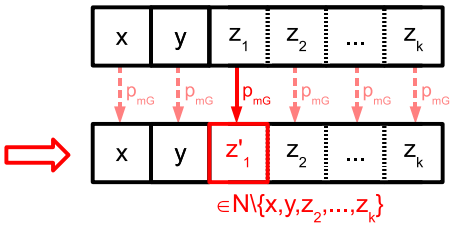


Fig. 11 Genome content mutation

- 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 12, that allows randomly adding or removing 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, for tuning the growth of Z.

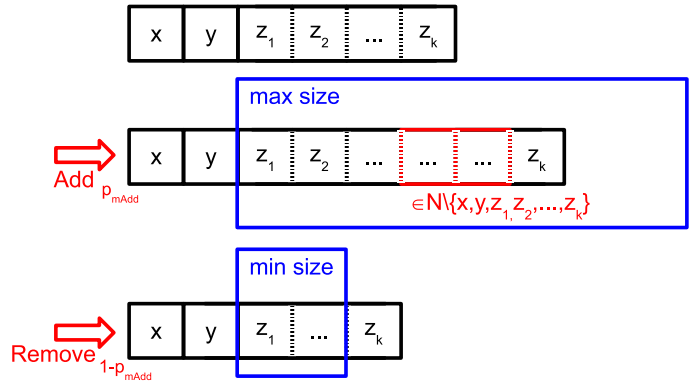


Fig. 12 Add/remove mutation

- Crossover
  The crossover consists 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 probability p_{cXY} and Z with probability p_{cZ} (see Figure 13). 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.

4.4 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 [20] and is based on the fact that fitness is considered as a shared resource, that is to say that individuals having too many neighbours are penalised. Thus we need a way to compute the distance between individuals so that we can count the number of neighbours of a given individual. A simple Hamming distance was chosen: two elementary triplets $\langle x,y|Z\rangle$ and $\langle x',y'|Z'\rangle$ are said to be neighbours if they test the same two nodes (i.e., $\{x,y\} = \{x',y'\}$), whatever Z. Finally, dividing the fitness of each individual by the number of its neighbours would result in sharing the population into sub-populations whose size is proportional to the height of the peak

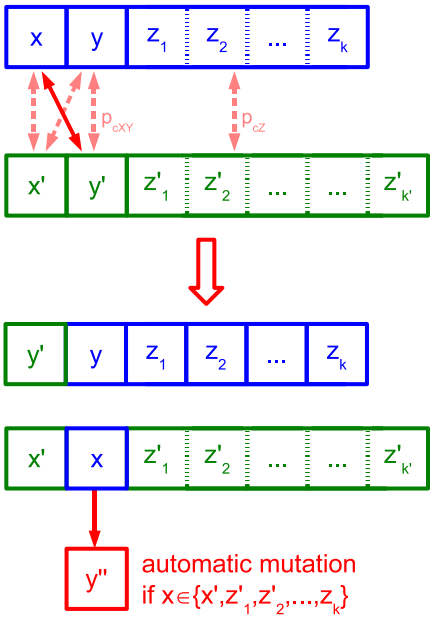


Fig. 13 Robust crossover

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 [42]. This scheme allows one to equilibrate the sub-populations within peaks, whatever their height.

4.5 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 evolving 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 $1-\varepsilon$, 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 penalise this region, encouraging the exploration of other zones.

4.5.1 Clustering and partial restart

Despite the sharing mechanism, we experimentally observed 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.

4.6 Description of the main parameters

The Table 3 describes the main parameters of IMPEA and their typical values or range of values, in order of appearance in the text above. 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.

4.7 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 does not allow rejecting 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 a constrained one, called C-IMPEA, which adds a constraint on the number of desired edges.

Name	Description	Typical value
MaxGens	Number of generations	$50\ldots200$
Ninds	Number of individuals	$50\ldots500$
Alpha	Significance level of the χ^2 test	$0.01\ldots0.25$
Parcim (#Z)	Array of parsimony coefficient (decreases with the length of Z)	$0.5\ldots1$
PmG	Probability of genome content mutation	$0.1/(2+\#Z)$
PmAR	Probability of adding or removing nodes in Z	$0.2\ldots0.5$
PmAdd (#Gen)	Array of probability of adding nodes in Z along generations	$0.25\ldots0.75$
MinNodes (#Gen)	Array of minimal number of nodes in Z along generations	$0\ldots2$
MaxNodes (#Gen)	Array of maximal number of nodes in Z along generations	$0\ldots6$
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	$1\ldots5$

Table 3 Parameters of IMPEA. Values are chosen within their typical range depending on the size of the network and the desired computation time.

4.7.1 Pure conditional independence

Then, as in PC, P-IMPEA starts from a fully connected graph, and for each individual of the aggregated population, 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.

4.7.2 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 14.

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

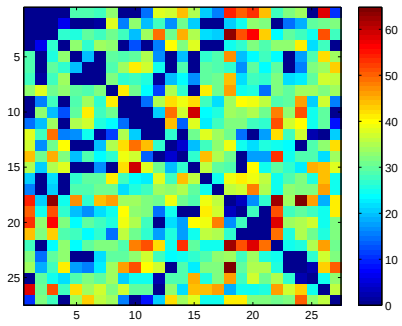


Fig. 14 Accumulated adjacency matrix of a network with 27 nodes (from Insurance network).

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.

4.8 Experiments and results

Prior to a test on the the cheese-ripening data, the experimental analysis has been first performed on simulated data, where the true BN structure is known. A first experiment has been done on a toy-problem (section 4.8.1) in order to analyse the behaviour of IMPEA on a case where the complexity of the dependencies is controlled (i.e. where there is one independence statement that involves a long conditional set Z). A second test has been made on a classical benchmark of the domain, the insurance network (section 4.8.2), where input data are generated from a real-world BN. The test on cheese ripening data is detailed in section 4.8.3.

4.8.1 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 15. 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.

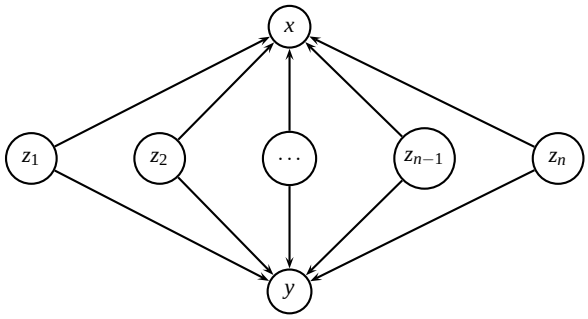


Fig. 15 A n-comb network

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 $k < n$ (recall that 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* [44] 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 16, the minimal number of nodes allowed in Z is always 0, and the maximal number is increasing on the first two third of the generations and is kept constant to 6 on the last ones. The average number of nodes in the current population is also slowly rising up but remains rather small since in this example, there are a lot of small *easy to find* independence statements and only a single big one.

The correct structure (Figure 17) is found after 40 (out of 50) generations.

The Figure 18 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.

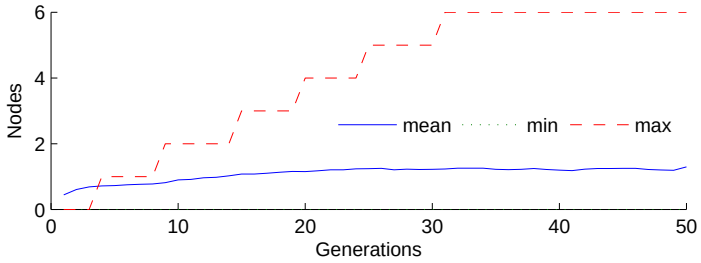


Fig. 16 Evolution of Minimal, Maximal and Average number of nodes in Z along generations

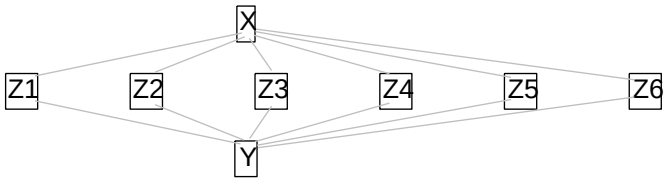


Fig. 17 Final evolved structure for the comb network

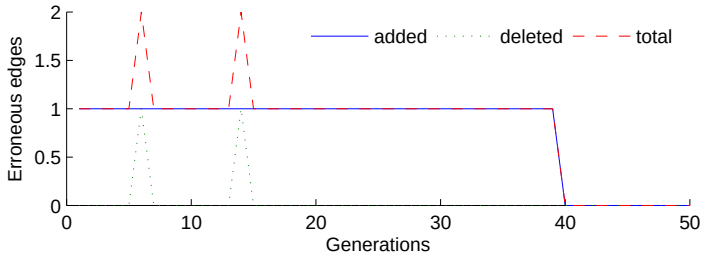


Fig. 18 Evolution of the number of erroneous edges of the structure along generations

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 4 and 5):

  - How many edges were found? Among these, how many were erroneous? (added or deleted)
  - What is the percentage of runs in which the $x - y$ edge is removed?
- PC is tuned with a fan-in k limited to 3 (a larger fan-in is not used as PC is performing a full combinatorial research) and P-IMPEA is tuned with 50 generation of 50 individuals in order to take the same computational time as PC. 50 generation are more than enough to converge to a solution due to the small size of the problem. Both algorithms share the same significance level α.

The actual network contains 8 edges and 6 nodes. Therefore the number of possible 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 column of Table 4) 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.

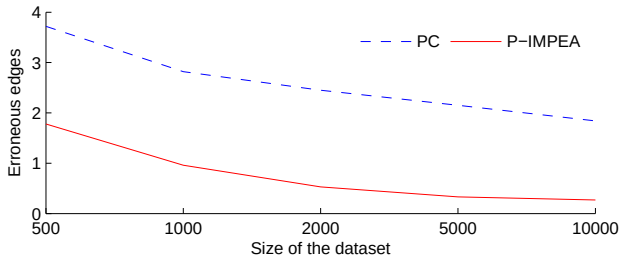


Fig. 19 Number of erroneous edges (added+deleted) for PC and P-IMPEA, depending on the size of the dataset

Regarding the global performance, the Figure 19 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 4 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

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 5 Averaged results of P-IMPEA algorithm after 100 runs

P-IMPEA (Table 5) whereas it is greater than 9 for the PC algorithm since it can't remove the $x - y$ edge (Table 4).

4.8.2 Classical benchmark: the Insurance Bayesian network

Insurance is a network for evaluating car insurance risks developped by [7]. The Insurance Bayesian network contains 27 variables and 52 arcs. It is a large instance. A database of 50000 cases generated from the network has been used for the experiments below.

Once again, we start from a population with small Z and let it increase up to 4 nodes. The Figure 20 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.

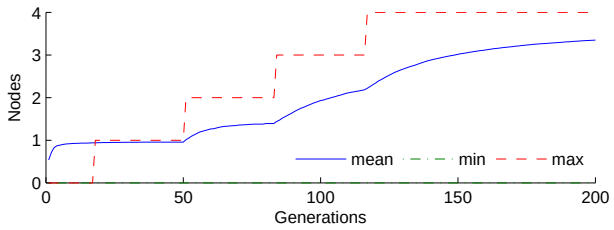


Fig. 20 Evolution of Minimal, Maximal and Average number of nodes in Z along generations

Concerning the evolution of the number of erroneous edges represented on Figure 21, 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 6). When asking for 52 edges, the actual number of edges in the Insurance network, it makes 14 errors (7 additions and 7 deletions).

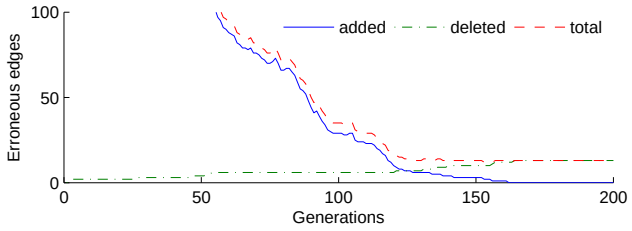


Fig. 21 Evolution of the number of erroneous edges of the structure along generations

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 6 Number of detected edges for all algorithms

4.8.3 Real Dataset: Cheese ripening data from the INCALIN project

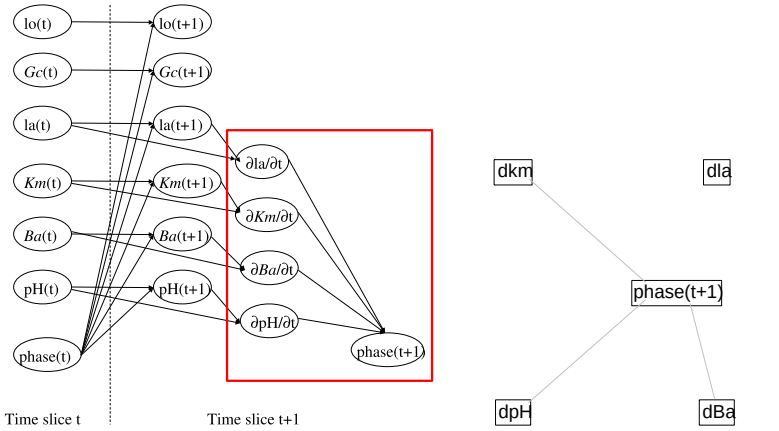
The last step is to test our algorithm on real data. Our aim is to compare the result of IMPEA with a part of the dynamic Baysian network, already described at section 3, built with human expertise in the scope of the INCALIN project. We are interested in the part of the network that predicts the current phase knowing the derivatives of some bacteria proportions. We used the same data as in the first part of the report (see section 3.2.6), 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 expert. As one can see on the right of Figure 22, no extra edge is added, but one edge is missing, between the derivative of la and the phase.

4.9 Analysis

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

The two experiments of section 4.8 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



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

(b) Results of P-IMPEA.

Fig. 22 Comparison between the model proposed by experts and the network found by IMPEA on a real dataset from the INCALIN project.

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.

5 Conclusion

Parisian CCEAs and cooperative co-evolution in general, when applicable, yield efficient and robust algorithms. As we have seen in this chapter, the main concern is the design of adequate representations for cooperative co-evolution schemes, in other words, representations that allow a collective evolution mechanism. One has to build an evolution mechanism that uses pieces of solutions instead of complete solutions as individual. It is also needed to evaluate the pieces of solutions (local fitness) before being able to select the best pieces that can be considered as components of a global solution.

In the example of section 3, we first designed a classical GP, where the phase estimator was searched as a single best "monolithic" function. Although it already outperforms the previous other methods, we obtained additional improvements by splitting the phase estimation into four combined (and simpler) "phase detectors". We actually used additional a priori informations about the problem. The structures evolved here were binary output functions that characterised one of the four phases. Their aggregation was made via a robust voting scheme. The resulting phase detector has almost the same recognition rate as the classical GP but with a lower variance, evolves simpler structure during less generations, and yield results that are easier to interpret.

In section 4, the cooperative coevolution algorithm IMPEA has allowed overcoming a known drawback of the classical approach, that is to find an efficient representation of a direct acyclic graph. We have shown that the cooperative scheme is particularly adapted to an alternate representation of Bayesian Networks: Independence Models (IM). IM represent data dependencies via a set of Independence Statements (IS) and IS can directly be considered as individuals of a CCEA.

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

Table 7 gives an overview of the features of the two CCEAs schemes presented in this chapter. There are some major differences between the two approaches.

- With respect to the nature of the cooperation within the population: the Parisian phase prediction is relying on components that are structured in 4 clusters (each individual only votes for the phase it characterises the best), while IMPEA collects the best individuals of its population in an archive to build the global independence model.

	Parisian Phase Prediction	**IMPEA**
Individuals	Phase predictors	IS : Independence Statements $\langle x,y\|Z\rangle$
Population/groups	Classifier	IM : Independence Model
Nb of cooperating components	4	variable
Aggregation	clustering + selection of the 5% best	at the end of the evolution only
Local fitness	capability to characterise a phase $\max\{F_1,F_2,F_3,F_4\}$ $\times$ pressure toward simple structure	adjusted *p-value* $\times$ pressure toward small conditional parts
Global fitness	voting scheme + evaluation on the learning set	none
Sharing	Euclidean distance on $\{F_1,F_2,F_3,F_4\}$	Hamming distance on $\{x,y\}$
Specific features	variable population size inflation / deflation	archive embossing points

Table 7 Features of the two Parisian schemes

- With respect to the synchonisation of the global fitness calculation: the Parisian phase prediction computes a global fitness at each generation and use a bonus distribution mechanism, while IMPEA only relies on local calculations at each generation. The global calculation is made only once at the end of the evolution.

It is interesting to note that IMPEA may be considered as an incomplete Parisian scheme, as it does not use any global calculation. Future work on this algorithm will be aimed at evaluating if a global calculation may accelerate its convergence and robustness. Note however that for instance the fly algorithm [63, 41] does not use any global fitness either, but is able to provide extremely rapid results: the cooperation mechanisms may operate in some cases without global fitness.

The common characteristics of these two examples is that the cooperative scheme has allowed representing in an indirect way some complex structures (classification rules in the first example and Bayesian Networks in the second one). This way of exploiting the artificial evolution scheme is versatile enough to facilitate the integration of constraints and the development of various strategies (archive and embossing points as in section 4, or variable population size as stated in [5] for instance). The experiments described in this chapter join previous studies on “Parisian evolution”, that experimentally proved that very efficient algorithms can be built on this cooperation-coevolution basis, in terms of rapidity [41], or in terms of size and complexity of the problems [17, 63].

Acknowledgements

This work has been funded by the French ANR (National Research Agency), via a PNRA fund (Agri-food research program, 2007-2009).

References

1. Aldarf, M., Fourcade, F., Amrane, A., Prigent, Y.: Substrate and metabolite diffusion within model medium for soft cheese in relation to growth of penicillium camembertii. J. Ind. Microbiol. Biotechnol. **33**, 685–692 (2006)
2. Arfi, K., Amrita, F., Spinnler, H., Bonnarme, P.: Catabolism of volatile sulfur compounds precursors by brevibacterium linens and geotrichum candidum, two microorganisms of the cheese ecosystem. J. Biotechnol. **105**(3), 245–253 (2003)
3. A.Tucker, Liu., X.: Extending evolutionary programming methods to the learning of dynamic bayesian networks. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 99). Morgan Kaufman, Orlando, Florida, USA (1999)
4. Barile, D., Coisson, J., Arlorio, M., Rinaldi, M.: Identification of production area of ossolano italian cheese with chemometric complex aproach. Food Control **17**(3), 197–206 (2006)
5. Barrière, O., Lutton, E.: Experimental analysis of a variable size mono-population cooperative-coevolution strategy. In: Proceedings of International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2008). Puerto de La Cruz, Tenerife (2008)
6. Baudrit, C., Wuillemin, P.H., Sicard, M., Perrot, N.: A dynamic bayesian network to represent a ripening process of a soft mould cheese. In: 12th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, *Lecture Notes in Computer Science*, vol. 2, pp. 265–272. Zagreb, Croatia (2008)
7. Binder, J., Koller, D., Russell, S., Kanazawa, K.: Adaptive probabilistic networks with hidden variables. Machine Learning **29**, 213–244 (1997)
8. Bongard, J., Lipson, H.: Active coevolutionary learning od deterministic finite automata. Journal of Machine Learning Research **6**, 1651–1678 (2005)
9. Boutrou, R., Guguen, M.: Interests in geotrichum candidum for cheese technology. Int. J. Food Microbiol. **102**, 1–20 (2005)
10. Bucci, A., Pollack, J.B.: On identifying global optima in cooperative coevolution. In: GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 539–544. ACM, New York, NY, USA (2005)
11. Cheng, J., Bell, D.A., Liu, W.: Learning belief networks from data: An information theory based approach. In: Proceedings of Sixth ACM International Conference on Information and Knowledge Management, pp. 325–331. Las Vegas, Nevada, USA (1997)
12. Chickering, D., Boutilier, C.: Learning equivalence classes of bayesian-network structures. Journal of Machine Learning Research pp. 150–157 (1996)
13. Chickering, D., Heckerman, D., Meek, C.: Large-sample learning of bayesian networks is np-hard. Journal of Machine Learning Research **5**, 1287–1330 (2004)
14. Choisy, C., Desmazeaud, M., Gripon, J., Lamberet, G., Lenoir, J.: Le fromage, chap. La biochimie de l'affinage, pp. 86–105. Lavoisier, Paris (1997)
15. Choisy, C., Desmazeaud, M., Gueguen, M., Lenoir, J., Schmidt, J., Tourneur, C.: Le fromage, chap. Les phénomènes microbiens, pp. 86–105. Lavoisier, Paris (1997)
16. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory **14**(3), 462–467 (1968)
17. Collet, P., Lutton, E., Raynal, F., Schoenauer, M.: Polar ifs + parisian genetic programming = efficient ifs inverse problem solving. Genetic Programming and Evolvable Machines Journal **1**(4), 339–361 (2000)
18. Cooper, G., Herskovits, E.: A bayesian method for the induction of probabilistic networks from data. Machine Learning **9**(4), 309–347 (1992)
19. Davis, L.: Adapting operators probabilities in genetic algorithms. In: Proceedings of the third Conference on Genetic Algorithms, pp. 61–69. Morgan-Kaufmann (1989)
20. Deb, K., Goldberg, D.: An investigation of niche and species formation in genetic function optimization. In: Proceedings of the third Conference on Genetic Algorithms, pp. 42–50 (1989)

21. Ellis, D., Broadhurst, D., Goodacre, R.: Rapid and quantitative detection of the microbial spoilage of beef by fourier transform infrared spectroscopy and machine learning. Analytica Chimica Acta **514**(2), 193–201 (2004)
22. Eriksson, R., Olsson, B.: Cooperative coevolution in inventory control optimisation. In: Proceedings of the Third International Conference on Artificial Neural Networks and Genetic Algorithms. Springer-Verlag, East Lansing, MI, USA (1997)
23. Francois, O., Leray., P.: Etude comparative d'algorithmes d'apprentissage de structure dans les réseaux bayésiens. Tech. rep., Rencontres des Jeunes Chercheurs en IA (2003)
24. Friedman, N.: Learning belief networks in the presence of missing values and hidden variables. In: Proceedings of 14th International Conference on Machine Learning, pp. 125–133. Morgan Kaufmann, Nashville, Tenessee, USA (1997)
25. Friedman, N., Linial, M., Nachman, I., Pe'er., D.: Using bayesian network to analyze expression data. J. Computational Biology **7**, 601–620 (2000)
26. Goldberg, D., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of Second International Conference on Genetic Algorithms and their application, pp. 41–49. Lawrence Erlbaum Associates, Inc., Cambridge, MA, USA (1987)
27. Gripon, A.: Cheese: Chemistry, Physics and Microbiology, chap. Mould-ripened cheeses, pp. 111–136. Chapman and Hall, London, United Kingdom (1993)
28. Holland, J., Reitman, J.: Cognitive systems based on adaptive algorithms. SIGART Bull. **63**, 49–49 (1977)
29. Husbands, P.: Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. In: Selected Papers from AISB Workshop on Evolutionary Computing, pp. 150–165. Springer-Verlag, London, UK (1994)
30. Husbands, P., Mill, F.: Simulated co-evolution as the mechanism for emergent planning and scheduling,. In: Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 264–270. Morgan Kaufman, San Diego, CA, USA (1991)
31. Ioannou, I., Mauris, G., Trystram, G., Perrot, N.: Back-propagation of imprecision in a cheese ripening fuzzy model based on human sensory evaluations. Fuzzy Sets And Systems **157**, 1179–1187 (2006)
32. Ioannou, I., Perrot, N., Curt, C., Mauris, G., Trystram, G.: 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 **64**, 497–506 (2004)
33. Ioannou, I., Perrot, N., Mauris, G., Trystram, G.: 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. Journal Of Food Engineering **64**, 507–514 (2004)
34. Jia, H., Liu, D., Yu, P.: Learning dynamic bayesian network with immune evolutionary algorithm. In: Proceedings of Fourth International Conference on Machine Learning and Cybernetics. Guangzhou, China (2005)
35. Jimenez-Marquez, S., Thibault, J., Lacroix, C.: Prediction of moisture in cheese of commercial production using neural networks. Int. Dairy J. **15**, 1156–1174 (2005)
36. Jong, E.D., Stanley, K., Wiegand, R.: Introductory tutorial on coevolution. In: GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation. London, UK (2007)
37. Larranaga, P., Poza, M.: 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)
38. Leclercq-Perlat, M., Buono, F., Lambert, D., Latrille, E., Spinnler, H., Corrieu, G.: Controlled production of camembert-type cheeses. part i: Microbiological and physicochemical evolutions. J. Dairy Res. **71**, 346–354 (2004)
39. Leclercq-Perlat, M., Picque, D., Riahi, H., Corrieu, G.: Microbiological and biochemical aspects of camembert-type cheeses depend on atmospheric composition in the ripening chamber. J. Dairy Sci. **89**, 3260–3273 (2006)

40. Lenoir, J.: The surface flora and its role in the ripening of cheese. Int. Dairy Fed Bull. **171**, 3–20 (1984)
41. Louchet, J., Guyon, M., Lesot, M., Boumaza, A.: Dynamic flies: a new pattern recognition tool applied to stereo sequence processing. Pattern Recognition Letters **23**, 335–345 (2002)
42. Lutton, E., Martinez, P.: 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, pp. 287–303. Lille, France (1995)
43. Moriarty, D., Miikkulainen, R.: Forming neural networks through efficient and adaptive coevolution. Evolutionary Computation **5**, 373–399 (1998)
44. Murphy, K.: The bayes net toolbox for matlab. Computing Science and Statistics **33**(2), 1024–1034 (2001)
45. Myers, J., Laskey, K., DeJong, K.: Learning bayesian networks from incomplete data using evolutionary algorithms. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 99), pp. 458–465. 1, Morgan Kaufmann, Orlando, Florida, USA (1999)
46. Ni, H., Gunasekaran, S.: Food quality prediction with neural networks. Food Technology **52**, 60–65 (1998)
47. Ochoa, G., Lutton, E., Burke, E.: Cooperative royal road functions. In: Evolution Artificielle, pp. 29–31. Tours, France (2007)
48. Panait, L., Luke, S., Harrison, J.: Archive-based cooperative coevolutionary algorithms. In: GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation. Seattle, Washington, USA (2006)
49. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufman (1988)
50. Pearl, J., Verma, T.: A theory of inferred causation. In: Proceedings of Second International Conference on the Principles of Knowledge Representation and Reasoning. Cambridge, MA, USA (1991)
51. Pinaud, B., Baudrit, C., Sicard, M., Wuillemin, P.H., Perrot, N.: Validation et enrichissement interactifs d'un apprentissage automatique des paramètres d'un réseau bayésien dynamique appliqué aux procédés alimentaires. In: Journées Francophone sur les Réseaux Bayésiens. Lyon, France (2008)
52. Popovici, E., Jong, K.D.: The effects of interaction frequency on the optimization performance of cooperative coevolution. In: Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2006. Seattle, Washington, USA (2006)
53. Potter, M., Jong, K.D.: A cooperative coevolutionary approach to function optimization. In: Parallel Problem Solving from Nature (PPSN III), Lecture Notes in Computer Science 866, pp. 249–257. Springer, Jerusalem, Israel (1994)
54. Potter, M., Jong, K.D.: Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evolutionary Computation **8**(1), 1–29 (2000)
55. Potter, M., Jong, K.D.: The coevolution of antibodies for concept learning. In: Parallel Problem Solving from Nature (PPSN V), Lecture Notes in Computer Science 1498, pp. 530–539. Springer, Amsterdam, The Netherlands (2008)
56. Riahi, M., Trelea, I., Leclercq-Perlat, M., Picque, D., Corrieu, G.: 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 (2000)
57. Robinson, R.: Counting unlabeled acyclic digraphs. In: Combinatorial Mathematics V: Proceedings of the Fifth Australian Conference, pp. 28–43. Springer, Melbourne, Australia (2000)
58. Ross, J., Zuviria, E.: Evolving dynamic bayesian networks with multi-objective genetic algorithms. Applied Intelligence **26**(1), 13–23 (2007)
59. Silva, S.: GPLAB A Genetic Programming Toolbox for MATLAB, `http://gplab.sourceforge.net/` (2008)
60. Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search, second edn. The MIT Press (2001)
61. Tarantilis, C., Kiranoudis., C.: Operational research and food logistics. Journal of Food Engineering **70**(3), 253–255 (2005)

62. Tonda, A., Lutton, E., Squillero, G.: Lamps : A test problem for cooperative coevolution. In: NICSO 2011, the 5th International Workshop on Nature Inspired Cooperative Strategies for Optimization, October 20-22, Cluj Napoca, Romania (2011)
63. Vidal, F., Lazaro-Ponthus, D., Legoupil, S., Louchet, J., Lutton, E., Rocchisani, J.M.: Artificial evolution for 3d PET reconstruction. In: Proceedings of the 9th international conference on Artificial Evolution (EA'09). Strasbourg, France (2009)
64. Wang, S., Li., S.: Learning bayesian networks by lamarckian genetic algorithm and its application to yeast cell-cycle gene network reconstruction from time-series microarray data. In: Proceedings of BioADIT 2004 : Biologically inspired approaches to advanced information technology, pp. 49–62. Lausanne, Suisse (2004)
65. Wiegand, R., Liles, W., Jong, K.D.: Analyzing cooperative coevolution with evolutionary game theory. In: Proceedings of the 2002 Congress on Evolutionary Computation CEC2002, pp. 1600–1605. Honolulu, Hawaii (2000)
66. Wiegand, R., Potter, M.: Robustness in cooperative coevolution. In: GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation. Seattle, Washington, USA (2006)
67. Wong, M., Leung, K.: An efficient data mining method for learning bayesian networks using an evolutionary algorithm-based hybrid approach. IEEE transactions on evolutionary computation **8**, 378–404 (2004)