Journal of Universal Computer Science, vol. 15, no. 4 (2009), 826-839
submitted: 30/6/08, accepted: 27/2/09, appeared: 28/2/09 © J.UCS

Bayesian Gene Regulatory Network Inference Optimization by means of Genetic Algorithms

Vitoantonio Bevilacqua, Giuseppe Mastronardi
(Department of Electrotechnics and Electronics, Politecnico di Bari, 70126 Italy
and
e.B.I.S. s.r.l. (electronic Business in Security), Spin-Off of Polytechnic of Bari
Str. Prov. per Casamassima Km. 3 70010 Valenzano (BA) Italy
bevilacqua@poliba.it, mastrona@poliba.it)

Filippo Menolascina, Paolo Pannarale, Giuseppe Romanazzi
(Department of Electrotechnics and Electronics, Politecnico di Bari 70126 Italy
filippo.menolascina@ieee.org, p.pannarale@gmail.com, g.romanazzi@libero.it)

Abstract: Inferring gene regulatory networks from data requires the development of algorithms devoted to structure extraction. When time-course data is available, gene interactions may be modeled by a Bayesian Network (BN). Given a structure, that models the conditional independence between genes, we can tune the parameters in a way that maximize the likelihood of the observed data. The structure that best fit the observed data reflects the real gene network's connections. Well known learning algorithms (greedy search and simulated annealing) devoted to BN structure learning have been used in literature. We enhanced the fundamental step of structure learning by means of a classical evolutionary algorithm, named GA (Genetic algorithm), to evolve a set of candidate BN structures and found the model that best fits data, without prior knowledge of such structure. In the context of genetic algorithms, we proposed various initialization and evolutionary strategies suitable for the task. We tested our choices using simulated data drawn from a gene simulator, which has been used in the literature for benchmarking [Yu et al.(2002)]. We assessed the inferred models against this reference, calculating the performance indicators used for network reconstruction. The performances of the different evolutionary algorithms have been compared against the traditional search algorithms used so far (greedy search and simulated annealing). Finally we individuated as best candidate an evolutionary approach enhanced by Crossover-Two Point and Selection Roulette Wheel for the learning of gene regulatory networks with BN. We show that this approach outperforms classical structure learning methods in elucidating the original model of the simulated dataset. Finally we tested the GA approach on a real dataset where it reach 62% of recovered connections (sensitivity) and 64% of direct connections (precision), outperforming the other algorithms.

Key Words: Genetic algorithms, Bayesian network, gene networks
Category: D.0, G.1.6, G.3, J.3

1 Introduction

In this paper we describe how computational approaches to gene regulatory networks (GRN) can be developed in order to describe complex mechanisms underlying cell behavior. Among the different regulatory mechanisms at work in the cell, transcriptional regulation plays an important role as it links a coding space of genes to a functional space of proteins. The availability of a wide range of genome wide experimental techniques, such as DNA microarrays or ChIP on chip, gives the modelers the opportunity to consider reverse engineering of transcriptional networks from experimental data. The elucidation of these networks is usually implemented by choosing a mathematical model to describe the interactions between a regulee and its regulators and then using the data to learn both the graph of interactions and the parameters of the mathematical model. We consider here the case when the graph structure is unknown and the learning task consists of discovering the nature of interactions. Several different frameworks have been proposed in order to accomplish this modeling task [de Jong(2002)] [van Someren et al.(2002)] . Among them, probabilistic graphical models appear to be a successful approach [Friedman et al.(2000)] [Segal et al.(2003)] . They offer an adequate representation of conditional (in)dependencies between variables and allow the management of uncertainty which is relevant in case of noisy data and stochastic processes. Regarding the learning issue, the choice between dynamic and static modeling depends mainly on data availability. Learning dynamic systems requires observations of temporal variations of gene expressions which are costly to produce while the availability of static data in complex organisms is growing. Among the probabilistic models, Bayesian Networks (BN) [Pearl(1988), Cowell(1999)] [Jensen(2001)] [Cowell(1999)] were selected that cover acyclic interaction networks. This class of models has often been used in the field of computational biology [Friedman et al.(2000)] in the past few years. Many approaches have been proposed to learn the structure of Bayesian networks [Imoto et al.(2002)]. Structure learning algorithms are generally based on a search within a set of candidate structures. The underlying idea is to discover the BN that best fits the available data. A scoring metric is required to assess the quality of each candidate structure with respect to the data. To undertake the search in the huge space of BN structures [Robinson(1977)], deterministic heuristics like greedy search [Chickering(2003)] or the K2 algorithm [Cooper and Herskovits(1992)] have been proposed. However, since the problem of structure learning is known to be NP-hard [Chickering(1996)], stochastic heuristics like MCMC [Friedman and Koller(2003)] [Kocka and Castelo(2001)] or Evolutionary Programming [Wong et al.(1999)] are usually preferred. They are supposed to overcome some limitations of deterministic search strategies, such as local optimality and dependence on the initial solution. In this work, we used a classical Evolutionary Algorithms (EA), namely GA, to learn network structures; as framework for our implementation we selected BANJO [Hartemink et al.(2005)] since it is one of the most used platforms in this field and it allowed establishing fair comparisons with well established alternative algorithms. In order to establish the performances of the proposed approach we used the metrics reported in [Supper et al.(2007)] (Recovered Connections, Direct Connections, Indirect Connections, Spurious Connections). We will present herein a detailed description of the approach and we will show its performances in a real world problem.

2 Background

2.1 modeling GRN with Bayesian Networks

A Bayesian network is a graph-based model of joint multivariate probability distributions that captures properties of conditional independence between variables. Such models are attractive for their ability to describe complex stochastic processes and because they provide a clear methodology for learning from (noisy) observations. Formally a Bayesian network is a representation of a joint probability distribution. This representation consists of two components. The first component, G, is a directed acyclic graph (DAG) whose vertices correspond to the random variables $X_1, \ldots, X_n$. The second component, θ describes a conditional distribution for each variable, given its parents in G. Together, these two components specify a unique distribution on $X_1, \ldots, X_n$. The graph G represents conditional independence assumptions that allow the joint distribution to be decomposed, economizing on the number of parameters. The graph G encodes the Markov Assumption.

Markov Assumption

Each variable X_i is independent of its nondescendants, given its parents in G.

By applying the chain rule of probabilities and properties of conditional independencies, any joint distribution that satisfies the Markov Assumption can be decomposed into the product form

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i, Pa^G(X_i)) \tag{1}$$

where $Pa^G(X_i)$ is the set of parents of X_i in G. A graph G specifies a product form as in equation 1. To fully specify a joint distribution, we also need to specify each of the conditional probabilities in the product form. The second part of the Bayesian network describes these conditional distributions, $P(X_i|Pa^G(X_i))$ for each variable X_i. We denote the parameters that specify these distributions by θ. In specifying these conditional distributions, we can choose from several representations. In this study, we use discrete random variables to model the gene expression levels and non-parametric modeling (e.g. Conditional Probability Tables or CPT) to represent the conditional probabilities. CPT presents at least two main advantages. They enable representation of any complex regulatory interactions without requiring fixation of the nature of the interactions before learning and they also lead to very simple maximum likelihood estimators. The parametrization of the BN relies on the coefficients of the CPT : $\{\theta^l_{ik}\}$ with k, a given state of variable X_i, and l a given configuration of its parental set Pa_i

$$PG(X_i = k|Pa_i = l) = \{\theta^l_{ik}\} \tag{2}$$

The problem of finding the BN that best approximates a set of data can be formulated as follows. To identify both structure G and parameters given a sample of size s, $D = (x_1, \ldots, x_s)$ of n random variables $X = \{X_1, \ldots, X_n\}$, we first need to define a scoring metric that evaluates how the structure and the parameters fit the data. In the case of biological networks, it is not possible to state what the true cost function is. In order to infer the model from data, we know from the learning theory that the scoring metric should incorporate a term responsible for data fitting and a term that controls the complexity of the model. The Bayesian Information Criterion (BIC) fulfils these requirements. BIC was first defined by Schwarz in 1978 as a general proposal for estimating the complexity of a statistical model. Considering G, the set of all possible DAGs containing the aforementioned n variables, the best DAG structure $\hat{G}$ can be determined by selecting in G the graph structure G that minimizes the BIC:

$$\hat{G} = argmin_G\{-2logP(D|G, \hat{\theta}) + K_G log(s)\} \tag{3}$$

where $\hat{\theta}$ is the maximum likelihood estimate of θ, the set of parameters of model G:

$$\hat{\theta} = argmax_\theta P(D|G, \theta) \tag{4}$$

and K_G the number of free parameters of model G. For the class of models that we chose and given the i.i.d. data, the likelihood can be expressed as follows:

$$P(D|G, \theta) = \prod_i \prod_k \prod_l (\theta^l_{ik})^{N^l_{ik}} \tag{5}$$

with exponent N^l_{ik} being the number of co-occurrences of both $X_i = k$ and $Pa(X_i) = l$ in the data. Therefore, the BIC can be rewritten as follows:

$$BIC(G) = \sum_i \sum_k \sum_l -2N^l_{ik} log(\hat{\theta}^l_{ik} + K^i_G log(s) \tag{6}$$

with K^i_G the number of parameters in the CPT of X_i. $\hat{\theta}^l_{ik}$ is the maximum likelihood estimate of θ^l_{ik}. The latter can be computed by $\frac{N^l_{ik}}{N^l_i}$ where $N^l_i =$

$\sum_k N_{ik}^l$ is the number of times the X_is parental configuration equals l. Since it relies on frequencies, computing these estimators from data is straightforward. This allows dedicating most of the computational time to the exploration of the structure space. Note that the BIC can be read as the sum of local scores: one local score only depends on the parental set of the node for which it is computed. To achieve BIC minimization, an appropriate search in the space of candidate graphs has to be defined. To avoid testing all of the possible graphs, searches based on appropriate heuristics are usually preferred. The Bayesian Scoring Metric, on the other hand, defines the score of each network according to equation 7:

$$Score(S) = logP(S|D) = logP(S) + logP(D|S) + c \tag{7}$$

where the firt addendum in equation 7 is the logarithm of the previous probability distribution of S, the second term is the logarithm of the probability of the likelihood that observed data have been generated by S and c is a constant term independent of S. Evidently we can also write:

$$P(D|S) = \int \dots \int \rho(D, \theta|S) d\theta = \int \dots \int P(D, \theta|S) \rho(\theta|S) d\theta \tag{8}$$

Where we can observe that the component of the probability of the model scoring can be seen as the average probability of generating the observed data on all the possible values of the parameter vector. If we consider N_{ijk} the number of occurrences of the i^{th} variable in the k^{th} state given the j^{th} parents' configuration, we can define:

$$N_{ij} = \sum_{k=1}^{r_i} N_{ijk} \qquad \alpha_{ij} = \sum_{k=1}^{r_i} \alpha_{ijk} \tag{9}$$

We can demonstrate that the Bayesian Scoring Metric can be expressed in a closed form:

$$Score(S) = logP(S) + log \left\{ \prod_{i=1}^{n} \prod_{j=1}^{r_i} \left(\frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ij} + N_{ij})}{\Gamma(\alpha_{ij})} \right) \right\} \tag{10}$$

and then:

$$Score(S) = logP(S) + \sum_{i=1}^{n} \sum_{j=1}^{r_i} \left\{ log \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \cdot \sum_{k=1}^{r_i} log \frac{\Gamma(\alpha_{ij} + N_{ij})}{\Gamma(\alpha_{ij})} \right\} \tag{11}$$

In this work, we use Evolutionary Algorithms to explore the solution space. Various implementations of EA are described as well as their qualitative and quantitative performances assessed on bio-realistic data.

2.2 Network Structure Search Algorithms

To identify BN structures with high scores, search methods are employed that search for the highest scoring graph among a set of graphs using different heuristic methods. The reason for heuristic search methods is that identifying the highest scoring network using scoring metrics is NP complete. As such, heuristic searches are iterative and thus can be run indefinitely and stopped at any time to reveal the highest scoring graph visited thus far. The longer the search, the likely of finding a higher-scoring graph. A suitable cutoff for running time is found empirically, where longer running times do not result in significant changes to the highest scoring graph found. In our study, we tested three heuristic search methods:

- greedy search with random restarts
- simulated annealing
- genetic algorithm

For each type of search we used E to denote the set of eligible changes to a graph and $\Delta(e)$ to denote the change in score of a graph resulting from the modification $e \in E$.

2.2.1 Greedy Algorithms

Greedy search with random restarts initializes itself by choosing a random graph, then evaluates the change in score $\Delta(e)$ associated with every possible change $e \in E$, and finally selects the change for which $\Delta(e)$ is maximized, provided the maximal $\Delta(e)$ is positive. It proceeds in this fashion until all $\Delta(e)$ are negative and no score improvement can be made. To escape this local maximum, the algorithm then restarts from another random graph, and the entire process is repeated until the total number of iterations is reached.

2.2.2 Simulated Annealing

Simulated annealing also initializes itself by choosing a random graph, but is given an initial temperature 0 T, a search parameter. An eligible change $e \in E$ is selected at random and the probability expression $p = e^{\frac{\Delta(e)}{T_0}}$ is evaluated. If $p > 1$ (which occurs whenever $\Delta(e)$ is positive), then the change e is made; otherwise, the change e is only made with probability p . The procedure begins at a very high temperature so that almost every eligible change in the graph can be made. As the search progresses, the temperature gradually decreases until a very low temperature is reached where very little change is made in the graph. The search then performs similarly to the local searches of the random greedy method.

3 Materials and Methods

3.1 Genetic Algorithms

A genetic algorithm (GA) [Holland(1992)] is a search method using three operators to explore a space of solutions or, in our case, a set of graphs. The three operators are: *reproduction*, which promotes the best graphs to the next generation, *mutation*, which explores new graphs by introducing variation in the population to avoid local optima, and crossover, which selects a swapping point in the parents and exchanges information between them to generate two new graphs, thereby increasing the average quality of a population. Graph structures are specified as the set of parents for every node, where graph i is donated as $\{Pa_i(X_1), Pa_i(X_2), \ldots, Pa_i(X_n)\}$ and graph j as $\{Pa_j(X_1), Pa_j(X_2), \ldots, Pa_j(X_n)\}$. To crossover, a randomly chosen variable X_k becomes the swap point leading to two new structures, graph i' and j'

$$\{Pa_i(X_1), \ldots, Pa_i(X_k), Pa_j(X_{k+1}), \ldots, Pa_j(X_n)\}$$

$$\{Pa_j(X_1), \ldots, Pa_j(X_k), Pa_i(X_{k+1}), \ldots, Pa_i(X_n)\}$$

For each GA iteration, a mutation and/or a crossover operation is chosen at random and the newly created graphs are reproduced in the next generation if they have higher scores than the current graphs in the stored population. As it is possible for crossovers to create bi-directional edges, we check for and eliminate such graphs. As selection strategy we selected a simple elitist approach. Since the probability of losing the best chromosome is not null elitistic approach saves the best solution(s) in the next population. On the other hand, however, elitism is known to bring premature convergence resulting in far from optimal solutions. The reason for this behavior can be found in the fact that despite its ability to quickly converge, elitism forces individuals to lose peculiar traits that can result to be winning in the evolutionary competition but that will be lost forever as they get out of the population. For this reason genes coming from top ranked individuals will rapidly dominate and spread among the population. In order to avoid this mechanism a choice has been made to let the GA evolve naturally for a variable number of generations till the elitism is activated by default: this is thought to let the algorithm explore the search space for some time and to speed up convergence after some sub-optimal solutions have been found.

In order to find the best performing genetic algorithm for this application we realized several versions. They are different in initial population and recombination operators. Two alternatives are eligible for the selection operator: elitist and roulette wheel. The selection operator is responsible for the selection of the solution individuals that will compose the next generation. The elitist selection pick the individuals that have the best fitness in the actual population. In the roulette wheel the parents are selected with a probability proportional to the fitness value. We can imagine this as a roulette wheel in which each individual receive a wheel portion proportional to its fitness. The algorithm act as the following:

- S = sum of all the individual's fitness
- Loop:
- R = random value in the range (0,S)
- Slide the population and sum the individual fitness you encounter. If the actual sum is $s > R$, stop and return the actual individual

The second to be changed is the crossover operator. This is the principal operator that permit to the population to evolve. For this operator we chose between two alternatives: one-point and two-point crossover. In the one-point version it is extracted randomly a swapping point k in the range $1 - n$, where n is the number of nodes in the graph. Hence the chromosome portion $[k, n]$ is exchanged between two individuals. In the two-point version, two points k_1 and k_2 are chosen randomly with uniform distribution: k_1 in the range $[1, n]$, k_2 in the range $[k_1, n]$. The chromosome portion $[k_1, k_2]$ is swapped between two individuals. Finally the initial population, from which the algorithm start its evolution, have been changed. Two alternatives are possible:

- Random initial population: each individual is a random graph
- Greedy initial population: each individual was obtained by a previous greedy search.

3.2 Simulated dataset

In order to infer a GRN we need an adequate amount of time series data. The experiments that produce this information are expensive and to skip this obstacle we can use gene profile simulators [Yu et al.(2002)]. In this work we developed a simulator implemented in the Matlab environment, that is *Gene Simulatore*. This tool can, given a matrix representing the gene network, generate time series expression values. At each time step the values updating is controlled by a simple stochastic process:

$$Y_{(t+1)} = f(Y_t) = A(Y_t - T) + e \tag{12}$$

Y_t is a vector representing the gene expression level at time t. The expression range is contained between 0 and 100, and each gene value is initialized randomly with a uniform distribution in this range. The matrix A represent the relations among genes that underlie reactions regulation. The value of each element of A

represent the regulation magnitude of one gene on the other target gene. The sign represents the interaction type, that is a positive value indicates an up-regulation effect while a negative sign means down-regulation effect. T is the vector of the regulation thresholds: each value of the vector is associated to a gene, and allows asserting if the latter exert a regulatory role on the associated genes. In this work all the values are fixed to 50. If the regulatory gene has an expression value superior to the threshold, his effects on the target genes are specified in the matrix A, and are proportional to the entity of the difference. If the expression is inferior to the threshold, the effects on the target genes are opposite to that specified in A. The term 'e' models the noise effects, and corresponds to a random value with uniform distribution in the range -10 to 10. This term includes all the noise effects, in particular the intrinsic biological noise. If a gene isn't regulated by any other gene (e.g. all the values in A are zero), it will have random step values. During the simulation the data are sampled each specified number of steps. The results are exported in a txt file. For example if we collect data every 5 steps, we will assume a sampling interval of 5 units, and the output data will correspond to the vectors $(Y_0, Y_5, Y_{10}, Y_{15}, \ldots)$, in this way we can simulate different sampling intervals like in microarray experiments.

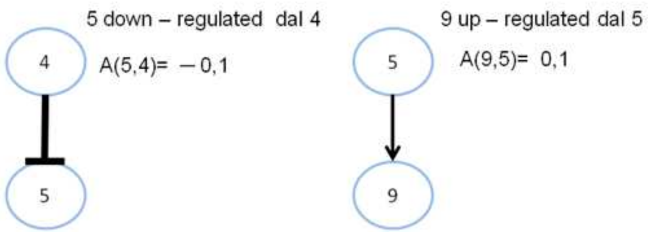

Figure 1: Gene regulation model.

The regulation matrix A elements are set in this way: if there is an up-regulation of gene y towards gene x, $A(y, x) = 0.1$; if there is a down-regulation relationship, hence $A(y, x) = -0.1$. In the case that the genes are independent $A(y, x) = 0$. In order to verify the *Gene Simulatore* behaviour we performed a 500 samples simulation and plotted the output. In the Fig. 1 we can observe that the results are consistent with the model definition: when gene 4 rises, gene 5 decrease, indeed gene 4 down-regulate gene 5. Gene 6 has a random trend, since it isn't regulated by any gene. When gene 5 is over-expressed and gene 6 is under-expressed, gene 9 is over-expressed and this is consistent with 6-down-regulation and 5-up-regulation of gene 9. Time steps are non-dimensional, but

Group	Description
Magenta	Genes expressed only in MATa cells: STE2, MFA1, MFA2, STE6, AGA2 and BAR1
Red	Genes expressed only in MAT$_{\alpha}$cells: STE3, MFALPHA1, MFALPHA2 and SAG1
Blue	Genes whose promoters are bound by STE12: FUS3, STE12, FAR1, FUS1 and AGA1
Green	Genes coding for the heterotrimeric G-protein complex: GPA1, STE4 and STE18
Yellow	Genes coding for the components of the signaling cascade (except FUS3 whitch is blue): STE7, STE11 nad STE5
Orange	Genes coding for auxiliary components of the signaling cascade: KSS1, STE20 and STE50
Brown	Genes coding for the SWI-SNF complex: SNF2 and SWI1
White	Others: SST2, KAR3, TEC1, MCM1, SIN3 and TUP1

Table 1: Figure 2 explanation

if we assume their duration to be 1 minute, they will be equivalent with a typical biological temporal scale. This model is used for all the simulations. We fixed an observation time of 10000 time steps, with a sampling interval of 5 time steps, so we obtained 2000 sampling vectors. This data, counting 2000 rows and 20 columns, corresponding to 20 genes, was used as an input for the software Banjo. After the simulated data we used real datasets. The real dataset was supplied by Hartemink, one of the Banjo developer. The database consists of 320 records with each record being characterized by 33 attributes. The records correspond to 320 samples of unsynchronized Saccharomyces cerevisiae (baker's yeast) populations observed under different experimental conditions. Yeast is considered an ideal eukaryotic organism and, thus, it has been widely studied [Hartemink et al.(2005), Supper et al.(2007)]. The first 32 attributes of each record represent the expression levels of 32 genes involved in yeast pheromone response. This pathway plays an essential role in the sexual reproduction of yeast. The last attribute of each record, named MATING TYPE, indicates the mating type of the strain of yeast in the corresponding sample, either $MATa$ or $MAT\alpha$, as some of the 32 genes measured express only in strains of a specific mating type. We note that gene expression levels are discretized into four states. We refer the reader to [Friedman et al.(2000)] for details on the data collection and preparation process, as well as for a thorough description of the 32 genes in the database. We summarize this description in Figure 2 by grouping the genes according to their function in the domain under study.

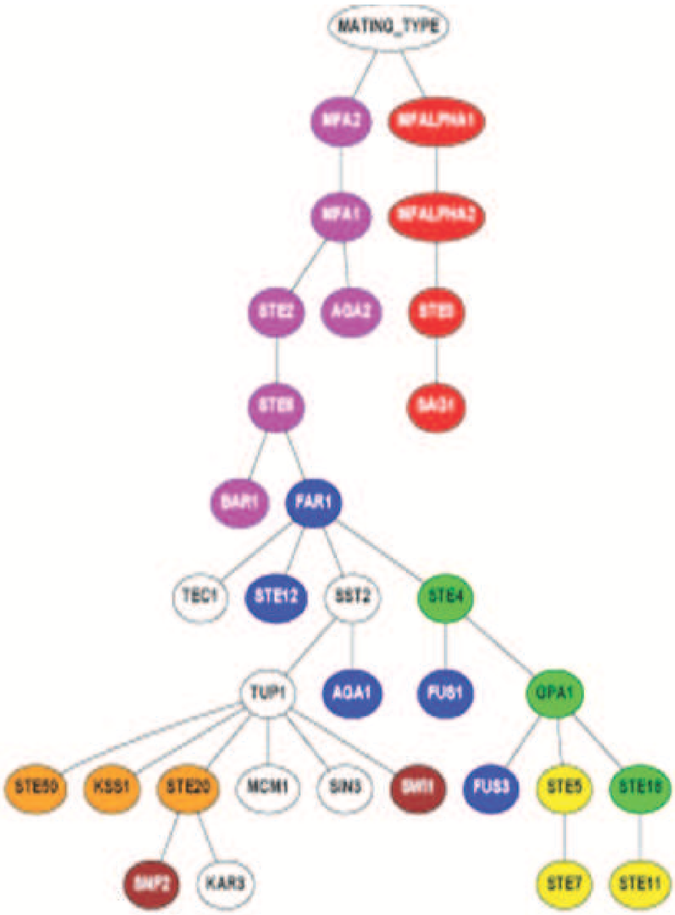

Figure 2: Best model learn.

3.3 Performance indicators

Suitable performance metrics have been used in order to compare the different gene networks inferred by the algorithm [Supper et al.(2007)]. These metrics have been calculated comparing the validation net and the best net obtained by the algorithm. The metrics are the following:

- Recovered connections: correctly identified arcs out of the total number of arcs to be identified.

- Direct connections: correctly identified arcs out of the total number of generated arcs.

Table 2: Validation of the GRN reconstruction with different methods.

Algorithms	Recovered	Direct	Indirect	Spurious
Best, 1-point, random	69.23	37.50	4.17	8.33
Best, 2-point, random	69.23	37.50	4.17	12.50
Best, 1-point, greedy	84.62	45.83	4.17	16.67
Best, 2-point, greedy	84.62	45.83	4.17	16.67
Roulette, 1-point, random	84.62	45.83	4.17	8.33
Roulette, 2-point, random	92.31	50.00	4.17	8.33
Roulette, 1-point, greedy	84.62	45.83	4.17	16.67
Roulette, 2-point, greedy	76.92	41.67	4.17	8.33
Greedy search	76.92	41.67	4.17	8.33
Simulated annealing	84.62	45.83	4.17	16.67

- Indirect connections: $a \rightarrow c$ arcs corresponding to a validation network topology of the type $a \rightarrow b \rightarrow c$ over the total number of generated arcs, e.g. direct arcs corresponding to an indirect interaction with distance two to all arcs inferred ratio.
- Sporious connections: like the previous but with distance greater than two.

We implemented a program in MATLAB for extrapolating the previous metrics given a validation net and an our net.

4 Results

The data set obtained within the Gene Simulatore has been used to compare the different approaches we implemented and the others yet used in literature. Results are reported in the table 2.

Comparing the indices presented in table 2 we see that the best configuration in this tests resulted to be the one using Roulette wheel operator together with the 2point crossover and random initial population. In order to gain a deeper understanding on how this data would overlap real data sets results, we carried out the same study on the yeast dataset kindly provided by Alexander Hartemink. In table 3 we show that our approach don't reach 100% performances, as long as there's noise in the data, but anyway outperforms classical structure learning methods in elucidating the original model.

5 Conclusions and Further Research

We presented herein an alternative approach to bayesian network inference based on a modified genetic algorithm. We showed that the best results can be obtained

Table 3: Validation of the GRN reconstruction with different methods.

	Greedy Search	Simulated Annealing	Genetic Algorithm
Recovered Connections	44.12	55.88	61.76
Direct Connections	41.67	57.58	63.64
Indirect Connections	8.33	12.12	0
Spuorious Connections	2.78	3.03	0

using GAs and in particular modifying their standard architecture to develop strategies that ease both convergence and optimal solution finding. GA in our study outperforms the classical methods adopted so far in all the performance metrics. Evolutionary learning is a promising framework for the inference of gene regulation networks. An interesting extension may be expected through the study of a more elaborated version of crowding methods, making use of a similarity metric between network structures, such as kernels on graphs. Another perspective of this work is to apply this approach to learn the structure and parameters of dynamical bayesian network. Indeed, as soon as a discrete representation of the dynamic model and a fitness function are available, EA can be applied as we proposed for the static BN. Gene Regulatory Network inference is an active area of research in current computational and systems biology and we proved that the contribution that evolutionary algorithms can give to this research still needs to be explored. In particular novel strategies need to be defined in order to optimize the employment of these techniques in the development of current biochemical model refinements. One of these cues of research can be found in the development of a machine learning approach to the estimation of the optimal crossover probability.

Acknowledgements

We thank Alexander J. Hartemink for providing the data used in this work.

References

[Chickering(1996)] Chickering, D.: "Learning Bayesian networks is NP-Complete"; Learning from Data: Artificial Intelligence and Statistics V; (1996), 121–130.

[Chickering(2003)] Chickering, D.: "Optimal structure identification with greedy search"; The Journal of Machine Learning Research; 3 (2003), 507–554.

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

[Cowell(1999)] Cowell, R.: Probabilistic Networks and Expert Systems; Springer, 1999.

[de Jong(2002)] de Jong, H.: "Modeling and Simulation of Genetic Regulatory Systems: A Literature Review"; Journal of Computational Biology; 9 (2002), 1, 67–103.

[Friedman and Koller(2003)] Friedman, N., Koller, D.: "Being Bayesian About Network Structure. A Bayesian Approach to Structure Discovery in Bayesian Networks"; Machine Learning; 50 (2003), 1, 95–125.

[Friedman et al.(2000)] Friedman, N., Linial, M., Nachman, I., Pe'er, D.: "Using Bayesian Networks to Analyze Expression Data"; Journal of Computational Biology; 7 (2000), 3-4, 601–620.

[Hartemink et al.(2005)] Hartemink, A., et al.: "Banjo (Bayesian Network Inference with Java Objects)"; web site: http://www. cs. duke. edu/amink/software/banjo; (2005).

[Holland(1992)] Holland, J.: Adaptation in natural and artificial systems; MIT Press Cambridge, MA, USA, 1992.

[Imoto et al.(2002)] Imoto, S., Goto, T., Miyano, S.: "Estimation of genetic networks and functional structures between genes by using Bayesian networks and nonparametric regression"; Pacific Symposium on Biocomputing; volume 7; 175–186; 2002.

[Jensen(2001)] Jensen, F.: Bayesian Networks and Decision Graphs; Springer, 2001.

[Kocka and Castelo(2001)] Kocka, T., Castelo, R.: "Improved learning of Bayesian networks"; Proc. of the Conf. on Uncertainty in Artificial Intelligence; 269–276; 2001.

[Pearl(1988)] Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference; Morgan Kaufmann Publishers, 1988.

[Robinson(1977)] Robinson, R.: "Counting unlabeled acyclic digraphs"; Lecture notes in mathematics; 622 (1977).

[Segal et al.(2003)] Segal, E., Shapira, M., Regev, A., Pe er, D., Botstein, D., Koller, D., Friedman, N.: "Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data"; Nature Genetics; 34 (2003), 2, 166–176.

[Supper et al.(2007)] Supper, J., Fröhlich, H., Spieth, C., Dräger, A., Zell, A.: "Inferring Gene Regulatory Networks By Machine Learning Methods"; Proceedings of the 5th Asia-Pacific Bioinformatics Conference; Imperial College Press, 2007.

[van Someren et al.(2002)] van Someren, E., Wessels, L., Backer, E., Reinders, M.: "Genetic network modeling"; Pharmacogenomics; 3 (2002), 4, 507–525.

[Wong et al.(1999)] Wong, M., Lam, W., Leung, K.: "Using Evolutionary Programming and Minimum Description Length Principle for Data Mining of Bayesian Networks"; IEEE Transactions On Pattern Analysis And Machine Intelligence; (1999), 174–178.

[Yu et al.(2002)] Yu, J., Smith, V., Wang, P., Hartemink, A., Jarvis, E.: "Using Bayesian Network Inference Algorithms to Recover Molecular Genetic Regulatory Networks"; 3rd International Conference on Systems Biology; 2002.