# Competing Mutating Agents for Bayesian Network Structure Learning

Olivier Regnier-Coudert and John McCall

IDEAS Research Institute, Robert Gordon University, Aberdeen, UK
{o.regnier-coudert,j.mccall}@rgu.ac.uk

**Abstract.** Search and score techniques have been widely applied to the problem of learning Bayesian Networks (BNs) from data. Many implementations focus on finding an ordering of variables from which edges can be inferred. Although varying across data, most search spaces for such tasks exhibit many optima and plateaus. Such characteristics represent a trap for population-based algorithms as the diversity decreases and the search converges prematurely. In this paper, we study the impact of a distance mutation operator and propose a novel method using a population of agents that mutate their solutions according to their respective positions in the population. Experiments on a set of benchmark BNs confirm that diversity is maintained throughout the search. The proposed technique shows improvement on most of the datasets by obtaining BNs of similar of higher quality than those obtained by Genetic Algorithm methods.

**Keywords:** Bayesian network, permutations, distance mutation, genetic algorithm, island model, diversity maintainance.

## 1 Introduction

Bayesian Networks (BN) are probabilistic graphical models composed of a Directed Acyclic Graph (DAG) and a parameter set. Combination of both allows factorization of the joint probability distribution $P$ of a set of variables $X_i$, according to their respective parents $Pa(X_i)$ in the DAG as expressed in (1).

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

Learning BN structure from data is a NP-hard problem and thus a challenging task for the machine learning community. The number of possible DAGs that can be drawn grows super-exponentially with the number of variables and is quantified as $O(n!2^{\frac{n}{2}})$. Typically, two families of methods are considered for BN structure learning, respectively based on conditional independence tests and on search and score approaches. While conditional independent methods focus on determining relationships that exist between variables using statistical tests, this paper makes use of the search and score approach. Solutions are generated and

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part I, LNCS 7491, pp. 216–225, 2012.

scored against a fitness function and are improved until reaching a satisfactory level. In recent years, nature-inspired meta-heuristics have proved successful in efficiently learning structures from data on various problems whether using Genetic Algorithms (GA) [1], Island Models (IM) [2], Ant Colony Optimization (ACO) [3] or Particle Swarm Optimization [4]. Yet, most approaches suffer from the multi-optimal and plateaued nature of the search space, which leads to consequent loss of diversity within the set of solutions and early convergence [2]. In this paper, a novel technique is presented that takes advantage of the permutation representation of the solutions. By considering a population of agents and by assigning different roles to each according to their positions in the population, diversity is maintained throughout the search.

The paper is structured as follows. In sections 2 and 3, we respectively provide background information on BN structure learning and describe a novel approach based on agents and mutation. The experimental approach is introduced in section 4. We finally present and discuss the findings in section 5.

## 2 Background

### 2.1 Search and Score Bayesian Network Structure Learning

In recent years, K2GA [5] has been used as a reference for comparison with many algorithms [1,3]. K2GA uses the greedy K2 [6] algorithm to score solutions within a GA framework. Scoring solutions using K2 is not singular to K2GA and it has been extended to other metaheuristics such as ACO [3] or IM GA [2]. In all K2-based methods, solutions are represented as permutations denoting variable orderings. Given an ordering, K2 only allows a variable to be parent of another variable if the latter is at a further position. This constraint ensures that no cycle is found in solutions produced by K2, hence, that all structures are DAGs. In K2, each network is evaluated using the CH score given in (2), which is a measure of the likelihood of a particular structure to represent the data [7]. $q_i$ represents the number of possible different combinations the parents of the node $X_i$ can take and $r_i$, the arrity of $X_i$, that is its number of possible states. From a dataset $D$, the CH score of the BN structure $B_s$ takes into consideration for each of the $n$ variables the number $N_{ijk}$ of instances where $X_i$ is set to its $k$-th state while its parents are in their $j$-th state. $N_{ij}$ is the sum of $N_{ijk}$ over the different states of $X_i$. K2 starts by assuming that all nodes are independent. Edges are added one at a time and kept if the score of the network is improved.

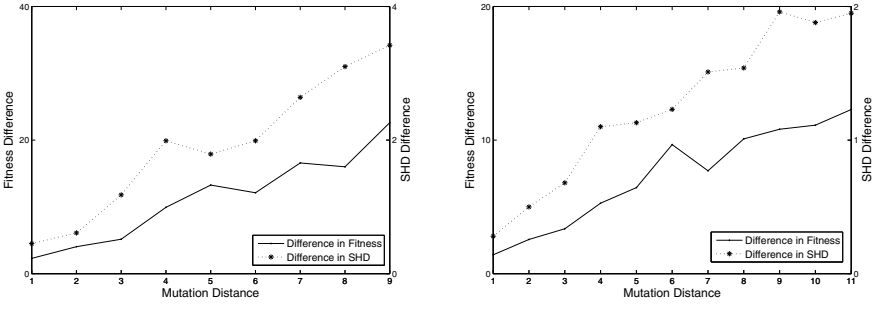
$$P(B_s, D) = P(B_s) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{2}$$

### 2.2 Diversity Enhancements

In [2], IMK2GA, an IM version of K2GA, is compared to K2GA on five benchmarks. In IMK2GA, several evolutions are run in parallel and paused in order to

allow exchange of solutions. The study highlights difficulties for K2GA to avoid early convergence. By migrating solutions between populations, IMK2GA is able to increase the diversity in each population and prevent the search to stop. In addition, the structures learned using IMK2GA were overall better than those obtained by K2GA, illustrating how local optima can be brought together.

In [8], distance mutation (DM), an operator for variable orderings, is introduced. Several Evolutionary Algorithms (EAs) are implemented and tested against each other on benchmark data. Results suggest that DM can be used as the only operator to produce new solutions within EAs and that crossover does not help when used in conjunction of DM. A common way to perform mutation in permutation representations is to swap two genes of an individual. However, due to the ordered nature of the representation and on K2 behavior, the distance between two variables being swapped in an ordering has an influence on how different the mutated solution is from its predecessor. Mutation is often seen as a local change in GA. With regards to BN structure learning, this statement is true when adjacent variables are mutated. However, the locality breaks down as the distance between two swapped variables increases. Structural Hamming Distance (SHD) can be measured to evaluate differences between two BN structures. In order to observe the effect of the distance parameter in the mutation of orderings, 100 random solutions were generated and mutated once for each distance, that is one gene is selected at random in the ordering and is swapped with another at a position related to the mutation distance. Differences in fitness and SHD to the true structure were recorded for the networks obtained after running K2 on these orderings. These are plotted in Figure 1 for two selected benchmarks and show that changes in the BN structures are more important when the distance is increased. In [8], mutation distance varies along the search in order to cope with the loss of diversity in the population and allow exploration in early generations and exploitation in later ones.




**(a)** Car **(b)** Boerlage

**Fig. 1.** Effect of mutation distance on mutated BNs

# 3 Competing Mutating Agents Using Distance Mutation

## 3.1 Competing Mutating Agents

Since early convergence is an issue for many methods, a new algorithm has been implemented that aims at performing both exploratory and exploitative tasks throughout the search. A population of agents is considered in which each agent aims at improving its assigned solution by means of DM. The use of adaptive mutation is not a new concept in EAs as has been seen with mutation rate in previous works including [9]. In this paper, DM distances differ between agents and are set according to the agents' positions in the population. Large distances are allowed for agents in low positions while best agents, that is those with the highest quality solutions, are constrained with smaller distances. Since each agent only uses mutation to improve its solution and because each agent aims at reaching the best positions in the population, we call this approach COMpeting Mutating Agents (COMMA). Algorithm 1 presents the outline of COMMA used for maximization optimization. For each position $pos_j$ in the population $pop$ sorted in ascending order, a DM distance $d_j$ is set such that for two agents at positions $e$ and $f$, $d_e \leq d_f$ if $e < f$ . Since it can be beneficial to allow degrading solutions to be accepted, as seen in simulated annealing [10], a probability $p_j$ is also set for each $pos_j$. Each agent $a_i$ is initially assigned a random solution $s_i$. The population is then sorted by fitness. At each generation, each agent mutates $s_i$ using the distance $dist_i \in [1, d_r]$ defined according to its position $r$ in the population. If the mutated solution $s_{new}$ has a better fitness than $s_i$, $a_i$ replaces $s_i$ with $s_{new}$. If $s_{new}$ has a poorer fitness than $s_i$, $s_{new}$ only replaces $s_i$ with probability $p_r$.

**Algorithm 1.** *COMMA*

```
Initialize pop of σ agents with random solutions, distance vector d of size σ and
probability vector p of size σ
repeat
  Sort pop by fitness in ascending order
  for each agent a_i, i ∈ [0, σ − 1] do
    Get position r of a_i in pop
    Generate new solution s_new with fitness fit_new by mutating s_i with distance
    dist_i selected with uniform probability from [1, d_r]
    if fit_new > fit_i then
      Assign s_i = s_new
    else
      Assign s_i = s_new with probability p_r
    end if
  end for
until Stopping condition met
```

### 3.2 Island Model Competing Mutating Agents

The use of IM showed improvements over its serial counterparts when learning BN structures [2] by maintaining diversity. Hence, IM was also implemented for COMMA and is referred to as IM-COMMA. The outline of IM-COMMA is presented in Algorithm 2 for $n$ islands, $m$ migrations and migration intervals of size $migInterval$. The search is split into evolution stages $stg_l$ where $migInterval$ generations are performed. At each $stg_l$, the COMMA process is performed and paused to allow migration. Solutions from the $\rho$ best agents from each island are sent to the neighbouring island at migration and assigned to the $\rho$ worse agents in its population following a ring topology with a best-worse policy.

**Algorithm 2.** $IM - COMMA$

```
Initialize k populations of σ agents with random solutions, distance vector d of size
σ and probability vector p of size σ
for each evolution stage stg_l, l ∈ [0, m − 1] do
  for each island isl_k, k ∈ [0, n − 1] do
    gen_k = 0
    repeat
      Sort pop_k by fitness in ascending order
      for each agent a_i, i ∈ [0, σ − 1] do
        Get position r of a_i in pop_k
        Generate new solution s_new with fitness fit_new by mutating s_i with dis-
        tance dist_i selected with uniform probability from [1, d_r]
        if fit_new > fit_i then
          Assign s_i = s_new
        else
          Assign s_i = s_new with probability p_r
        end if
      end for
      gen_k + +
    until gen_k = migInterval
  end for
  if l ≠ m then
    Select subpopulation mig_k of size ρ for migration
    if i ≠ 0 then
      Replace ρ worse orderings in pop_k by mig_{i−1}
    else
      Replace ρ worse orderings in pop_k by mig_{n−1}
    end if
  end if
end for
```

## 4 Experimental Approach

In order to evaluate the abilities of the different methods in learning BN structures we selected some known benchmark BNs from which we sampled datasets.

These include *asia* [11], *tank* [12], *credit* [12], *car* [13] and *boerlage* [14]. Characteristics of the benchmarks differ and are summarized in Table 1. K2GA and IMK2GA, were set following [2]. IMK2GA was set with 4 islands and 3 migrations of size $\rho = 2$. Population sizes in K2GA and IMK2GA were set with similar values as described in Table 1. Migration intervals for IMK2GA were set in a way that all migrations occur within a maximum number of 1000 individual fitness evaluations (FEs). Two versions of COMMA and IM-COMMA were implemented in order to observe the effect of degradation. We set one of the COMMA and one of the IM-COMMA with degradation probabilities equal to zero while the two other versions were set with degradation probabilities of [0.2, 0.2, 0.4, 0.4, 0.6, 0.6, 0.6, 0.8, 0.8, 0.8], respectively stated from the best to the worse position in the population of agents. For the time of the experiments, we respectively call these methods $COMMA$, $IM-COMMA$, $COMMA_d$ and $IM-COMMA_d$ where $d$ stands for degradation. As the population size for $COMMA$ and $COMMA_d$ was set to 10, using 4 distinct DM distances and probabilities helped observing how agents evolve. DM distances were set relative to the number of nodes in each benchmark. $IM-COMMA$ and $IM-COMMA_d$ were both set with 4 islands, 3 migrations and 7 generations were chosen as migration interval in order to reach 1000 FEs.

In order to assess the behavior and efficiency of the algorithms, measurements are taken at each generation. Fitness of the best solution is measured, as it helps understanding how the algorithms converge. Kendall Tau Distance (KTD) in the population is also calculated in order to evaluate diversity. The KTD is obtained by averaging the KTD between every pair of orderings in the population. The approach is described in [2]. In addition, it has been shown that increase in the fitness value does not always correlate with improvement in the actual BN structure. Since the true structure of the BNs is known, it is possible to measure the number of correct (C), reversed (R), added (A) and omitted (O) edges of a solution. Based on C, R, A and O, several metrics can be used to describe the distance of a solution to an optimal structure [15]. In the current paper, algorithms are compared according to their respective SHD, representing the number of changes needed to be performed to retrieve the true structure, that is the sum of R, A and O. In addition, R is often regarded as an approximation of low importance. For this reason, the number or relevant edges, that is C + R, is also expressed. On the other hand, it is important not to omit edges, nor to add

**Table 1.** Dataset characteristics and algorithm settings

| **Dataset** | **Nodes /Edges** | **K2GA pop** | **IMK2GA migInterval** | **COMMA pop** | **mutation distances** |
|---|---|---|---|---|---|
| Asia | 8/8 | 100 | 150 | 10 | 1,1,2,2,3,3,3,4,4,4 |
| Tank | 14/20 | 50 | 250 | 10 | 1,1,3,3,5,5,5,7,7,7 |
| Credit | 12/12 | 50 | 300 | 10 | 1,1,2,2,4,4,4,6,6,6 |
| Car | 18/17 | 20 | 1200 | 10 | 1,1,3,3,6,6,6,9,9,9 |
| Boerlage | 23/36 | 20 | 800 | 10 | 1,1,3,3,7,7,7,11,11,11 |

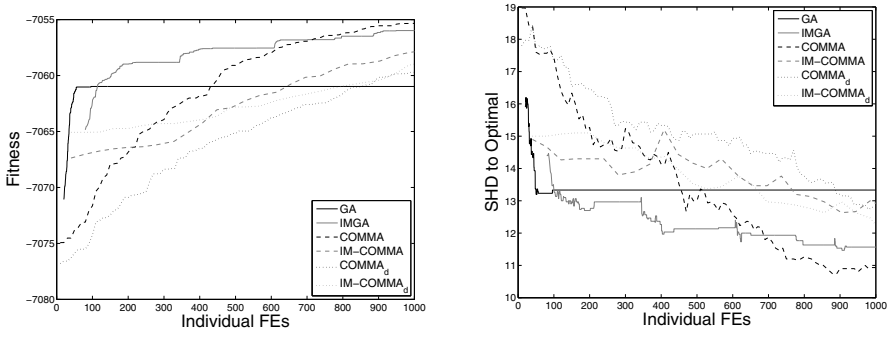
spurious edges to the structure. Consequently, erroneous edges are measured as A + 0. Finally, a greyscale representation was used to map agents' positions in the population over time and illustrate the state of convergence of the method.

## 5 Results and Discussion

For brevity only a sample of all results is presented although similar patterns are observable across benchmarks[1]. Table 2 summarizes the empirical results in terms of C, SHD, relevant (*Rel.*) and erroneous edges (*Err.*) obtained after 1000 FEs. 30 runs were performed for each algorithm on each dataset and unpaired t-tests carried out. Best values over all methods appear in bold while those not statistically significant from the best (p-value>0.003 after Bonferroni correction) are marked with a $*$ symbol. On small problems, K2GA exhibits results that are either the best or not significantly different from it. BNs obtained by K2GA on larger problems such as *car* and *boerlage* suffer from poorer quality in comparisons with other methods as illustrated by the corresponding C and SHD values. The use of IM generally improves the BNs obtained by the GA as the dimension of the data grows and confirms results foreseen in [2]. On the other hand, the standard COMMA is always competitive with the GA-based methods in terms

**Table 2.** Characteristics of best BNs obtained by each algorithm after 1000 FEs

| | | **Asia** | **Tank** | **Credit** | **Car** | **Boerlage** |
|---|---|---|---|---|---|---|
| *K2GA* | C | 6.43 (0.56) | **14.77** (1.75) | 8.87$^*$(1.18) | 10.83 (1.34) | 17.43 (2.26) |
| | SHD | 1.60 (0.66) | **10.93** (2.66) | 4.70$^*$(1.64) | 13.33 (2.88) | 24.80 (3.03) |
| | Rel. | **8.00** (0.00) | 19.67$^*$(0.54) | **12.00** (0.0) | 14.27$^*$(0.81) | 28.07$^*$(0.63) |
| | Err. | 0.03$^*$(0.18) | 6.03$^*$(1.52) | 1.57 (0.56) | 9.90$^*$(2.71) | 14.17$^*$(2.44) |
| *IMK2GA* | C | 6.57$^*$(0.56) | 14.20$^*$(2.04) | 9.23$^*$(0.50) | 12.13$^*$(0.76) | **19.60** (2.14) |
| | SHD | 1.47 (0.67) | 11.30$^*$(2.76) | 3.80$^*$(0.60) | 11.57 (1.96) | **20.90** (3.23) |
| | Rel. | **8.00** (0.00) | 19.83$^*$(0.37) | **12.00** (0.00) | 14.23$^*$(0.76) | 28.03$^*$(0.48) |
| | Err. | 0.03$^*$(0.18) | **5.67** (1.07) | **1.03** (0.18) | 9.47$^*$(2.08) | **12.47** (2.16) |
| *COMMA* | C | 6.70$^*$(0.46) | 13.43$^*$(1.87) | 9.13$^*$(1.23) | **12.47** (0.96) | 18.70$^*$(2.08) |
| | SHD | 1.37 (0.66) | 12.27$^*$(2.64) | 4.27$^*$(1.71) | **10.93** (2.14) | 22.87$^*$(2.59) |
| | Rel. | **8.00** (0.00) | **19.87** (0.34) | **12.00** (0.00) | **14.33** (0.54) | **28.17** (0.64) |
| | Err. | 0.07$^*$(0.36) | 5.83$^*$(1.13) | 1.40 (0.55) | **9.07** (1.67) | 13.40$^*$(1.65) |
| $IM-COMMA$ | C | **6.90** (0.30) | 14.17$^*$(2.03) | **9.63**(1.20) | 11.63$^*$(1.17) | 18.13$^*$(2.32) |
| | SHD | **1.10** (0.30) | 11.83$^*$(2.78) | **3.60** (1.74) | 12.80 (2.41) | 23.53 (3.33) |
| | Rel. | **8.00** (0.00) | 19.70$^*$(0.46) | **12.00** (0.00) | 13.87$^*$(0.81) | 28.07$^*$(0.63) |
| | Err. | **0.00** (0.00) | 6.30$^*$(1.39) | 1.23$^*$(0.62) | 10.57$^*$(2.38) | 13.60$^*$(2.24) |
| $COMMA_d$ | C | 6.87$^*$(0.34) | 14.53$^*$(2.06) | 9.37$^*$(1.28) | 11.40 (1.11) | 17.10 (1.78) |
| | SHD | 1.13$^*$(0.34) | 11.23$^*$(3.24) | 4.10$^*$(1.81) | 12.80 (2.18) | 24.93 (2.39) |
| | Rel. | **8.00** (0.00) | 19.83$^*$(0.37) | **12.00** (0.00) | 14.20$^*$(0.54) | 27.83$^*$(0.90) |
| | Err. | **0.00** (0.00) | 5.93$^*$(1.59) | 1.47 (0.62) | 10.00$^*$(1.73) | 14.20$^*$(2.27) |
| $IM-COMMA_d$ | C | 6.87$^*$(0.34) | 14.43$^*$(2.46) | 9.53$^*$(1.33) | 11.63 (1.08) | 17.87$^*$(2.31) |
| | SHD | 1.13$^*$(0.34) | 11.20$^*$(3.69) | 3.70$^*$(1.93) | 11.93$^*$(2.62) | 24.77 (2.92) |
| | Rel. | **8.00** (0.00) | 19.70$^*$(0.46) | **12.00** (0.00) | 14.17$^*$(0.73) | 28.03$^*$(0.80) |
| | Err. | **0.00** (0.00) | 5.93$^*$(1.67) | 1.23$^*$(0.67) | 9.40$^*$(2.22) | 14.60 (1.98) |




**(a)** Best Fitness **(b)** SHD

**Fig. 2.** Evolution of the best fitness and SHD on *car*, averaged over 30 runs

of BN quality. Figure 2 illustrates how best fitness and SHD vary over time on *car*. The best fitness in $IMK2GA$ appears to be better than those of other methods along the search, while $K2GA$ converges very early. The variation between COMMA and GA-based methods shows different patterns. For both SHD and fitness, $IMK2GA$ follows cycles of improvement and convergence interrupted by migrations of solutions. COMMA approaches show a more gentle improvement over time and the effect of migration is less obvious. Here it seems that migration has a different effect on the algorithms. While migrating solutions helps bringing together local optima within a GA, it does not affect as much COMMA because convergence is never reached. A migration may just be considered as a successful mutation from the $\rho$ worst agents in the population.

To investigate how the different methods behave with respect to diversity within their populations, KTD is plotted in Figure 3 for *car*. The effect of the three migrations on $IMK2GA$ is very clear, but a consequent loss of diversity is still observed that leads to its convergence. Figure 3 also points out that diversity is maintained with the different COMMA algorithms throughout the run. Here, it can be argued that $COMMA$ would perform better on longer runs because it had not converged when experiments were stopped. Although it is difficult to draw a clear picture of the impact of allowing degradation of solutions in $COMMA$, it seems that in the chosen configuration, it does not bring obvious improvements. Figure 4 illustrates how positions of agents vary along the search on *tank* in $COMMA$ and $COMMA_d$ respectively. Each column represents the ordered population of agents in a typical run from the initial state on the left hand side to its final state on the right hand side. Each agent is represented by a shade of grey. For both $COMMA$ and $COMMA_d$, the constant exploration and exploitation is clear as agents change positions many times. During the run of $COMMA$, a total of 8 distinct agents have reached the two first positions in

[1] Additional results and coloured figures are available at `http://www.comp.rgu.ac.uk/staff/orc/ppsn2012-results`

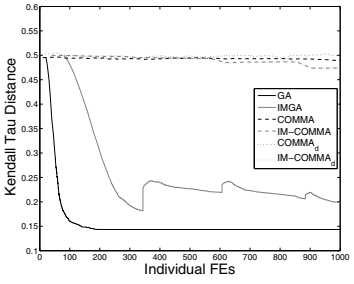



**Fig. 3.** Evolution of the KTD on *car*, averaged over 30 runs

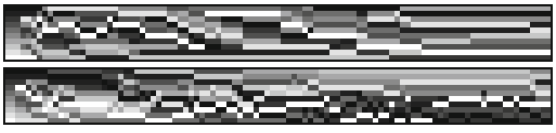


**Fig. 4.** Evolution of agent positions along one typical run of $COMMA$ (top) and $COMMA_d$ (bottom) on *tank*

the population and the final best agent has moved and occupied every position for at least one generation. High pixelation in the lower ranks of the position representation of $COMMA_d$ highlights that new solutions are being explored even when variation in top ranks becomes rare due to the high solution quality.

## 6 Conclusions

In this paper, a novel approach to learning BN structures was presented. The algorithm focuses on avoiding early convergence, a limitation foreseen in other methods such as K2GA by means of agents mutating their solutions to different extents. Since the effect of such mutation largely depends on the distance that is defined, each agent was assigned a mutation distance relative to its position in the population. Results show that COMMA is generally competitive with IMK2GA and is able to outperform it on some of the problems from the selected test suite when considering the quality of BNs obtained at the end of the runs. In addition, diversity remains high in COMMA all along the search while GA-based methods converge fast, suggesting that better BNs could be produced with more FEs. Future work will focus on assessing the effect of different mutation operators on the performance of COMMA, but also to study how such an approach performs on other permutation-based optimization problems. Finally the idea of monitoring the changes in agents positions should be extended in order to perform migrations in a dynamic manner within an IM.

## References


1. Kabli, R., Herrmann, F., McCall, J.: A chain-model genetic algorithm for bayesian network structure learning. In: Proc. of the 9th Conference on Genetic and Evolutionary Computation, pp. 1271–1278 (2007)
2. Regnier-Coudert, O., McCall, J.: An island model genetic algorithm for bayesian network structure learning. In: Proce. of the IEEE CEC 2012 (2012)
3. Wu, Y., McCall, J., Corne, D.: Two novel ant colony optimization approaches for bayesian network structure learning. In: Proce. of the IEEE CEC 2010, pp. 4473–4479 (2010)
4. Cowie, J., Oteniya, L., Coles, R.: Particle swarm optimization for learning bayesian networks. In: Proc. of World Congress on Engineering, pp. 2–4 (2007)
5. Larranaga, P., Kuijpers, C., Murga, R., Yurramendi, Y.: Learning bayesian network structures by searching for the best ordering with genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics 26(4), 487–493 (1996)
6. Cooper, G., Herskovits, E.: A bayesian method for the induction of probabilistic networks from data. Mach. Learn. 9(4), 309–347 (1992)
7. Heckerman, D., Geiger, D., Chickering, D.: Learning bayesian networks: the combination of knowledge and statistical data. Mach. Learn. 20(3), 197–243 (1995)
8. dos Santos, E., Hruschka, E., Ebecken, N.: A distance-based mutation operator for learning bayesian network structures using evolutionary algorithms. In: Proc. of the IEEE Congress on Evolutionary Computation, pp. 1–8 (2010)
9. Thierens, D.: Adaptive mutation rate control schemes in genetic algorithms. In: Proc. of the IEEE CEC, pp. 980–985 (2002)
10. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
11. Lauritzen, S., Spiegelhalter, D.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society, 157–224 (1988)
12. Genie and smile, `http://genie.sis.pitt.edu/` (accessed: March 30, 2012)
13. Norsys, `http://www.norsys.com` (accessed: March 30, 2012)
14. Boerlage, B.: Link strength in bayesian networks. Master's thesis, University of British Columbia (1992)
15. de Jongh, M., Druzdzel, M.: A comparison of structural distance measures for causal bayesian network models. In: Recent Advances in Intelligent Information Systems, pp. 443–456 (2009)