# Probability estimation by an adapted genetic algorithm in web insurance

Anne-Lise Bedenel, Laetitia Jourdan, Christophe Biernacki

# Probability estimation by an adapted genetic algorithm in web insurance

Anne-Lise Bedenel[1,2,3], Laetitia Jourdan[2], Christophe Biernacki[3]

[1] MeilleureAssurance, France
anne-lise.bedenel@meilleureassurance.com
[2] Université Lille 1 CRIStAL, UMR 9189, France
laetitia.jourdan@univ-lille1.fr
[3] Inria, Université Lille 1, France
christophe.biernacki@{inria,math.univ-lille1}.fr

**Abstract.** In the insurance comparison domain, data constantly evolve, implying some difficulties to directly exploit them. Indeed, most of the classical learning methods require data descriptors equal to both learning and test samples. To answer business expectations, online forms where data come from are regularly modified. This constant modification of features and data descriptors makes statistical analysis more complex. A first work with statistical methods has been realized. This method relies on likelihood and models selection with the Bayesian information criterion. Unfortunately, this method is very expensive in computation time. Moreover, with this method, all models should be exhaustively compared, what is materially unattainable, so the search space is limited to a specific models family.
In this work, we propose to use a genetic algorithm (GA) specifically adapted to overcome the statistical method defaults and shows its performances on real datasets provided by the company MeilleureAssurance.com.

**Keywords:** Genetic Algorithms, BIC, insurance, WEB

## 1 Introduction

The objective of online insurance comparators is to propose to web users the offer the most adapted to their expectations, according to their profiles. Most of the online insurance comparators compare with only one criterion: the price. To improve web users comparison, the company MeilleureAssurance.com wishes to create a model allowing predicting the best offer according to web user profiles. It is a classical objective in statistics but, with the functioning of an insurance comparator, standard methods cannot be used. To do an online comparison, a web user has to fill a form of questions. When the form is filled, data are sent to insurer partners with a web service. So, they can send the real price of the offer back to the company. An insurance comparator adapts and changes regularly its forms:

– For insurers: Each insurer has his particularly pricing system, questions are not homogeneous between all insurers partner.
– For web users: For more clarity and simplicity, questions are regularly adapted.

This adaptability is a specificity of insurance comparators. Due to this specificity, building a supervised classification model with these features becomes complex.

A first work has been realized to solve this problem, using several statistical tools as the likelihood, to estimate the parameters. In this first work, the modeling realized shows many constraints and involves a problem of model selection [1]. The model selection is realized with an exhaustive search. Indeed, with the statistical process, the selection model is performed by comparing all available models with an information criterion [2]. This method shows good results for the estimation and the model selection. However, it is time-consuming and the number of models to compare has to be limited. To avoid the exhaustive search, an optimization approach is considered.

In the literature, many papers can be found where a genetic algorithm is used to do a model selection [8], [9], [10], [11] and parameters estimation [12]. In this work, we propose a new genetic algorithm to overcome defaults of statistical methods and new operators to have the best metaheuristics. Section 2 describes the statistical work with the modeling problem. Section 3 presents the algorithm used and new operators proposed. Section 4 drives experiments and compares results between statistical method and genetic algorithm on simulated and real datasets. Section 5, gives some conclusions and perspectives for future works.

## 2 Modeling of the problem

### 2.1 Probabilistic modeling

To introduce the general modeling, a use case is studied where the question is: "How web users react when data descriptors of feature change?"

To answer to this question, the feature *Coverage levels* is studied.

In a first time, this feature had four data descriptors {Third-party (T), Third-party++ (T++), third-party, fire and theft (TPFT), comprehensive (C)}. This feature denoted by $X \in \{1, \ldots, I\}$, $I$ designating the number of data descriptors ($I = 4$ in our use case). In a second time, it has been decomposed into seven data descriptors: {Third-party (T), Third-party+ (T+), Third-party++ (T++), third-party, fire and theft (TPFT), comprehensive (C), comprehensive+ (C+), comprehensive++ (C++)}. This feature is symmetrically denoted by $Y \in \{1, \ldots, J\}$, $J$ designating the new number of data descriptors ($J = 7$ in our use case).

Another specificity for an insurance comparator is that there is no data history of web users. So, the available observations on $X$ and $Y$ are never matched. More precisely, this property can be written like this:

*Period before the modification* $N^-$ Web users have filled the feature $X$, so there are *observed* realisations $\mathbf{X}^- = (X_1^-, \ldots, X_{N^-}^-)$. The feature $Y$ has never been filled, so there are *unobserved* realisations $\mathbf{Y}^- = (Y_1^-, \ldots, Y_{N^-}^-)$.

*Period after the modification* Symmetrically, $N^+$ Web users have filled the feature $Y$, so there are *observed* realisations $\mathbf{Y}^+ = (Y_1^+, \ldots, X_{N^+}^+)$. The feature $X$ has never been filled, so there are *unobserved* realisations $\mathbf{X}^+ = (X_1^+, \ldots, X_{N^+}^+)$.

### 2.2 Parameters estimation and models selection

We assume each couple $(X_n^*, Y_n^*)$ is an independent and identically distributed realization of the couple $(X, Y)$ with $n = 1, \ldots, N^*$ and $* \in \{-, +\}$. The distribution of the couple $(X, Y)$ can be written as:

$$P(X = i, Y = j) = p_{ij} p_i \tag{1}$$

where $p_{ij} = P(Y = j | X = i)$ and $p_i = P(X = i)$, with $i = 1, \ldots, I$ and $j = 1, \ldots, J$. The interest, here, is to show the transition probabilities $p_{ij}$ between data descriptors $X$ and $Y$. The objective is to estimate the whole of transition probabilities $\mathbf{p}_{..} = (p_{ij})$. It can be noted $\mathbf{p}_{.} = (p_i)$, which is also an unkown parameter.

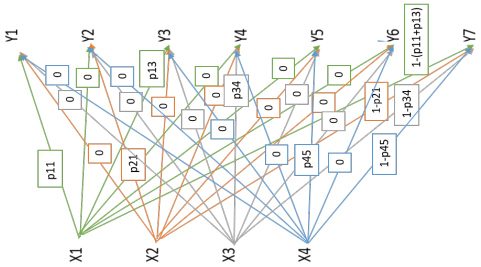



**Fig. 1.** Graph of possible matching between X- feature before modification and Y-feature after modification.

The set of transition (matching) probabilities $\mathbf{p}_{..}$ is introduced with the graph displayed on Fig 1, where the oriented edges represent estimated parameters in the use case. It can be noted that the number of parameters $\mathbf{p}_{..}$ is larger than the number of parameters of Y distribution. So, the model is statistically over-parameterized and therefore has multiple solutions whose range can be found through repeated optimization from different starting points. More precisely, in the use case, there are 28 matching probabilities (so 24 free parameters). However, there are only 6 free parameters for $Y$ distribution. So the model is said unidentifiable. To have an identifiable model, some constraints have to be imposed on $\mathbf{p}_{..}$ to limit the number of free parameters to 6 or less ($\dim(\mathbf{p}_{..}) \leq J - 1$). To respect the identifiability constraint, it has been proposed to fix some value of $\mathbf{p}_{..}$ to 0. This type of constraint will be a model noted $\mathbf{m}$. So, it leads to a set of models $\mathcal{M} = \{\mathbf{m}\}$ and they will be challenged. For one model $\mathbf{m}$, parameters $\mathbf{p}_{..}$ are estimated with log-likelihood maximization of observed data [3] defined by:

$$\ell_{\mathbf{m}}(\mathbf{p}_{..}, \mathbf{p}_{.}; \mathbf{X}^-, \mathbf{Y}^+) = \sum_{j=1}^{J} N_j^+ \ln \left( \sum_{i=1}^{I} p_{ij} p_i \right) + \sum_{i=1}^{I} N_i^- \ln p_i \tag{2}$$

where $N_i^- = \#\{X_n^- = i, n = 1, \ldots, N^-\}$ and $N_j^+ = \#\{Y_n^+ = j, n = 1, \ldots, N^+\}$. With this maximization of log-likelihood, estimators of $\mathbf{p}_{\cdot}$ and $\mathbf{p}_{\cdot\cdot}$ (respectively $\hat{\mathbf{p}}_{\cdot}$ and $\hat{\mathbf{p}}_{\cdot\cdot}$) are obtained. To choice the best model $\mathbf{m}$ in the set $\mathcal{M}$, the conditional BIC criterion (Bayesian Information Criterion) [2], given by:

$$\mathrm{BIC}_{\mathbf{m}} = -2\ell_{\mathbf{m}}(\hat{\mathbf{p}}_{\cdot\cdot}, \hat{\mathbf{p}}_{\cdot}; \mathbf{Y}^+|\mathbf{X}^-) + \nu_{\mathbf{m}} \ln N^+ \tag{3}$$

where $\nu_{\mathbf{m}} = \dim(\mathbf{m})$ is the number of free parameters for the model $\mathbf{m}$, is used. The model $\hat{\mathbf{m}}$ having the lowest value of BIC criterion is selected. This method gives good results for the estimation and model selection [1]. However, it is an exhaustive method that is time-consuming. Indeed, the exhaustive method handles two problems. The first one is to find values of estimated parameters. The second one is to select the best model. So, in the exhaustive method, there is a continuous problem (estimation of parameters) for each model and a combinatorial problem (models selection). For example, in the use case, the feature $X$ has 4 levels and $Y$ has 7 levels. Therefore, there are $\binom{6*4}{6} = 134\,596$ possible models. For each model, probability values have to be found. So there are 134 596 continuous problems where probability values have to be found. All models have to be compared to select the best one. Among these possibilities, some of them are not allowed. Indeed, each $X$ level and $Y$ level has to be joined. Consequently, the whole of possibilities is reduced. However, it is not enough to do the exhaustive method. To do it, only a specific model family is compared which is defined by:

- The number of parameters is fixed and equal to the number of $Y$ levels - 1.
- The parameters are probabilities, so for each $X_i$ levels, the sum has to be equal to one. This constraint imposes, for each $X_i$ levels, the last level of $Y$ has to be equal to $(1 - \sum_{j=1}^{J} \mathbf{p}_{ij})$. Figure 1 illustrates this. So the last level of $Y$ is fixed as no free parameters. It cannot be set to 0 and is not an estimated parameter.

Even with these constraints, the exhaustive method stays time-consuming. For example, in the use case, to do the estimation and the comparison of 4 095 models, the process needs 1h07. In a business context, it is very long. Moreover, the firm MeilleureAssurance.com could have features with more levels, so with more parameters to estimate and a larger combinatorial problem. The objective is to reduce the computation time to do the estimation and the comparison. To reach this objective, it can be interesting to perform a non-exhaustive search. The problem involves having a flexibility according to constraints and the objective is to have a high-quality solution. A stochastic method responds to these expectations. Moreover, according to the problem, a metaheuristic based on population (P-Metaheuristic), and more especially an evolutionary method, is adapted. To challenge the statistical method, we propose to use a genetic algorithm [16].

## 3 Algorithms

The use of a genetic algorithm will allow obtaining a good solution quickly. Moreover, a genetic algorithm is particularly adapted to real problems. To have a better speed of convergence, a steady-state algorithm (ssGA) [15] is also used.

### 3.1 Encoding and evaluation

A genetic algorithm is defined by a potential solution, a population, an environment (search space) and a fitness function.

*Definition of the solution* To define the solution, the probabilistic modeling is used. A model $\mathbf{m}$ is a set of transition probabilities $\mathbf{p}_{..}$ where some of them are estimated or fixed to 0. Work with probabilities involve: each estimated probability $\mathbf{p}_{..} \in [0, 1]$ and is a real number. A solution corresponds to a model. Figure 2 shows an example of representation of model $\mathbf{m}$ and Figure 3 his matrix form. In the solution, to each X level, the sum of probabilities has to be equal to 1. Figure 3 shows estimated probabilities of X for the level T which are 0.6 and 0.4.

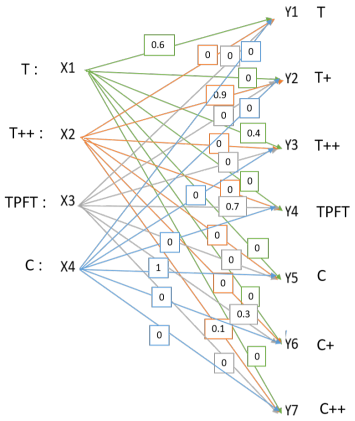



**Fig. 2.** Graph of possible matching between X- feature before modification and Y- feature after modification for a fixed model $\mathbf{m}$.

| $X \backslash Y$ | T | T+ | T++ | TPFT | C | C+ | C++ |
|---|---|---|---|---|---|---|---|
| T | 0.6 | 0 | 0.4 | 0 | 0 | 0. | 0 |
| T++ | 0 | 0.9 | 0 | 0 | 0 | 0 | 0.1 |
| TPFT | 0 | 0 | 0 | 0.7 | 0 | 0.3 | 0 |
| C | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**Fig. 3.** Matrix of transition corresponding to the model $\mathbf{m}$ between the levels X and Y.

*Population* The population is composed of available models in the search space: the set of models $\mathcal{M}$. Contrary to the statistical method, the set of models $\mathcal{M}$ does not need to be reduced.

*Fitness function* To evaluate the solution (model), the BIC criterion defined by Equation 3 is used as the fitness function which has to be minimized.

### 3.2 Operators

A genetic algorithm relies on three operators that are: selection, crossover and mutation. The choice of these operators depends on encoding parameters and for this problem a real encoding is used. The selection will be performed by a classical operator. To perform the crossover and mutation, the choice is more

complex and these operators have to be adapted to the problem. Indeed, models have many 0 and the statistical process shows that the place of all 0 is a real information. The first objective will be to design operators adapted to manage the 0 information.

**Selection operator** The selection will be performed with a classical operator: the Binary Tournament Selection [6]. This operator selects randomly two solutions of population $\mathcal{M}$. In a second time, it evaluates solutions with $\mathrm{BIC}_{\mathbf{m}}$ and selects the best solution $\hat{\mathbf{m}}$ which is the solution with lowest BIC criterion. So, we have :

$$\hat{\mathbf{m}} = \operatorname{argmin} \mathrm{BIC}_{\mathbf{m}} \tag{4}$$

**Crossover operators** Two crossover operators relying on real encoding are compared to find the most efficient:

- Uniform crossover [5]: The uniform crossover operator is very simple, according to the crossover probability, each parameter can be crossed with a probability 0.5.
- SBX crossover (Simulated Binary Crossover) [4] [13]: The SBX crossover is an operator which adapts to the evolution algorithm according to the fitness function of parents and their offspring. From two parents $p_1(i)$ and $p_2(i)$, this crossover generates two offspring $c_1(i)$ and $c_2(i)$ by the following relation:

$$\begin{cases} c_1(i) = 0.5[(1+\beta)p_1(i) + (1-\beta)p_2(i)] \\ c_2(i) = 0.5[(1-\beta)p_1(i) + (1+\beta)p_2(i)] \end{cases}$$

  where $\beta$ is a spread factor given by:

$$\beta = \begin{cases} (2u)^{\frac{1}{\eta+1}} \text{ if } u < 0.5 \\ (\frac{1}{2(1-u)})^{\frac{1}{\eta+1}} \text{ otherwise.} \end{cases}$$

  where u is random number uniformly generated on the interval [0,1] and $\eta$ a non-negative real parameter.

The constraints with parameters fixed to 0 in the probabilistic modeling involve that these operators have to be adapted. Indeed, in the problem, there are many 0. How handle them? Two solutions are proposed. In the first solution, the crossover is applied if and only if parameters of two parents are estimated else there is no crossover. In the second solution, the crossover is applied without constraints, so a 0 can become an estimated parameter and vice versa.

The example Figure 4, shows the process of the first solution. An SBX crossover is used and adapted to not to cross when the parameter is a 0. In the table showing the offspring, parameters (in bold) has been crossed. On the two first X levels (T and T++), there are no changes. On these two lines, only two parameters can be moved with a probability equals to 0.5 because these are

| $X\backslash Y$ | T | T+ | T++ | TPFT | C | C+ | C++ |
|---|---|---|---|---|---|---|---|
| T | 0 | 0 | 0 | 0 | 0.9 | 0.1 | 0 |
| T++ | 0.19 | 0.07 | 0.73 | 0 | 0 | 0 | 0 |
| TPFT | 0 | 0.15 | 0 | 0.27 | 0 | 0 | 0.57 |
| C | 0 | 0 | 0.024 | 0.097 | 0 | 0 | 0 |

Parent 1

| $X\backslash Y$ | T | T+ | T++ | TPFT | C | C+ | C++ |
|---|---|---|---|---|---|---|---|
| T | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| T++ | 0 | 0.25 | 0.4 | 0 | 0 | 0 | 0.35 |
| TPFT | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0.46 | 0.17 | 0.21 | 0.32 | 0.15 | 0 |

Parent 2



| $X\backslash Y$ | T | T+ | T++ | TPFT | C | C+ | C++ |
|---|---|---|---|---|---|---|---|
| T | 0 | 0 | 0 | 0 | 0.9 | 0.1 | 0 |
| T++ | 0.19 | 0.07 | 0.73 | 0 | 0 | 0 | 0 |
| TPFT | 0 | **0.97** | 0 | 0.27 | 0 | 0 | 0.57 |
| C | 0 | 0 | **0.018** | **0.16** | 0 | 0 | 0 |

Offsprings generated after crossover without 0 changed

**Fig. 4.** Example of crossover keeping 0

estimated probabilities for both parents (1 and 2) at the same place. On the line of X level TPFT of the parent 2, the crossover operator can be applied only on the parameter equals to 1. In the last line (the X level C), two parameters estimated have been crossed because they are in common. This example shows when one of two parents has a 0, there is no crossover.

| $X\backslash Y$ | T | T+ | T++ | TPFT | C | C+ | C++ |
|---|---|---|---|---|---|---|---|
| T | 0 | 0 | 0 | 0 | 0.9 | 0.1 | 0 |
| T++ | 0.19 | 0.07 | 0.73 | 0 | 0 | 0 | 0 |
| TPFT | 0 | **0.54** | 0 | **0.15** | 0 | 0 | **0.31** |
| C | 0 | 0 | **0.1** | **0.9** | 0 | 0 | 0 |

**Table 1.** Offsprings generated after crossover and correction operator

For each level of X, the sum of probabilities has to be equal to 1. As shown on Table 4, the crossover does not respect this constraint. Indeed, for the two parents, if the sum on each X level is performed, it will be equal to 1. But, for the offspring, the sum of probabilities of level TFPT or the level C is not equal to 1. For the level FPFT, it is equal to $0.97 + 0.27 + 0.57 = 1.8$. So to keep the sum of probabilities equals to 1 for each level of X, a correction operator (Algorithm 1) has been created.

For each X level, this operator sums probabilities. If the sum is equal to 1, it does nothing, else it adjusts the estimated parameters to have a sum equal to 1. Table 1 shows the offspring after the application of the correction operator. Parameters in bold are the parameters where a correction has been applied, so for the levels TPFT and C. After the correction, the sum of the parameters for the level TPFT is equal to $0.54 + 0.15 + 0.31 = 1$. It is the same for the level C. It can be noticed that each parameter is adjusted proportionally to his initial value. Two solutions have been compared to find the most relevant in the treatment of 0. The first one was to not cross the parameters equal to 0. On the contrary, the second one was to let the possible crossing on all the parameters.

**Algorithm 1:** Correction algorithm

```
Data: integer sup, i, j;
float difference, sum;
float value;
for Each X levels do
    sum=0;
    for Each Y levels do
        sum += Value of solution to the index(i+j*Y);
    end
    if (sum ≥ 1) then
        sup=1;
    else
        sup=0;
    end
    difference = Math.abs(sum-1);
end
for Each X levels do
    value = Value of solution to the index(i+j*Y);
    if (sup == 1) then
        set the value by (value-difference*value/sum);
    else
        set the value by (value+difference*value/sum);
    end
end
```

Figure 5 shows the process of the second solution. As for the first solution, an SBX crossover is used but for this solution, it is allowed to apply on all parameters. The bold parameters in the table representing the offspring obtained correspond to the parameters where the crossover operator has been applied. The first parameter of level T of the feature X shows the difference with the first solution. Indeed, whereas the parameter is a 0 for the parent 2, the crossover operator has been applied and so this parameter becomes a 0 for the offspring. It is the same for the other parameters. In this solution, the information on the place of the parameters equal to 0 is not kept. As for the first example, a correction operator is applied to have the sum of probabilities of each X level equals to 1.

**Mutation operator** For the mutation operator, a polynomial mutation [13], [14] is used. In this operator, a polynomial probability distribution is used to perturb a solution in a parent's vicinity. The probability distribution in both left and right of a variable value is adjusted. So that no value outside the specified range [a, b] is created by the mutation operator. In the polynomial mutation the offspring is generated as follows:

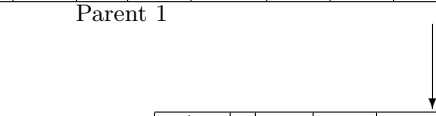
$$x_i' = x_i + (x_i^u - x_i^L)\delta_i \tag{5}$$

| $X\backslash Y$ | T | T+ | T++ | TPFT | C | C+ | C++ |
|---|---|---|---|---|---|---|---|
| T | 0.011 | 0 | 0.97 | 0 | 0.002 | 0.016 | 0 |
| T++ | 0 | 0 | 0 | 0.62 | 0.14 | 0 | 0.23 |
| TPFT | 0 | 0.67 | 0 | 0.33 | 0 | 0 | 0 |
| C | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Parent 1

| $X\backslash Y$ | T | T+ | T++ | TPFT | C | C+ | C++ |
|---|---|---|---|---|---|---|---|
| T | 0 | 0.29 | 0.48 | 0.21 | 0 | 0.02 | 0 |
| T++ | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| TPFT | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0.002 | 0 | 0.03 | 0.87 | 0 | 0.093 |

Parent 2



| $X\backslash Y$ | T | T+ | T++ | TPFT | C | C+ | C++ |
|---|---|---|---|---|---|---|---|
| T | **0** | **0.29** | 0.95 | 0.013 | 0.002 | 0.016 | 0 |
| T++ | 0 | 0 | 0 | 0.021 | 0.15 | 0 | 0.23 |
| TPFT | 0 | **0.01** | 0 | **0.016** | 0 | 0 | 0 |
| C | 0 | 0 | 0 | 1 | **0.07** | 0 | **0.1** |

Offsprings generated after crossover with 0 changed

**Fig. 5.** Example of crossover with chaging 0 allowed

where $x_i^u$ (resp. $x_i^L$) represents the upper bound (resp. lower bound) for $x_i$. The parameter $\delta_i$ is computed from the polynomial probability distribution: $p(\delta) = 0.5(\eta_m + 1)(1-|\delta|^{\eta_m})$

$$\delta_i = \begin{cases} (2r_i)^{\frac{1}{\eta_m+1}} \text{ if } r_i < 0.5 \\ 1 - (2(1-r_i))^{\frac{1}{\eta_m+1}} \text{ otherwise.} \end{cases}$$

where $\eta_m$ is the distribution index and $r_i$ is a random number in [0,1].

## 4 Experiments

### 4.1 Experimental protocol

To compare 8 genetic algorithms obtained with different crossover and mutation operators, each meta-heuristic is run on the same simulated dataset. The simulated dataset is generated with the programming language R and the function Rmultinom(see Section 4.2). Each metaheuristic is implemented in JAVA with JMETAL platform [7] on a Linux machine with processor Intel Core i5-4590 CPU 3.3GHz*4 and 4 GO of RAM. Each genetic algorithm is stopped after the same maximum number of iteration and is run 25 times. For each execution of the genetic algorithm, the best solution found is saved and stored. The average of 25 best solutions stored is calculated for each meta-heuristic. A statistical test of average comparison (Kruskal Wallis) is performed to compare the performance of these metaheuristics. This test has been chosen because the samples are small (25) and independent. If the test is significant, a left unilateral test of Mann Whitney will be performed to have the metaheuristics that have an average significantly lowest. A boxplot of each meta-heuristic is also realized. The meta-heuristics with the lowest average and the most efficiency is selected to be compared with the results of the statistical process.

To compare the performance of the best meta-heuristic selected to the statistical method, several simulated datasets have been generated. They are generated by the same process that for the comparison of the 8 metaheuristics. 2 real datasets are also used to compare the results. On each dataset, metaheuristic selected is run once and is stopped after a number of maximum iterations. The best solution is saved and the BIC value is compared to the BIC value found by the statistical process. Moreover, the place and the value of the estimated parameters is also studied to verify the simulated solution is found for both statistical method and genetic algorithm. In the simulated datasets, the place and value of the parameters are known, as the model to find is simulated. Contrary, to the real datasets where only the data in X and Y are known, and the model to find is unknown.

### 4.2 Description of instances

Experiments are computed using 10 simulated datasets and 2 real datasets. Simulated datasets are generated with the software R and the function Rmultinom. This function allows generating datasets according to their multinomial distribution. Table 2 shows an example of data representation for simulated dataset. In this example, there are 10 000 couples ($X$,$Y$). As it is a simulated model, transition probabilities and the place of 0 are known. Table 3 shows probabilities associated to Table 2.

| $X\backslash Y$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 2122 | 1260 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 961 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1975 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 3529 | 143 | 10 |

**Table 2.** Matrix of simulated datasets.

| $X\backslash Y$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 0.63 | 0.37 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0.96 | 0.038 | 0.002 |

**Table 3.** Matrix of probabilities of simulated datasets.

Table 4 details information about the different datasets used for experiments. The dataset DS3 is used for the comparison of 8 metaheuristics. In these experiments, there are two real datasets (CoverageLevels, DriverStatut) that come from the company MeilleureAssurance.com. The dataset **CoverageLevels** corresponds to the use case used for the modeling.

### 4.3 Parameters

Different parameter settings were studied before deciding which one to use for the final experiments.

All parameters have been chosen experimentally. To choose the number of maximum iteration a study of the convergence of the algorithm has been performed. The maximum iteration number and the size of the population differ according to the size of the search space. Table 5 shows parameters involved in this study.

**Table 4. Instances description**. The size of the feature X $|X|$ and Y $|Y|$ (i.e., number of observations), the number of levels of X ($\#X_i$) and Y ($\#Y_j$), the number of parameters to estimate ($\#p_{..}$) and the number of models compared with the statistical process $\#\mathcal{M}_s$

| **Name** | $\lvert X\rvert$ | $\lvert Y\rvert$ | $\#X_i$ | $\#Y_j$ | $\#p_{..}$ | $\#\mathcal{M}_s$ |
|---|---|---|---|---|---|---|
| DS1 | 15000 | 15000 | 4 | 7 | 6 | 4095 |
| DS2 | 11035 | 10035 | 4 | 7 | 6 | 4095 |
| DS3 | 10000 | 10000 | 4 | 7 | 6 | 4095 |
| DS4 | 5000 | 5000 | 4 | 7 | 6 | 4095 |
| DS5 | 15000 | 15000 | 3 | 5 | 4 | 81 |
| DS6 | 10000 | 10000 | 3 | 5 | 4 | 81 |
| DS7 | 5000 | 5000 | 3 | 5 | 4 | 81 |
| DS8 | 15000 | 15000 | 3 | 4 | 3 | 26 |
| DS9 | 7880 | 7880 | 3 | 4 | 3 | 26 |
| DS10 | 5000 | 5000 | 3 | 4 | 3 | 26 |
| CoverageLevels | 11441 | 8668 | 4 | 7 | 6 | 4095 |
| DriverStatut | 7438 | 8238 | 3 | 5 | 4 | 81 |

| **Parameters** | **Value** |
|---|---|
| Number of maximum iteration for the comparison of metaheuristics | 50 000 |
| Number of maximum iteration for datasets (DS1,DS2,DS3,DS4 and CoverageLevels) | 100 000 |
| Number of maximum iteration for the other datasets | 10 000 |
| Size of population for datasets (DS1,DS2,DS3,DS4 CoverageLevels) | 15 000 |
| Size of population for the other datasets | 500 |
| Crossover probability | 0.8 |
| Mutation probability | 0.8 |
| Crossover index distribution | 20 |
| Mutation index distribution | 20 |

**Table 5.** Genetic algorithm parameters

### 4.4 Sensitivity analysis of the operators

The 8 metaheuristics compared are described as follows:

- C1Ua: A uniform crossover, with the first solution where the crossover is allowed only on estimated parameters, the polynomial mutation is realized on all estimated parameters according to the mutation probability.
- C2Ua: A uniform crossover, with the second solution where the crossover is allowed on all parameters, the polynomial mutation is realized on all estimated parameters according to the mutation probability.
- CSBX1a: An SBX crossover, with the first solution where the crossover is allowed only on estimated parameters, the polynomial mutation is realized on all estimated parameters according to the mutation probability.
- CSBX_class_a: An SBX crossover, with the second solution where the crossover is allowed on all parameters, the polynomial mutation is realized on all estimated parameters according to the mutation probability.
- CSBX1: An SBX crossover, with the first solution where the crossover is allowed only on estimated parameters, a one-point polynomial mutation is used.

– CSBX_class: A uniform crossover, with the second solution where the crossover is allowed on all parameters, a one-point polynomial mutation is used.
– C1U: A uniform crossover, with the first solution where the crossover is allowed only on estimated parameters, a one-point polynomial mutation is used.
– C2U: A uniform crossover, with the second solution where the crossover is allowed on all parameters, a one-point polynomial mutation is used.

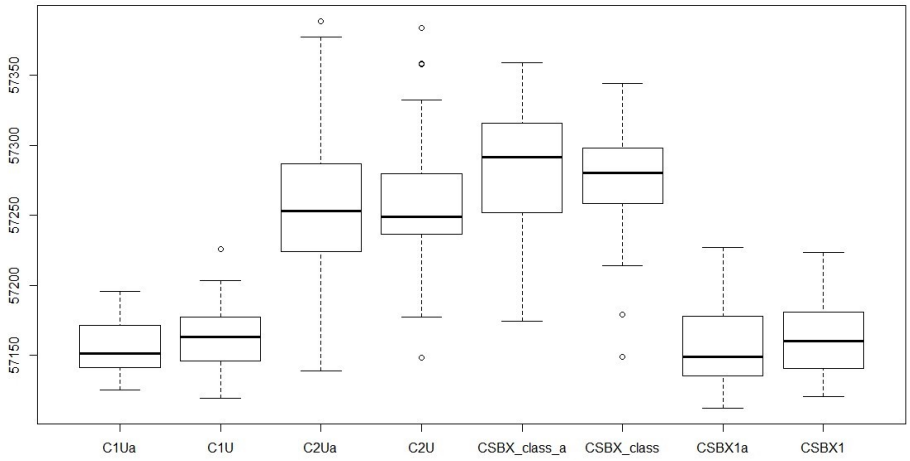



**Fig. 6.** Boxplot of 8 meta-heuristics compared.

The boxplot Fig 6 shows the 8 compared metaheuristics. The metaheuristics C2Ua, C2U, CSBX_class and CSBX_class_a correspond to the metaheuristics which use the proposition where the crossover is allowed on all parameters (0 and estimated). The boxplot shows their results seem worse than the results of the other metaheuristics. The metaheuristics which seem to have better results correspond to the metaheuristic where the crossover operator is applied only to estimated parameters. The first idea will be to use this crossover. Metaheuristics with a multi-point polynomial mutation (C1Ua, CSBX1a) seems slightly better than the results with a one-point mutation.

Many statistical tests have been performed to compare these metaheuristics and to know if the difference is significant. The first test applied is a Kruskal Wallis test. This test allows to compare the 8 metaheuristics and to know if there is really a significant difference between all these metaheuristics results. The Null hypothesis is the following:

$$\begin{cases} H0: \text{ All metaheuristic results are similar} \\ H1: \text{ There are differences between metaheuristics results.} \end{cases}$$

With a $p.value < 2.2e{-}16$ the null hypothesis is rejected. So, there is a significant difference between the metaheuristics results.

To know which metaheuristics have significantly different results, a Mann Withney test is performed. It is a test where the metaheuristics results are compared (in pairs). In a first time, a bilateral test has been applied. The assumptions

are:

$$\begin{cases} H0: \text{ Metaheuristic results (1)} = \text{metaheuristic results (2)} \\ H1: \text{ Metaheuristic results (1)} \neq \text{metaheuristic results (2).} \end{cases}$$

This test shows, metaheuristics can be grouped in 3 subsets, which are: {C1Ua, C1U, CSBX1a, CSBX1}, {C2Ua, C2U} and {CSBX_class_a, CSBX_class}. The first subset group metaheuristics that have the smallest results, left unilateral tests of Mann Withney are performed on this subset to know if one of the metaheuristics is better. However, results show that there is not a significant difference. So, this comparison shows the choice of crossover is important and we have to use the metaheuristics with the crossover applied only to the estimated parameters. For the polynomial mutation, use a one-point mutation or a multi-point mutation does not have a significant impact on the results.

### 4.5 Comparing statistical results and genetic algorithm results

We choose to use the metaheuristic CSBX1 to compare the genetic algorithm results to the statistical results. Table 6 shows the results of BIC criterion for the statistical process and the GA. The execution time of each process is also shown in Table 6. On the different simulated datasets, the GA finds a result

| Dataset | Best BIC GA | Best BIC SM | Time GA | Time SM | Average time ratio GA-SM |
|---|---|---|---|---|---|
| DS1 | 87 018.41 | 87 008.41 | 2.46min | 35min | 14.17 |
| DS2 | 62 640.8 | 62 637.1 | 2.74 min | 1h17 | **28.1** |
| DS3 | 57 105.8 | 57 101.31 | 2.5min | 1h13 | **29.2** |
| DS4 | 30 770.08 | 30 766.48 | 2.47 min | 26min | 10.52 |
| DS5 | **64 084.99** | 64 084.99 | 1.95sec | 10sec | 5.13 |
| DS6 | **49 875.41** | 49 875.41 | 2.12sec | 30 sec | **14.15** |
| DS7 | **23 604.8** | 23 612.49 | 1.95sec | 11sec | 5.64 |
| DS8 | **67 325.2** | 67 325.20 | 1.9sec | 4sec | 2.1 |
| DS9 | **32 836.38** | 32 836.38 | 1.8sec | 8sec | 4.44 |
| DS10 | **21 358.85** | 21 358.85 | 2.09sec | 3sec | 1.43 |
| Real dataset 1 | **58 287.7** | 58 306.6 | 2.10min | 33min | **15.71** |
| Real dataset 2 | **25 558.9** | 25 560.9 | 1.9sec | 30 sec | **15.78** |

**Table 6.** Result of CSBX1 on simulated and real datasets.

very close to the result of the statistical method. A test of Mann Whitney has been realized and gives a p-value = 0.9396. It means, the assumption H0 is accepted and there is no significant difference the statistical results and results of the genetic algorithm. Moreover, with the real dataset and the dataset DS7, the result of the BIC criterion found by the GA is better than the result found by the statistical method. The GA compares more models, so it can do better than the statistical method. For example, Figure 7 shows the results of parameters estimation for DS7. Because of the constraint to reduce the set of models $\mathcal{M}$ in the statistical method, the statistical method can not find the true estimation of parameters. As the GA can cover the whole set of model $\mathcal{M}$, it can find the true estimation of parameters. Concerning the execution time, for each dataset,

the GA is much smaller than the MS. Indeed, according to the dataset used, the GA is 30 times faster than the MS. Moreover, for the two real datasets, the GA is 15 times faster than the MS.

| $X\backslash Y$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0.1 | 0 | 0 | 0 | 0.9 |
| 2 | 0 | 0.3 | 0.7 | 0 | 0 |
| 3 | 0.2 | 0 | 0 | 0.8 | 0 |

Simulated datatset

| $X\backslash Y$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0.3 | 0.7 | 0 | 0 |
| 3 | 0.22 | 0 | 0 | 0.78 | 0 |

Results of SM

| $X\backslash Y$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0.11 | 0 | 0 | 0 | 0.89 |
| 2 | 0 | 0.29 | 0.71 | 0 | 0 |
| 3 | 0.199 | 0 | 0 | 0.801 | 0 |

Results of GA

**Fig. 7.** Estimation parameter results for DS7.

With the real dataset, the best model is unknown, so the set of models of the statistical method can be too small and the genetic algorithm can find more quickly a better model which is not in the set of models of the statistical method.

## 5 Conclusion and perspectives

In the insurance comparison domain, data constantly evolves. Indeed, to answer business expectations, online forms where data comes from are regularly modified. This constant modification of features and data descriptors makes analysis more complex. A first work has been realized to solve this problem, using several statistical tools as the likelihood, to estimate the parameters. In this first work, the modeling realized shows many constraints and involves a problem of model selection. In the statistical process, the model selection is an exhaustive search realized by comparing all models according to the BIC criterion. According to the feature studied, the combinatorial problem can quickly become huge and the method time-consuming. In this work, we propose to use a stochastic optimization method to overcome the statistical method defaults. According to the objective and the problem, a steady state genetic algorithm is used. In the genetic algorithm, a solution corresponds to a model which is composed of estimated parameters which are transitions probabilities and parameters fixed to 0. To handle parameters fixed to 0, a new crossover has been proposed to keep some information. It is compared with classical crossover applied on all parameters. Finally, to have the best results, 8 metaheuristics have been compared. As each solution is a probabilistic model, a new correction operator is applied after each crossover and mutation operator. This correction operator allows keeping a sum of probabilities equal to 1. The results of the comparison show that the GA with the new crossover is more efficient than a GA with the classical crossover applied to all parameters. The main objective of this work is to challenge the statistical method results. The results for BIC criterion with the GA and the SM are similar on the simulated datasets. So, with the simulated datasets, the simulated model and BIC criterion is found by both SM and GA. Moreover, as the GA compares more models, it can find a better model than the models found by the SM. It is the case with the real datasets. Concerning the execution time, the GA is much faster than the SM. On real datasets, GA is 15 times faster than SM. In this work, we show the GA can give good results for the estimations of parameters

and very quickly. These results are very interesting and encouraging. Currently, operators chosen are basics so a perspective is to add more intelligence to these operators. For example, by using statistical knowledge to create new operators and to improve results.

## References


1. A-L. Bedenel, C. Biernacki and L. Jourdan. *Appariement de donnees evoluant en temps.* Societe francaise de statistique, (2016).
2. K.P. Burnham and D.R. Anderson. *Multimodel inference: understanding AIC and BIC in model selection..* Sociological methods & research, vol. 33(2), pp. 261–304 2004.
3. G. McLachlan and K. Thriyambakam. *The EM algorithm and extensions.* Vol. 382. John Wiley & Sons, 2007.
4. R.B. Agrawal and K. Deb. *Simulated binary crossover for continuous search space.* Complex systems, vol. 9, pp. 115–148, 1995
5. C. Reeves. *Genetic algorithms.* Handbook of metaheuristics, pp. 55–82, 2003.
6. B.L. Miller and D.E. Goldberg. *Genetic algorithms, tournament selection, and the effects of noise.* Complex systems, vol. 9(3), pp. 193–212, 1995.
7. J.J. Durillo and A.J. Nebro. *jMetal a Java Framework for Multi-Objective Optimization.* Advances in Engineering Software, vol. 42, pp. 760–771, 2011.
8. K. G. Balcombe. *Model Selection Using Information Criteria and Genetic Algorithms* Computational Economics Vol. 25(3), pp. 207–228, 2005.
9. R. R. Bies, M. F. Muldoon, B. G. Pollock, S. ManuckGwenn Smith and M. E. Sale *A Genetic Algorithm-Based, Hybrid Machine Learning Approach to Model Selection* Journal of Pharmacokinetics and Pharmacodynamics , Vol. 33 (2), pp 195–221, 2006.
10. A. Blauth and I. Pigeot *Using Genetic Algorithms for Model Selection in Graphical Models* Sonderforschungsbereich 386, Paper 278, 2002.
11. S. Paterlini and T. Minerva. *Regression model selection using genetic algorithms.* 11th WSEAS, pp. 19–27, 2010.
12. L. Yao and W.A. Sethares. *"Nonlinear parameter estimation via the genetic algorithm," in IEEE Transactions on Signal Processing* vol. 42(4), pp. 927–935, 1994.
13. E.G. Talbi. *Metaheuristics: from design to implementation* (Vol. 74). John Wiley & Sons. (2009).
14. K. Deb and D. Deb. *Analysing mutation schemes for real-parameter genetic algorithms.* International Journal of Artificial Intelligence and Soft Computing, vol. 4(1), pp. 1-28, 2014.
15. F. Vavak and T.C. Fogarty. *A comparative study of steady state and generational genetic algorithms for use in nonstationary environments.* AISB workshop on Evolutionary Computing. Springer, Berlin, Heidelberg, 1996.
16. J.H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press, 1992.