New Local Move Operators for Bayesian Network Structure Learning

Jimmy Vandel and Brigitte Mangin and Simon de Givry
UBIA, UR 875, INRA, F-31320 Castanet Tolosan, France
{jvandel,mangin,degivry}@toulouse.inra.fr

Abstract

We propose new local move operators incorporated into a score-based stochastic greedy search algorithm to efficiently escape from local optima in the search space of directed acyclic graphs. We extend the classical set of arc addition, deletion, and reversal operators with a new operator replacing or *swapping* one parent to another for a given node, *i.e.* combining two elementary operations (arc addition and deletion) in one move. The operators are further extended by doing more operations in one move in order to overcome the acyclicity constraint of Bayesian networks. These extra operations are temporally performed in the space of directed *cyclic* graphs, acyclicity being restored at the end and the move kept if it increases the score. Our experimental results on standard Bayesian networks and challenging gene regulatory nets show large BDeu improvements compared to state-of-the-art structure learning algorithms when the sample size is small.

1 Introduction

Learning the structure of Bayesian networks from fully observed data is known to be an NP-hard problem (Chickering and Heckermann, 1996) which has received a lot of attention from researchers during the last two decades (Daly et al., 2011). Due to its difficulty, heuristic methods have been widely used to learn Bayesian network structures. Among them, score-based methods use a scoring function f to score a network structure with respect to the data. They explore the space of network structures to find the highest-scoring network. This space being superexponential in the number of variables, local search methods are used such as greedy ascent search (also called hill-climbing), tabu search, simulated-annealing, and other complex metaheuristics. In spite of its simplicity, the (repeated randomized or *stochastic*) greedy search method reveals to be a competitive method compared to more complex algorithms (Gámez et al., 2011). Starting from an initial network structure, it performs a series of local moves until a local optimum is found. Each move selects and applies the best elementary operation(s) on the current network structure. The set of candidate neighboring structures is called the *neighborhood* in the sequel. A classical neighborhood is composed of single arc additions, deletions, and reversals. Using *larger* neighborhoods efficiently allows to find better local optima, and thus better network structures.

(Moore and Wong, 2003) proposed an optimal reinsertion of a target node by removing all its edges and reinserting it optimally. (Campos et al., 2002) explored a new reversal operator which deletes all the parents of both nodes if they share some parent, inverts the arc, and adds any subset of the union set of the old parents for each node. However these approaches are limited to small problems only. (Holland et al., 2008) used a restricted form of look ahead called LAGD, combining several operations in a single move. In this paper, we follow a similar approach by focusing our local operations on a target node and combining several operations to improve the score and restore the global acyclicity constraint of Bayes nets. By doing so, we are able to exploit large neighborhoods efficiently. Other approaches use a compact representation of a set of network structures like *Greedy Equivalence Search* (GES) (Chickering, 2002; Nielsen et al., 2003) which considers Markov-equivalent structures.

In Section 2, we give Bayesian network background. Next, we define a specific stochastic greedy search algorithm and introduce the new local move operators in Section 3. We report experimental results in Section 4 and conclude.

2 Bayesian network structure learning

A Bayesian network (Koller and Friedman, 2009) denoted by $B = (G, \mathbf{P}_G)$ is composed of a directed acyclic graph (DAG) $G = (\mathbf{X}, \mathbf{E})$ with nodes representing p random discrete variables $\mathbf{X} = \{X_1, \ldots, X_p\}$, linked by a set of directed edges or *arcs* $\mathbf{E}$[1], and a set of conditional probability distributions $\mathbf{P}_G = \{P_1, \ldots, P_p\}$ defined by the topology of the graph: $P_i = \mathbb{P}(X_i|Pa(X_i))$ where $Pa(X_i) = \{X_j \in \mathbf{X} \,|\, (X_j \rightarrow X_i) \in \mathbf{E}\}$ is the set of parent nodes of X_i in G. A Bayesian network B represents a joint probability distribution on $\mathbf{X}$ such that: $\mathbb{P}(\mathbf{X}) = \prod_{i=1}^{p} \mathbb{P}(X_i|Pa(X_i))$.

Conditional probability distributions $\mathbf{P}_G$ are determined by a set of parameters. Given the network structure G, and the fully observed data D, parameters can be estimated by simple counting, following the maximum likelihood principle.

Learning the structure of a Bayesian network consists in finding a DAG G maximizing $\mathbb{P}(G|D)$. We have $\mathbb{P}(G|D) \propto \mathbb{P}(D|G)\mathbb{P}(G)$. Under specific assumptions, the *marginal loglikelihood* $\log(\mathbb{P}(D|G))$ can be expressed as a decomposable scoring function f (*e.g.* BDeu score (Heckerman et al., 1995)):

$$f(D, G) = \sum_{i=1}^{p} f_{X_i}(D, G) = \sum_{i=1}^{p} f_{X_i}(D, Pa(X_i)) \quad (1)$$

A set of Bayesian networks are Markov-equivalent if they imply exactly the same set or *map* of independence constraints among variables[2]. Next, we describe a novel greedy search method maximizing f in the space of DAGs.

3 Stochastic Greedy Search

We define the Stochastic Greedy Search (SGS) algorithm for structural learning of Bayesian networks. It collects the best DAG found by r randomized hill-climbing algorithms. Stochasticity comes from two random draws. The first one, often considered, is the initial structure used for each restart. The second, more original, is when both edge orientations lead to Markov equivalent DAGs. The DAG is randomly drawn among the best Markov equivalent neighbors of the current DAG G at each step of the hill-climbing algorithm. The neighborhood of G is composed of the usual operations on DAGs: arc addition (ADD), arc deletion (DELETE) and arc reversal (REVERSE). This neighborhood is denoted $\mathcal{N}^{ADR}$ in the sequel. Only operations which do not create cycles are considered. In the next subsections, we are going to extend this set of operations.

Proposition 1. *(Gámez et al., 2011) Let D be a dataset of n records that are identically and independently sampled from some distribution $\mathbb{P}(\cdot)$. Let f be a locally consistent scoring function. The hill-climbing algorithm in SGS returns a minimal independence map of $\mathbb{P}(\cdot)$ as sample size n grows large.*

Recall that BDeu score is locally consistent (Chickering, 2002). The local consistency property ensures adding any arc that eliminates an independence constraint that does not hold in the generative distribution $\mathbb{P}(\cdot)$ increases the score. Conversely, deleting any arc that results in a new independence constraint that holds in $\mathbb{P}(\cdot)$ also increases the score.

The main interest of our randomization approach is to simulate a search in the space of score-equivalent networks. Each greedy search moves from a DAG instance randomly-selected from a Markov-equivalence class $\mathcal{E}(G)$ to another DAG randomly-selected from an *adjacent*[3] Markov-equivalence class $\mathcal{E}(G')$ thanks to our random selection among the best neighbors. It results in a stronger property:

Proposition 2. *(Chickering, 2002) Let D be a dataset of n* iid *fully observed samples of some* faithful *distribution $\mathbb{P}(\cdot)$. Let f be locally consistent. SGS returns a perfect map of $\mathbb{P}(\cdot)$ as both the sample size n and the number of restarts r grow large.*

Recall a faithful distribution admits a unique perfect map corresponding to the optimal structure. Compared to the GES algorithm (Chickering, 2002), which offers the same optimality guarantee within a two-phase greedy search, SGS chooses the orientation of some *compelled arcs*[4] of the *true* DAG at random, whereas GES waits while no *v-structures* impose orientation constraints. See an example in Figure 1.

[1] We use $G = \mathbf{E}$ when the set of nodes is implicit.

[2] BDeu give the same score for Markov-equivalent DAGs.

[3] Two equivalence classes $\mathcal{E}(G)$ and $\mathcal{E}(G')$ are adjacent iff G is an I-map of G' or vice-versa and the number of edges in the graphs G and G' differs by one.

[4] An arc $X \rightarrow Y$ in G is *compelled* if that arc exists in every DAG of $\mathcal{E}(G)$, otherwise it is said *reversible*.

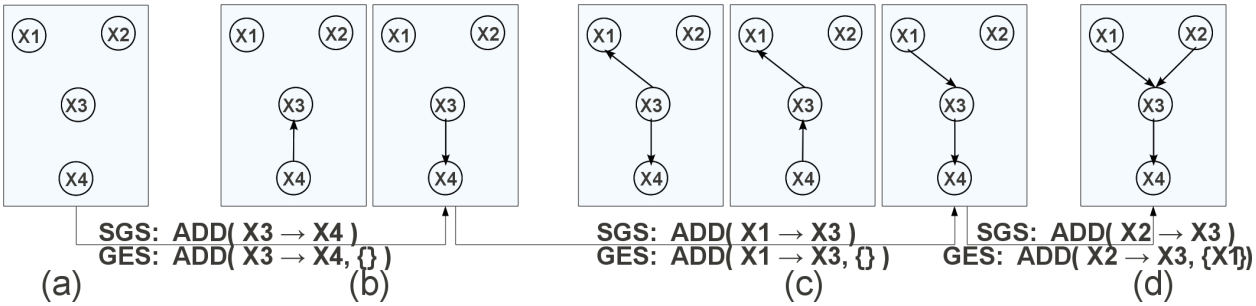

Figure 1: Four adjacent Markov-equivalence classes found by GES during its first phase of edge and v-structure insertions. (a) GES and SGS start from the empty graph . (d) The true DAG is found after three moves. The orientation of $X3 \rightarrow X4$ and $X1 \rightarrow X3$ edges are chosen at random by SGS, whereas GES waits until its third move to decide on edge orientations based on DAG score comparisons (enforcing the v-structure $X1 \rightarrow X3 \leftarrow X2$ as stated by the extra ADD parameter $\{X1\}$, and forbidding $X1 \rightarrow X3 \leftarrow X4$ in its second move).

We observed in the experiments that a small number of restarts r allows to find DAGs with better scores than GES, especially when the sample size n is limited, in this case GES found a local optimum and SGS is able to find other better local optima thanks to randomization. This was also observed in (Nielsen et al., 2003).

When the sample size is small the learning problem becomes more difficult: the empirical distribution may be far from a perfect map resulting in many local optima and the scoring function is no more consistent, *i.e.* the likelihood does not dominate the penalty term on the complexity of the structure which is a non additive function of the parent variable domain sizes (Chickering, 2002). In this complex situation, we propose a new operator to escape from some local optima.

3.1 SWAP operator

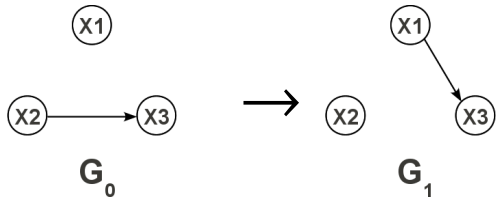

Figure 2: The operator SWAP($X2|X1 \rightarrow X3$) applied to a 3-variable problem.

Consider the 3-variable example in Figure 2 with observed data D, scoring function f, and initial DAG $G_0 = \{X2 \rightarrow X3\}$. Let assume $f(D, \{X1 \rightarrow X3\}) > f(D, \{X2 \rightarrow X3\}) > f(D, \{X1 \rightarrow X3, X2 \rightarrow X3\}) > f(D, \{X3 \rightarrow X1, X2 \rightarrow X3\}) > f(D, \{X2 \rightarrow X1, X2 \rightarrow X3\}) > f(D, \emptyset)$. Then G_0 is a local minimum for the classical neighborhood $\mathcal{N}^{ADR}$. Our new operator, denoted SWAP($X|Y \rightarrow Z$), consists in changing one parent X to another parent Y for one target node Z. This is equivalent to a simultaneous pair of ADD and DELETE operators restricted to the same target node. In our example, applying SWAP($X2|X1 \rightarrow X3$) corresponds to DELETE($X2 \rightarrow X3$),ADD($X1 \rightarrow X3$), resulting in the better DAG $G_1 = \{X1 \rightarrow X3\}$ as shown in Figure 2. The extended neighborhood using the 4 operators is denoted $\mathcal{N}^{ADRS}$ and SGS using $\mathcal{N}^{ADRS}$ (respectively $\mathcal{N}^{ADR}$) is denoted SGS^2 (SGS with Swap) (resp. SGS^1) in the sequel.

Let p be the number of variables in the DAG and k be the maximum number of parents per node. Assuming a sparse graph, $p \gg k$, the number of SWAP operations is larger than for classical operators but it remains bounded by $O(kp^2)$, the complexity of $\mathcal{N}^{ADRS}$ is therefore in $O(kp^2)$, whereas it is in $O(p^2)$ for $\mathcal{N}^{ADR}$. Other approaches using larger neighborhoods such as *h-look ahead in l good directions* (LAGD) has a worst-case complexity in $O(l^{h-1}p^2)$ (Holland et al., 2008), optimal reinsertion is in $O(2^k p^{k+1})$ (Moore and Wong, 2003), and modified reversal in $O(2^{2k}p^2)$ (Campos et al., 2002).

Another source of suboptimality comes from the global acyclicity constraint of Bayesian networks.

3.2 Breaking cycles by successive deletions and swaps

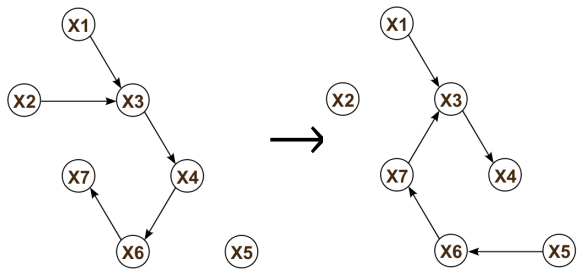

Figure 3: Applying an extended SWAP* operation breaking a cycle by an additional SWAP operation: SWAP*($X2|X7 \rightarrow X3$) = {SWAP($X2|X7 \rightarrow X3$), SWAP($X4|X5 \rightarrow X6$)}.

Consider the 7-variable DAG example in Figure 3. Swapping the parent $X2$ of $X3$ by $X7$ in DAG G (Fig. 3.left) introduces a directed cycle $\{X7 \rightarrow X3, X3 \rightarrow X4, X4 \rightarrow X6, X6 \rightarrow X7\}$ and is therefore forbidden in our $\mathcal{N}^{ADRS}$ neighborhood. However it may correspond to a large *local* score improvement with respect to variable $X3$. Let us denote this improvement by $\Delta_{X3}(G, \text{SWAP}(X2|X7 \rightarrow X3)) = f_{X3}(D, G') - f_{X3}(D, G)$ with G' obtained by applying the SWAP operation on G (G' is not a valid DAG), and D and f being the sample and scoring function. Our idea is to heuristically guide the search for a second (or more) local operator to be applied on G' in order to restore graph acyclicity (G' becomes valid) and such that the *true* score of the final DAG is greater than the score of the original one. In Figure 3, it is obtained by applying a second SWAP.

For that purpose, we define an extended SWAP operator, denoted SWAP*, able to break all directed cycles by performing a succession of deletion or swap operations. It can be seen as a kind of greedy descent search in the space of directed cyclic graphs, trying to remove the less important arcs or to swap them in order to compensate for their loss, until a better valid DAG is found. We use local score changes to guide the search: $\Delta_{Xi}(G, OP) = f_{Xi}(D, G') - f_{Xi}(D, G)$, with G' the result of applying the local move operator $OP \in \{DELETE, SWAP\}$ to G. A negative sum of local changes aborts the search. Recall that finding a minimum number of arc deletions in order to restore acyclicity is NP-hard. We use a greedy approach instead. The pseudo-code of SWAP* is given in Algorithm 1. The local score improvement of the initial SWAP operation is evaluated at line 1. It corresponds to a putative gain on the current score. If it is positive then this operation is applied to a copy of the input DAG G, checking next if it creates some directed cycles. Each cycle is retrieved by the `NextCycle` function and the algorithm searches for an arc deletion in this cycle with minimum deterioration of the local score at line 4. If the combined local score change of the SWAP and DELETE operations is positive then it applies the selected arc deletion and continues to test if there are no more directed cycles at line 2. If the combined local score change is negative then it tries to swap an arc of the cycle such

Algorithm 1: SWAP*($X|Y \rightarrow Z$) operator.

Input : operation $X|Y \rightarrow Z$, sample D, score f, DAG $G(\mathbf{X}, \mathbf{E})$
Output : a set of local operations $\mathbf{L}$
$\mathbf{L} \leftarrow \emptyset$ /* *Initialize output operations to the empty set* */ ;
$\mathbf{X'} \leftarrow \mathbf{X}$ /* *Candidate parent set for future swaps* */ ;
$G' \leftarrow G$ /* *Copy of input DAG* */ ;
1 $\Delta = \Delta_Z(G', \text{SWAP}(X|Y \rightarrow Z))$ /* *Putative score improvement* */ ;
if $\Delta > 0$ **then**
  $\mathbf{L} \leftarrow \mathbf{L} \cup \{\text{SWAP}(X|Y \rightarrow Z)\}$;
  Apply SWAP($X|Y \rightarrow Z$) to G' ;
  /* *Repeat deletion or swap operations until no more cycles* */
2 **while** $\Delta > 0 \wedge (\boldsymbol{C} \leftarrow \texttt{NextCycle}(G')) \neq \emptyset$ **do**
3 $\mathbf{X'} \leftarrow \mathbf{X'} \setminus nodes(\mathbf{C})$;
  /* *Choose the best deletion to break cycle* $\boldsymbol{C}$ */ ;
4 $(U^* \rightarrow W^*) \leftarrow$ argmax$_{(U \rightarrow W) \in \mathbf{C} \setminus \{Y \rightarrow Z\}} \Delta_W(G', \text{DELETE}(U \rightarrow W))$;
  /* *Test if the sum of local score changes is positive* */ ;
  if $\Delta + \Delta_{W^*}(G', DELETE(U^* \rightarrow W^*)) > 0$ **then**
   $\mathbf{L} \leftarrow \mathbf{L} \cup \{\text{DELETE}(U^* \rightarrow W^*)\}$;
   $\Delta \leftarrow \Delta + \Delta_{W^*}(G', \text{DELETE}(U^* \rightarrow W^*))$;
   Apply DELETE($U^* \rightarrow W^*$) to G' ;
  else
   /* *Choose the best swap to get a positive change* */ ;
5   $(U^*|V^* \rightarrow W^*) \leftarrow$ argmax$_{(U \rightarrow W) \in \mathbf{C}, V \in \mathbf{X'}} \Delta_W(G', \text{SWAP}(U|V \rightarrow W))$;
   $\Delta \leftarrow \Delta + \Delta_{W^*}(G', \text{SWAP}(U^*|V^* \rightarrow W^*))$;
   if $\Delta > 0$ **then**
    $\mathbf{L} \leftarrow \mathbf{L} \cup \{\text{SWAP}(U^*|V^* \rightarrow W^*)\}$;
    Apply SWAP($U^*|V^* \rightarrow W^*$) to G' ;
   else
6    $\mathbf{L} \leftarrow \emptyset$ /* *Abort all local operations* */ ;
return $\mathbf{L}$;

that the combined local score change is maximized (line 5) and positive. If it fails to find such an operation then it stops breaking cycles and returns an empty operation set. Finally if it succeeds, breaking all cycles, then it returns a feasible set of SWAP and DELETE operations resulting into a new valid DAG G' with a better score than G. The true score improvement is equal to Δ.

We used a restricted list of alternative candidate parent nodes **X'** at line 5 to avoid that new arcs created by swap operations were swapped again, which guarantees that algorithm 1 terminates.

We apply the same approach to extend the operator ADD*, we note that REVERSE* is unnecessary since REVERSE$^*(X \rightarrow Y)$ = ADD$^*(Y \rightarrow X)$. The resulting neighborhood exploiting these extended operators is denoted $\mathcal{N}^{A^*DS^*}$ and SGS using this neighborhood is denoted SGS^3 (Stochastic Greedy Search with Successive Swaps) in the experiments.

4 Experimental Results

In this section, we describe a set of experiments aimed at testing the performance of SGS^i algorithms compared with state-of-the-art Bayesian network structure learning algorithms on standard Bayesian nets and challenging gene regulatory nets.

4.1 Results on Standard Bayesian Networks

We used four gold-standard networks from the Bayesian Network Repository[5]: Alarm, Insurance, Hailfinder and Pigs networks. The number of nodes varies from 27 to 441 with a connectivity value around 1.5. 100 samples of size $n = 500$ and $n = 5,000$ were generated for each network using Causal Explorer (Aliferis et al., 2003).

We compared SGS^i algorithms with LAGD (Holland et al., 2008), available in the WEKA software (Hall et al., 2009) and GES (Chickering, 2002) implemented in Tetrad 4.4.0 (Scheines et al., 1998). LAGD and GES were shown to outperform or to have similar performance to several recent algorithms in (Salehi and Gras, 2009; Alonso-Barba et al., 2011). Recall that SGS^1 is similar to a repeated randomized orientations hill-climbing, SGS^2 uses the SWAP operator, and SGS^3 breaks cycles by successive DELETE and SWAP operators. Experiments were performed on a 3 GHz Intel Core2 computer with 4 GB running Linux 2.6.

We fixed the maximum number of parents per node at $k = 5$ for SGS^i and LAGD. LAGD exploits a $h = 2$-look ahead in $l = 5$ good directions. GES was restricted on the number of adjacent nodes: $d = 7$ for Hailfinder and $d = 10$ for Pigs network as done in (Alonso-Barba et al., 2011). All methods were initialized by an empty graph and optimized the BDeu score with *equivalent sample size* $\alpha = 1$ and no prior on the network structures. For each sample, we recorded the best score obtained by GES, and by $r = 10$ randomized greedy searches for SGS^i as for LAGD[6].

Table 1: Wilcoxon test comparing pairs of algorithms (familywise error rate = 5%). For Method1 versus Method2, + means that Method1 is significantly better than Method2, − means that Method1 is significantly worse than Method2 and ∼ means there is no significant result

	Alarm		Insurance	
Sample size	500	5,000	500	5,000
SGS^3 vs GES	+	+	+	+
SGS^3 vs LAGD	+	+	+	+
LAGD vs GES	+	∼	+	+
	Hailfinder		Pigs	
SGS^3 vs GES	+	+	+	-
SGS^3 vs LAGD	∼	+	n/a	n/a
LAGD vs GES	+	+	n/a	n/a

In order to test the statistical significance of the difference in BDeu score between two methods, we applied a non-parametric paired test, the Wilcoxon signed-rank test (Wilcoxon, 1945). Table 1 presents the test results for SGS^3 (which outperformed SGS^1 and SGS^2), LAGD and GES by using an unilateral alternative (no difference versus better) and a familywise error rate of 5%.

SGS^3 was the best method for the four networks, except for Pigs network with $n = 5,000$ which is more accurately estimated by GES. We conjecture that in this case, GES was closed to its asymptotic optimal behavior, which may be due to the structure of Pigs network with small nodes in-degree. LAGD

[5] http://www.cs.huji.ac.il/site//labs/compbio/Repository/

[6] We randomly permute the input variables at each run.

failed on the Pigs network due to the large number of variables $p = 441$ that makes the exploration of 2-look ahead neighborhoods infeasible in a reasonable time. GES was the worst method here (except for Pigs) due to limited sample sizes. Results not shown here indicate that SGS^2 improved over SGS^1 and reached the second position and SGS^1 outperformed LAGD which can be explained by a better randomization in our implementation.

Although the algorithms are designed to maximize a (BDeu) score, we generally look for a network structure as close as possible to the true one. We report in Table 2 the means over 100 datasets (rounded values to the nearest integer) of the missing and spurious edges without taking into account the edge orientations. The *structural Hamming distance* (SHD) is the sum of the above values. SGS^3 (resp. GES) got the best SHD in 4 (resp. 5) configurations and outperformed LAGD (which performed as SGS^3 in 1 configuration). GES performed extremely well on the Pigs network, finding the true network with 5,000 samples, whereas SGS^3 learned too many edges but recovered all the true edges (even with $n = 500$). The spurious edges learned by SGS^3 are exclusively due to random orientations of compelled arcs in v-structures (see Figure 1). Assuming $X1 \rightarrow X3 \leftarrow X2$ in the true network (v-structures are very frequent in the Pigs network) and a large sample size, if during its greedy search SGS^3 adds first $X1 \leftarrow X3$ and $X3 \rightarrow X2$, it will add next a *covering edge* $X1 \rightarrow X2$ or $X1 \leftarrow X2$ in order to find a minimal I-map (see Proposition 1). These *covered v-structures* can be found in post-processing by selecting for each one the best configuration among the 3 possible v-structures.

4.2 Detailed analysis on the Alarm network

We further analyzed the impact on performances of the number of restarts r and the initial graph used by SGS^i algorithms on the Alarm network with a sample size $n = 500$. Figure 4 reports averaged BDeu scores on 30 Alarm samples. The 30 initial random graphs, the same set used by all the SGS^i algorithms, are composed of 71 arcs with at most two parents per node[7]. All SGS^i methods reached a better BDeu score than GES in this small sample size

[7]For each node, we randomly select two parents and remove a parent if it creates a directed cycle.

Table 2: Number of spurious edges (+) and missing edges (-) to sum for the structural Hamming distance.

	Alarm		Insurance	
Sample size	500	5,000	500	5,000
SGS^3+	**8**	6	**4**	**2**
SGS^3-	**3**	2	**20**	**8**
LAGD+	11	8	**4**	5
LAGD-	4	2	**20**	11
GES+	**6**	**4**	2	3
GES-	**5**	**2**	23	12
	Hailfinder		Pigs	
SGS^3+	17	**16**	32	41
SGS^3-	24	**13**	0	0
LAGD+	21	20	n/a	n/a
LAGD-	26	19	n/a	n/a
GES+	**15**	11	**2**	**0**
GES-	**24**	22	**7**	**0**

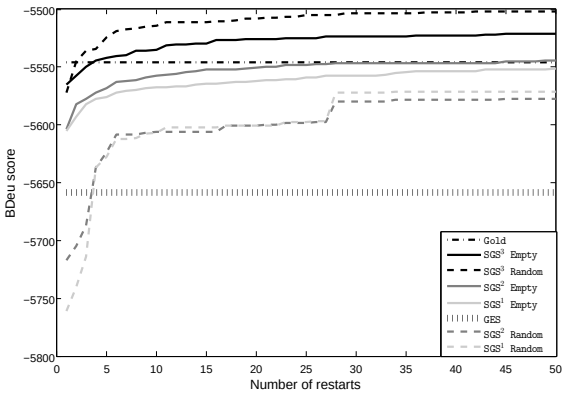

Figure 4: Best BDeu scores, averaged on 30 Alarm samples ($n = 500$), found by SGS^i algorithms as the number of restarts r increases and starting either from an empty (solid line) or a random graph (dashed line). Results of GES (dotted line) and BDeu score of the true network *Gold* (dash-dotted line) are also given. Methods are sorted by decreasing BDeu score at $r = 1$.

situation by the use of less than $r = 4$ restarts. SGS^i methods converged quite rapidly as r increases. For a fixed r, SGS^1 was twice faster than SGS^3. Only SGS^3 found a better score than the true network. Initial random graphs were counter-productive for all the methods, except for SGS^3. It shows that start-

ing from a random graph is useful if the available operators are able to move efficiently in the search space. On the contrary, using randomness to select among the best neighbors was always beneficial.

4.3 Results on Gene Regulatory Networks

Gene regulatory network reconstruction from gene expression data using Bayesian network structure learning was first proposed in (Friedman et al., 2000). We used simulated expression datasets of the DREAM5 Systems Genetics Challenge A (de la Fuente, 2010). Genetics data were not used as they require additional modelling to be taken into account, see *e.g.* (Vignes et al., 2011). Expression data were generated using the SysGenSIM generator (Pinna et al., 2011) based on ordinary differential equation simulation. Five datasets are available for three different sample sizes (n = 100, 300, and 999). The 15 datasets were obtained from *different* known gene networks composed of $1,000$ variables and containing directed cycles. For each sample size, the five network structures contain a different number of edges varying from $\approx 2,000$ (*Net1*) to more than $5,000$ (*Net5*). We discretized gene expression levels into 2 to 4 states using an adaptive method based on an adapted k-means algorithm and the more general framework of Gaussian mixture models as described in (Vignes et al., 2011).

With such large networks, we had to adapt the learning procedure of SGS^i algorithms[8]. We decided to restrict their lists of candidate parents as done in (Goldenberg and Moore, 2004): we selected for each variable X the set of parents $\mathbf{S}$ such that each element Y of S improves the local BDeu score when it is considered as a unique parent compared to the orphan situation ($f_X(D, \{Y \rightarrow X\}) > f_X(D, \emptyset)$). This filtering process was done before the search. In these experiments, SGS^i algorithms have a maximum number of parents per node fixed at k = 5 and use r = 10 restarts. Instead of LAGD (which was too slow), we used MMHC (Tsamardinos et al., 2006) having two steps similar to SGS^i but using mutual information measures. We recorded the best BDeu score of 10 runs for MMHC, by randomly permuting the input variables at each run. All the methods started from an empty graph and optimized the BDeu score with α = 1 and no prior on the network structures.

[8]GES managed in ~1-hour CPU time each network thanks to its better implementation of caching and heap data structure.

Table 3: Wilcoxon test (error rate = 5%) for different gene network sample sizes

	100	300	999
SGS^3 vs MMHC	+	+	+
SGS^3 vs GES	+	+	+
MMHC vs GES	-	~	+

As there were no replicated samples of the same network, we performed the Wilcoxon test on pooled groups for each sample size. We applied a pairwise type I error of 5% and we did not try to correct for multiple comparisons, see Table 3. However, it is worth noting SGS^3 was always the best method, increasing the BDeu score by about 2% in average.

Surprisingly, GES appeared to be better on smaller sample sizes compared to MMHC. MMHC was penalized by its filtering process, especially on the smallest sample size, whereas GES had no restrictions on the candidate parent sets.

In these experiments, the structural Hamming distance (SHD) was not informative due to the poor results reached by all tested algorithms for such large networks, even the empty structure appears better. For this reason, we computed the Euclidean distance to the origin ($\sqrt{precision^2 + recall^2}$). Contrary to SHD, a high distance indicates a better structural quality. We observed in Table 4 contrasted performances between the tested methods depending on the sample size: for n=100, MMHC got the best results, for n = 300, it was GES, and finally SGS^3 performed the best for the largest sample size. Better BDeu scores are not always synonymous with a better structural quality, the limited sample size in addition to the non faithfulness of the data could explain this behavior.

Table 4: Euclidean distances to the origin of the (precision, recall) values. Means of the 5 networks for each sample size.

n	SGS^3	MMHC	GES
100	0.201	**0.258**	0.224
300	0.442	0.437	**0.456**
999	**0.514**	0.482	0.500

5 Conclusion

We presented in this paper new greedy search algorithms called SGS^i exploiting stochasticity from two random draws. We have developed a new local move operator called SWAP and extended versions for ADD and SWAP operators to overcome frequent limitations of local search methods which are local maxima and cyclic situations. We compared SGS^3 using SWAP and extended operators to state-of-the-art methods and we obtained significant BDeu improvements on classical benchmarks and also simulated gene regulatory network datasets when the sample size is small. The complexity of SGS^3 stays moderate with sparse networks.

In the future, we would like to test our operators with other local search methods like tabu search.

Acknowledgements We would like to thank the anonymous reviewers for their helpful comments.

References

C. Aliferis, I. Tsamardinos, A. Statnikov, and L. Brown. 2003. Causal Explorer: A Probabilistic Network Learning Toolkit for Biomedical Discovery. In *Proc. of METMBS'2003*, Las Vegas, Nevada, USA.

J. Alonso-Barba, L. de la Ossa, and J. Puerta. 2011. Structural learning of bayesian networks using local algorithms based on the space of orderings. *Soft Comput*, pages 1881–1895.

L.M. de Campos, J.M. Fernandez-Luna, and J.M. Puerta. 2002. Local Search Methods for Learning Bayesian Networks Using a Modified Neighborhood in the Space of DAGs. In *LNCS (2527)*, pages 182–192.

D. Chickering and D. Heckermann. 1996. Learning bayesian networks is NP-complete. *In learning from data: Al and Statistics*.

D. Chickering. 2002. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554.

R. Daly, Q. Shen, and S. Aitken. 2011. Learning Bayesian networks: approaches and issues. *The Knowledge Engineering Review*, 26(2):99–157.

A. de la Fuente. 2010. The DREAM5 Systems Genetics Challenges. `http://wiki.c2b2.columbia.edu/dream/index.php/D5c3`.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. 2000. Using bayesian networks to analyse expression data. *Journal of computational biology*, 7(3/4):601–620.

J. Gámez, J. Mateo, and J. Puerta. 2011. Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. *Data Min. Knowl. Discov.*, 22:106–148.

A. Goldenberg and A. Moore. 2004. Tractable learning of large Bayes net structures from sparse data. In *Proc. of ICML'04*, pages 44–51.

M. Hall, F. Eibe, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. 2009. The WEKA Data Mining Software. *SIGKDD Explorations*, 11.

D. Heckerman, D. Geiger, and D. Chickering. 1995. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. In *Machine Learning*, volume 20, pages 197–243.

A. Holland, M. Fathi, M. Abramovici, and M. Neubach. 2008. Competing fusion for bayesian applications. In *Proc. of IPMU 2008*, pages 378–385.

D. Koller and N. Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

A. Moore and W.K. Wong. 2003. Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning. In *Proc. of ICML '03*, pages 552–559.

J. Nielsen, T. Kocka, and J. Pefia. 2003. On Local Optima in Learning Bayesian Networks. In *Proc. of UAI-03*, pages 435–442.

A. Pinna, N. Soranzo, I. Hoeschele, and A. de la Fuente. 2011. Simulating system genetics data with SysGenSIM. *Bioinformatics*, 27:2459–2462.

E. Salehi and R. Gras. 2009. An empirical comparison of the efficiency of several local search heuristics algorithms for Bayesian network structure learning. In *Proc. of LION 3*, Trento, Italy.

R. Scheines, P. Spirtes, C. Glymour, C. Meek, and T. Richardson. 1998. The TETRAD Project: Constraint Based Aids to Causal Model Specification. *Multivariate Behavioral Research*, 33(1):65–117.

I. Tsamardinos, L. Brown, and C. Aliferis. 2006. The max-min hill-climbing Bayesian network structure learning algorithm. *Mach. Learn.*, 65:31–78.

M. Vignes, J. Vandel, D. Allouche, N. Ramadan, C. Cierco, T. Schiex, B. Mangin, and S. de Givry. 2011. Gene Regulatory Network Reconstruction Using Bayesian Networks, the Dantzig Selector, the Lasso and Their Meta-Analysis. *PLoS ONE*, 6(12).

F. Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1:80–83.