A Memetic Approach to Bayesian Network Structure Learning

Alberto Tonda[1], Evelyne Lutton[2],
Giovanni Squillero[3], and Pierre-Henri Wuillemin[4]

[1] INRA UMR 782 GMPA,1 Av. Brétignières, 78850, Thiverval-Grignon, France
`alberto.tonda@grignon.inra.fr`
[2] INRIA Saclay-Ile-de-France, AVIZ team, Bâtiment 650, 91405, Orsay Cedex, France
`evelyne.lutton@grignon.inria.fr`
[3] Politecnico di Torino, DAUIN, Corso Duca degli Abruzzi 124, 10129, Torino, Italy
`giovanni.squillero@polito.it`
[4] LIP6 - Département DÉSIR, 4, place Jussieu, 75005, Paris
`pierre-henri.wuillemin@lip6.fr`

Abstract. Bayesian networks are graphical statistical models that represent inference between data. For their effectiveness and versatility, they are widely adopted to represent knowledge in different domains. Several research lines address the NP-hard problem of Bayesian network structure learning starting from data: over the years, the machine learning community delivered effective heuristics, while different Evolutionary Algorithms have been devised to tackle this complex problem. This paper presents a Memetic Algorithm for Bayesian network structure learning, that combines the exploratory power of an Evolutionary Algorithm with the speed of local search. Experimental results show that the proposed approach is able to outperform state-of-the-art heuristics on two well-studied benchmarks.

Keywords: Memetic Algorithms, Evolutionary Algorithms, Local Optimization, Bayesian Networks, Model Learning

1 Introduction

Bayesian networks are probabilistic graphical models that represent a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). They are widely used to encode knowledge and perform predictions in many different fields, ranging from medicine to document classification, from computational biology to law.

It is theoretically possible to learn the optimal structure for a Bayesian network from a dataset. However, the number of possible structures is superexponential in the number of variables of the model [1] and the problem of Bayesian network learning is proved to be NP-hard [2].

The machine learning community delivered fast heuristic algorithms that build the structure of a Bayesian network on the basis of conditional independence evaluations between variables [3] [4]. On the other hand, several attempts have been made in evolutionary computation to tackle this complex issue [5] [6] [7]. Interestingly, many evolutionary approaches also feature local search techniques to improve the quality of the results.

This paper presents a memetic approach to Bayesian network structure learning. The proposed technique exploits an evolutionary framework evolving initial conditions for a state-of-the-art heuristic that efficiently explores the search space. The fitness function is based on the Akaike information criterion, a metric taking into account both the accuracy and the complexity of a candidate model.

An additional objective of this work is to link the community facing the complex Bayesian network structure learning problem, to the community of memetic computing. While combinations of heuristics and evolutionary optimization are prominently featured in the literature related to structure learning, to the authors' knowledge the methods presented are almost never ascribed to the field of memetic algorithms. In the authors' opinion, an explicit interaction between the two communities could lead to extremely beneficial results.

2 Background

In order to introduce the scope of the present work, some necessary concepts about Bayesian networks and memetic algorithms are summarized in the following.

2.1 Bayesian networks

A Bayesian Network (BN) is defined as a *graph-based model of a joint multivariate probability distribution that captures properties of conditional independence between variables* [8]. For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. The network could thus be used to compute the probabilities of the presence of various diseases, given the symptoms.

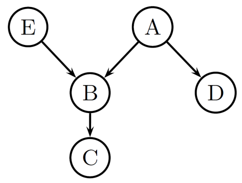
Formally, a Bayesian network is a directed acyclic graph (DAG) whose nodes represent variables, and whose arcs encode conditional dependencies between the variables. This graph is called the *structure* of the network and the nodes containing probabilistic information are called the *parameters* of the network. Figure 1 reports an example of a Bayesian network.

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

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

2.2 The structure learning problem

Learning the structure of a Bayesian network starting from a dataset is proved to be a NP-hard problem [2]. The algorithmic approaches devised to solve this



Node	Parents	Probabilities
A		$P(A=a_1) = 0.99$ $P(A=a_2) = 0.01$
B	A,E	$P(B=b_1 \| A=a_1,E=e_1) = 0.5$ $P(B=b_2 \| A=a_1,E=e_1) = 0.5$ $P(B=b_1 \| A=a_1,E=e_2) = 0.1$ $P(B=b_2 \| A=a_1,E=e_2) = 0.9$ $P(B=b_1 \| A=a_2,E=e_1) = 0.4$ $P(B=b_2 \| A=a_2,E=e_1) = 0.6$ $P(B=b_1 \| A=a_2,E=e_2) = 0.2$ $P(B=b_2 \| A=a_2,E=e_2) = 0.8$
C	B	$P(C=c_1 \| B=b_1) = 0.3$ $P(C=c_2 \| B=b_1) = 0.7$ $P(C=c_1 \| B=b_2) = 0.5$ $P(C=c_2 \| B=b_2) = 0.5$
D	A	$P(D=d_1 \| A=a_1) = 0.8$ $P(D=d_2 \| A=a_1) = 0.2$ $P(D=d_1 \| A=a_2) = 0.7$ $P(D=d_2 \| A=a_2) = 0.3$
E		$P(A=e_1) = 0.75$ $P(A=e_2) = 0.25$

Fig. 1. On the left, a directed acyclic graph. On the right, the parameters it is associated with. Together they form a Bayesian network BN whose joint probability distribution is $P(BN) = P(A)P(B|A,E)P(C|B)P(D|A)P(E)$.

problem can be divided into two main branches: score-and-search meta-heuristics and algorithms that rely upon statistical considerations on the learning set.

Evolutionary approaches Among score-and-search meta-heuristics, evolutionary algorithms are prominently featured. Several attempts to tackle the problem have been tested, ranging from evolutionary programming [6], to cooperative coevolution [5], to island models [7].

Interestingly, some of the evolutionary approaches to Bayesian network structure learning in literature already show features of memetic algorithms, hinting that injecting expert knowledge might be necessary to obtain good results on such a complex problem. For example, [6] employs a *knowledge-guided mutation* that performs a local search to find the most interesting arc to add or remove. In [9], a local search is used to select the best way to break a loop in a non-valid individual. The K2GA algorithm [10] exploits a genetic algorithm to navigate the space of possible node orderings, and then runs the greedy local optimization K2, that quickly converges on good structures starting from a given sorting of the variables in the problem.

Dependency analysis algorithms Dependency analysis algorithms are a class of heuristics that build Bayesian network structures from data through an evaluation of the conditional independence between variables. They are able to deliver results of high quality in negligible time, even if they suffer from the classical issues of greedy algorithms, such as being trapped into local optima.

One of the best algorithms in this category is known as *Greedy Thick Thinning* (GTT) [3]. Starting from a completely connected graph, first GTT applies the well-known PC algorithm [11], that cuts arcs on the basis of conditional independence tests; then, it starts first adding and then removing arcs, scoring the network after each modification and using a set of heuristic metrics to avoid a premature convergence.

Bayesian Search (BS) [4] is another state-of-the-art heuristic in the same group. Unlike GTT, BS is not deterministic: it makes use of a given number of restarts from random sparse network layouts, finally returning the best network to the user.

Both GTT and BS implementations can be found in products such as GeNie/SMILE [12].

2.3 Memetic algorithms

Memetic algorithms are *population-based metaheuristics composed of an evolutionary framework and a set of local search algorithms which are activated within the generation cycle of the external framework.* [13]. First presented in [14], they gained increasing popularity in the last years [15].

The main attractiveness of these stochastic optimization techniques lies in their ability of finding quickly high-quality results, but still maintaining the exploration potential of a classic evolutionary algorithm. Their effectiveness is proven in several real-world problems [16] [17].

3 Proposed approach

Trying to reap the benefits of both evolutionary algorithms (efficient exploration, resistance to local optima attraction) and human-devised heuristics (speed, efficient exploitation of small parts of the search space), a memetic algorithm is applied to the complex problem of Bayesian network structure learning. The algorithm evolves an initial Bayesian network that will be then optimized by a state-of-the-art dependency analysis algorithm. In this first approach, the local search heuristic is applied to every individual.

The main novelties introduced by the proposed approach are the local optimization used, the state-of-the-art GTT heuristic, and the evolutionary algorithm's individual representation, expressing a set of forced and forbidden links of arbitrary size.

The framework structure is summarized in Figure 2.

3.1 Evolutionary framework

The evolutionary core chosen for the experiments is an open-source EA [18]. Each individual represents a set of initial conditions, forbidden and forced arcs that have to appear inside the final network created by the local search. Every condition in the set follows the syntax:

```
<forbidden/forced> <starting node> <end node>
```

The genome has a minimum length of 1 condition, and no maximum length. It is interesting to notice that there is no *apriori* control on repetitions, or contradictory conditions (e.g., forbid *and* force the arc between A and B). Each group of repeated conditions is considered only once, while individuals with contradictions are discarded with a low fitness value. Finally, individuals whose condition enforce the local search to produce an invalid Bayesian network (e.g., forcing an arc from A to B and an arc from B to A would create a graph with a cycle) are discarded during fitness evaluation.

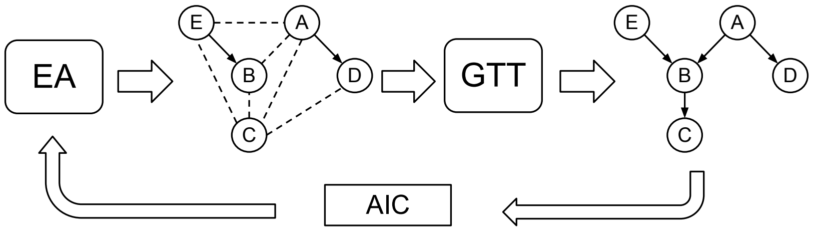


Fig. 2. Structure of the proposed approach. The evolutionary algorithm creates starting conditions where some arcs are forced (e.g. arc from E to B and from A to D) and some arcs are forbidden (e.g. arc from B to D and arc from E to D). The local search performed by GTT returns a complete structure, compliant with the conditions, manipulating freely the unconstrained arcs. The final structure is evaluated, and its AIC score is used as fitness by the evolutionary algorithm.

3.2 Fitness function

The Akaike information criterion (AIC) is a measure of the relative goodness of fit of a statistical model [19]. It is grounded in the concept of information entropy, in effect offering a relative measure of the information lost when a given model is used to describe reality. It can be said to describe the trade-off between bias and variance in model construction, or loosely speaking, between accuracy and dimension of the model. Given a data set, several candidate models may be ranked according to their AIC values: thus, AIC can be exploited as a metric for model selection.

When dealing with Bayesian networks, AIC is expressed as a composition of the loglikelihood, a measure of how well the candidate model fits the given dataset, and a penalty tied to the dimension of the model itself. The dimensional penalty is included because, on the one hand, the loglikelihood of a Bayesian network usually grows monotonically with the number of arcs, but on the other hand, an excessively complex network cannot be validated or even interpreted by a human expert. The loglikelihood of a model M given a dataset T is computed as

$$LL(M|T) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} log_2 \frac{N_{ijk}}{N_{ij}}$$

where n is the number of variables, q_i is the total number of possible configurations the parent set $pa(X_i)$ of the stochastic variable X_i, r_i is the number of different values that variable X_i can assume, N_{ijk} is the number of instances in the dataset T where the variable X_i takes its k-th value x_{ik} and the variables in $pa(X_i)$ take their j-th configuration w_{ij}, and N_{ij} is the number of instances in the dataset T where the variables in $pa(X_i)$ take their j-th configuration w_{ij}.

Taking for example the Bayesian network BN described in Figure 1, the loglikelihood of a dataset composed of one sample such as $T = (a_1, b_2, c_1, d_2, e_2)$

would be equal to

$$LL(BN|T) = log_2(P(A = a_1) \cdot P(B = b_2|A = a_1, E = e_2) \cdot \\ \cdot P(C = c_1|B = b_2) \cdot P(D = d_2|A = a_1) \cdot P(E = e_2)) = \\ = log_2(0.99 \cdot 0.9 \cdot 0.5 \cdot 0.2 \cdot 0.25) = -5.49$$

It is important to notice that datasets are usually composed by multiple samples, and that the final loglikelihood is the sum of the loglikelihoods of each sample. Using the same formulation, the dimensional penalty of model M can be expressed as

$$|M| = \sum_{i=1}^{n} (r_i - 1)q_i$$

In the canonical representation, the final AIC score is expressed as:

$$AIC = -2 \cdot (LL - |M|)$$

AIC is to be minimized.

4 Experimental setup

The effectiveness of the proposed approach is compared against GTT and BS. First, the memetic algorithm is run with a stagnation condition: if the best individual in the population remains the same for 10 generations, the algorithm stops. Then, the total number of evaluations performed is used as a reference to compare its performance against the two other approaches. BS is assigned an equal number of restarts; for the deterministic GTT, an equal number of starting random configurations are generated, following the same initialization procedure of the memetic algorithm.

In all the experimental evaluations, the algorithm has a population size μ=30, an offspring size[5] λ=30, a stagnation stop condition of 10 generations, and a set of operators that can collectively alter, remove or add a condition from an individual, and cross over two individuals, with one or two cut points. Individuals are chosen for reproduction with a tournament selection scheme. The strength and the activation probability of the genetic operators, as well as the size of the tournament selection, are self-adapted during the evolutionary run.

GTT and BS use default settings[6] with the exception of the maximum number of parents for each node, set to 10. GTT makes use of K2 as the type of priors, while BS has a probability 0.1 of an arc appearing in a random restart, a prior link probability 0.001 and a prior sample size 50.

When GTT is run as local search in the proposed memetic algorithm, it makes use of the same settings.

[5] In the chosen evolutionary framework, λ represents the number of genetic operators applied at each generation. Since some of the operators, most notably crossovers, can produce more than one child individual, the number of individuals actually generated at each step fluctuates between 30 and 60, with values often falling around 45.

[6] Default settings for the heuristics provided by the SMILE [12] framework, see `http://genie.sis.pitt.edu/wiki/SMILearn`

5 Experimental results

Two widely studied Bayesian network benchmarks are chosen for the experiments: ALARM [20], that features 37 variables and 42 arcs; and INSURANCE [21], that features 27 variables and 52 arcs. For each of the considered Bayesian network benchmarks, three datasets with 500, 1,000 and 10,000 samples respectively are created. The structure learning algorithms are executed 10 times for each dataset, each run with a given number of restarts/evaluations.

The Bayesian networks reconstructed by each algorithm are compared on three different metrics: loglikelihood, dimension and overfitting. Loglikelihood expresses the adherence of the given model to the training data. The dimension of the model is a measure of its complexity, where simpler models are preferred to more complex ones from a human perspective. Finally, the overfitting on the training data is evaluated by computing the loglikelihood of the candidate solution on a validation set of unseen data, composed of 5,000 samples.

Results are summarized in Table 1 and Table 2, with a highlight in Figures 3 and 4.

Table 1. Results for the three algorithms on the considered datasets of the Bayesian network ALARM. Results in **bold** are the best, on the basis of a two-sample Kolmogorov-Smirnov test with $p < 0.05$; when two distributions are indistinguishable but better than the third, they are highlighted in ***bold italics***.

alarm-500						
Methodology	Dimension	StDev	Loglikelihood	StDev	Overfitting	StDev
Original	509.00	-	-7,510.47	-	-75,195.1	-
(trained on dataset)	509.00	-	-7,588.84	-	-77,989.3	-
GTT (1,900 restarts)	***464.90***	11.30	-7,673.69	16.35	-79,618.9	84.41
BS (1,900 restarts)	1,298.90	87.34	-7,896.71	143.34	-85,260.0	1,781.41
Memetic Algorithm	***463.20***	28.44	**-7,629.34**	33.02	**-79,118.8**	451.57
alarm-1,000						
Original	509.00	-	-15,023.0	-	-75,195.1	-
(trained on dataset)	509.00	-	-15,045.8	-	-76,919.9	-
GTT (1,400 restarts)	564.70	19.78	-15,097.2	21.74	-77,659.3	187.28
BS (1,400 restarts)	1,546.20	149.36	-15,808.8	130.65	-83,381.4	680.57
Memetic Algorithm	**537.40**	35.80	**-15,057.7**	24.58	**-77,438.3**	236.46
alarm-10,000						
Original	509.00	-	-150,099	-	-75,195.1	-
(trained on dataset)	509.00	-	-149,993	-	-75,357.6	-
GTT (1,300 restarts)	779.00	40.57	-150,088	73.55	-75,506.8	31.79
BS (1,300 restarts)	3,369.90	553.5	-156,690	940.70	-79,550.9	447.24
Memetic Algorithm	**674.00**	53.80	**-150,026**	27.92	**-75,433.7**	26.96

6 Results discussion

The proposed approach is proved to outperform state-of-the-art heuristic techniques for the considered metrics on all the datasets, providing networks with smaller dimension, higher loglikelihood. There are, however, open research questions raised by the analyses, that is worth addressing separately.

Table 2. Results for the three algorithms on the considered datasets of the Bayesian network INSURANCE. Results in **bold** are the best, on the basis of a two-sample Kolmogorov-Smirnov test with $p < 0.05$; when two distributions are indistinguishable but better than the third, they are highlighted in ***bold italics***.

insurance-500						
Methodology	Dimension	StDev	Loglikelihood	StDev	Overfitting	StDev
Original	1,008.00	-	-9,337.06	-	-94,354.2	-
(trained on dataset)	1,008.00	-	-9,678.63	-	-101,884	-
GTT (1,700 restarts)	497.90	29.86	-9,598.68	14.67	-100,792	133.40
BS (1,700 restarts)	706.30	95.64	-9,668.27	74.27	-102,599	915.88
Memetic Algorithm	**458.60**	9.60	**-9,562.40**	9.86	**-100,278**	132.36
insurance-1,000						
Original	1,008.00	-	-19,024.7	-	-94,354.2	-
(trained on dataset)	1,008.00	-	-19,335.0	-	-98,346.4	-
GTT (1,600 restarts)	673.60	62.08	***-19,357.9***	51.16	-98,166.9	177.04
BS (1,600 restarts)	1,020.10	142.30	-19,606.9	196.80	-100,324	1,022.27
Memetic Algorithm	**574.80**	50.64	***-19,323.3***	52.60	**-97,884.2**	234.84
insurance-10,000						
Original	1,008.00	-	-187,858	-	-94,354.2	-
(trained on dataset)	1,008.00	-	-188,070	-	-95,194.7	-
GTT (2,000 restarts)	1,090.50	113.40	-188,274	96.98	-94,929.5	37.94
BS (2,000 restarts)	2,063.70	480.24	-192,121	883.16	-97,215.5	407.24
Memetic Algorithm	**882.00**	59.80	**-188,155**	57.76	**-94,834.0**	33.75

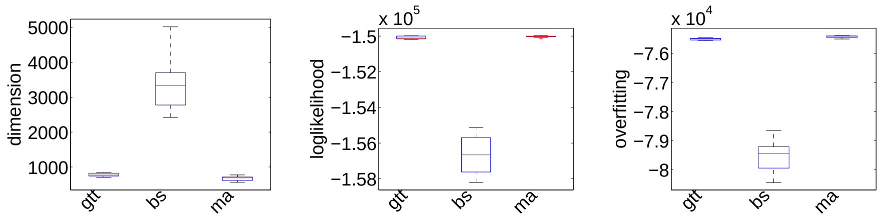


Fig. 3. Boxplots for the 10,000-sample dataset of ALARM network, for dimension, loglikelihood and overfitting.

The memetic framework presented is still relatively rough. Since the local search is applied to all individuals, the very same problem can be also expressed as finding the best set of initial conditions for the heuristic optimization algorithm. The authors see the current work as a first step, and are currently working on an extension of the framework to include other state-of-the-art optimization heuristics, such as BS, in order to give more freedom to the memetic algorithm.

The exact meaning of the initial conditions used in the framework is an interesting point of reflection. At a first glance, they might be simply considered as the point from which the heuristic will start its local search. The reality, however, is more complex: arcs forbidden and arcs forced in the initial conditions cannot be altered by the local search. This significantly changes the search space, providing the heuristic not only with a starting point, but also with a set of hard constraints that cannot be altered, and that will limit the exploration to a restricted part of the original search space. Further experiments are needed to

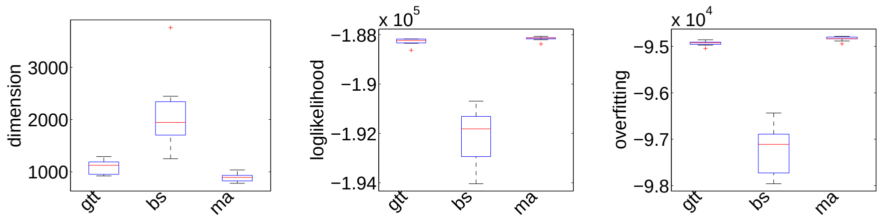


Fig. 4. Boxplots for the 10,000-sample dataset of INSURANCE network, for dimension, loglikelihood and overfitting.

fully understand the repercussions of forbidden and forced arcs on the adopted heuristics.

For a final remark on time, the heuristics with restarts and the proposed approach operate in the same order of magnitude, ranging from a few minutes for small training sets to a little more than an hour for larger ones, on the same machine.

7 Conclusions and future works

This paper proposes a memetic algorithm for Bayesian network structure learning, coupling the speed of local search with the exploration ability of an evolutionary algorithm. The algorithm evolves the initial conditions of a network, forcing and forbidding some of the arcs, and letting a local search manipulate the remaining connections from that starting point. Experimental results show that the approach is significantly more effective than a set of random restarts of state-of-the-art heuristic algorithms.

Future works will assimilate different structure learning heuristics in the memetic framework, embedding inside the genome of each individual additional information about the type and settings of the local search to apply.

References

1. Robinson, R.: Counting unlabeled acyclic digraphs. In Little, C., ed.: Combinatorial Mathematics V. Volume 622 of Lecture Notes in Mathematics. Springer Berlin / Heidelberg (1977) 28–43 10.1007/BFb0069178.
2. Chickering, D.M., Geiger, D., Heckerman, D.: Learning bayesian networks is np-hard. Technical Report MSR-TR-94-17, Microsoft Research, Redmond, WA, USA (November 1994)
3. Cheng, J., Bell, D.A., Liu, W.: An algorithm for bayesian belief network construction from data. In: Proceedings of AI & STAT'97. (1997) 83–90
4. Cooper, G.F., Herskovits, E.: A bayesian method for the induction of probabilistic networks from data. Machine Learning **9** (1992) 309–347 10.1007/BF00994110.

5. Barriere, O., Lutton, E., Wuillemin, P.H.: Bayesian network structure learning using cooperative coevolution. In: Genetic and Evolutionary Computation Conference (GECCO 2009). (2009)
6. Wong, M.L., Lam, W., Leung, K.S.: Using evolutionary programming and minimum description length principle for data mining of bayesian networks. Pattern Analysis and Machine Intelligence, IEEE Transactions on **21**(2) (feb 1999) 174 –178
7. Regnier-Coudert, O., McCall, J.: An Island Model Genetic Algorithm for Bayesian network structure learning. 2012 IEEE Congress on Evolutionary Computation (June 2012) 1–8
8. Friedman, N., Linial, M., Nachman, I., Pe'er, D.: Using bayesian networks to analyze expression data. In: Proceedings of the fourth annual international conference on Computational molecular biology. RECOMB '00, New York, NY, USA, ACM (2000) 127–135
9. Delaplace, A., Brouard, T., Cardot, H.: Computational intelligence and security. Springer-Verlag, Berlin, Heidelberg (2007) 288–297
10. Larranaga, P., Kuijpers, C., Murga, R., Yurramendi, Y.: Learning bayesian network structures by searching for the best ordering with genetic algorithms. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on **26**(4) (jul 1996) 487 –493
11. Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search, 2nd Edition. Volume 1 of MIT Press Books. The MIT Press (2001)
12. Druzdzel, M.J. In: SMILE: Structural modeling, inference, and learning engine and GeNIe: A development environment for graphical decision-theoretic models. American Association for Artificial Intelligence (1999) 902–903
13. Hart, W., Krasnogor, N., Smith, J.: Memetic evolutionary algorithms. In Hart, W.E., Smith, J., Krasnogor, N., eds.: Recent Advances in Memetic Algorithms. Volume 166 of Studies in Fuzziness and Soft Computing. Springer Berlin Heidelberg (2005) 3–27
14. Norman, M., Moscato, P.: A competitive and cooperative approach to complex combinatorial search. In: Proceedings of the 20th Informatics and Operations Research Meeting. (1991) 3–15
15. Neri, F., Cotta, C.: Memetic algorithms and memetic computing optimization: A literature review. Swarm and Evolutionary Computation **2** (2012) 1–14
16. Fan, X.F., Zhu, Z., Ong, Y.S., Lu, Y.M., Shen, Z.X., Kuo, J.L.: A direct first principles study on the structure and electronic properties of $be_xzn_{1-x}o$. Applied Physics Letters **91**(12) (2007) 121121
17. Nguyen, Q.H., Ong, Y.S., Lim, M.H.: A probabilistic memetic framework. Evolutionary Computation, IEEE Transactions on **13**(3) (june 2009) 604 –623
18. Sanchez, E., Schillaci, M., Squillero, G.: Evolutionary Optimization: the uGP toolkit. Springer (2011)
19. Akaike, H.: A new look at the statistical model identification. Automatic Control, IEEE Transactions on **19**(6) (December 1974) 716–723
20. Beinlich, I.A., Suermondt, H.J., Chavez, R.M., Cooper, G.F.: The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks. In: Second European Conference on Artificial Intelligence in Medicine. Volume 38., London, Great Britain, Springer-Verlag, Berlin (1989) 247–256
21. Binder, J., Koller, D., Russell, S., Kanazawa, K.: Adaptive probabilistic networks with hidden variables. Machine Learning **29** (1997)