Bayesian Network Hybrid Learning Using a Parent Reducing Site-specific Mutation Rate Genetic Algorithm

BY

CARLO CONTALDI

B.S., Politecnico di Torino, Turin, Italy, 2014

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Chicago, 2016

Chicago, Illinois

To my father, for silently instilling the mastery of the art of perseverance and always giving me a chance to thrive.

To my mother, for accepting nothing less than completion from me and having the faith to intercede for me to my grandfather.

To my sister, for always having the power to show me where the light is or how to create it with the brightest of smiles, when my path or even hers is the darkest.

ACKNOWLEDGMENTS

Though only my name appears on the cover of this dissertation, its production would have not been possible without the contribution and guidance of many, remarkable individuals.

This thesis work is of great significance to me: convinced that my Master's thesis would constitute the apex of my graduate experience, I spent a long time seeking the research task that best suited my intellectual proclivities. Despite my exhaustive search, my meticulous demands made me delay my choice until I realized a limited time divided me from expected submission date: I would like to express my deep appreciation and gratitude to my advisor, Dr. Peter C. Nelson, for promptly providing me with the freedom to pursue a task that inspired my creativity more than anything else in my academic career, and always committing to patiently addressing every single requirement of my graduate program. I am also thankful to him for believing in me from the very first moment we met; his enlightened and affable gaze always succeeded in dispelling my doubts and giving me the strength to achieve my final goal. Furthermore, I thank him for providing the academic support to carry out my research work within the Computer Science Department.

I am extremely grateful to Dr. Fatemeh Vafaee, co-advisor and mentor. She gave me the freedom to explore on my own, and at the same time the guidance to recover when my steps faltered. She has always been there to patiently listen and give advice, and I deeply respect the enthusiasm she expressed during her supervision. Despite the thousands of miles that divided us during this experience, she was intellectually entangled with me in the task more than anyone else, always on the front line, ready to tackle the challenge by my side. Moreover, I am thankful to her for encouraging consistency, as well as her critical analysis and research professionalism through the process of writing this thesis.

My grateful thanks are also extended to the rest of my thesis committee, Dr. Mariagrazia Graziano and Dr. Rashid Ansari, for their support and for always being there.

Lynn Thomas deserves a special mention: she has been my primary guide across my whole graduate experience in the USA; she took care of countless administrative duties related to my permanence in the States and flawlessly organized my path as a graduate student. This adventure would not have been the same without her genuine friendship and sympathetic support to my self-realization.

I would like to acknowledge the entire Artificial Intelligence Laboratory staff for being ready to promptly address my requests and for providing the facilities to carry out the research work.

I am deeply thankful for my relatives and friends for always being there for me, as well as for helping me stay sane and keep my humanity.

Most importantly, I would like to express my heart-felt gratitude to my family, who has been a constant source of love, concern, support and strength throughout these years.

I owe sincere and earnest thankfulness to all those people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

CC

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (continued)

FIGURE **PAGE**

LIST OF FIGURES (continued)

LIST OF FIGURES (continued)

LIST OF ABBREVIATIONS

BN	Bayesian Network
DAG	Directed Acyclic Graph
PDAG	Partially Directed Acyclic Graph
CPDAG	Completed Partially Directed Acyclic Graph
S&S	Search and Score
CB	Constraint-Based
SS	Super-Structure
CI	Conditional Independence
GA	Genetic Algorithm
SiRG	Site-specific Rate Genetic
DiG	Diversity Guided
PaRe	Parent Reduced
SPaRe	Self Parent Reducing

SUMMARY

Bayesian networks constitute a powerful framework for probabilistic reasoning and expert elicitation, capable of representing inner relationships underlying any kind of phenomenon in both causal and diagnostic directions. Motivated by the fact that Bayesian networks have been extensively used in a variety of research domains, I focused this thesis work on unsupervised Bayesian network structure learning: it is an ambitious as challenging task that, if accomplished, would pave the path for Bayesian modeling and would therefore contribute to provide further insight into countless state-of-the-art research topics.

The problem of Bayesian network structure learning can be formalized as a search task and, from its inception back to 1980s, it has been tackled by means of two distinct strategies or their hybrid combination: a Constraint-Based method operates by progressively reducing the search space, whereas the Search and Score generic approach explores the search space guided by some knowledge-driven metric.

This thesis work aims at providing a method able to learn the structure of the Bayesian network underlying a set of data samples, by focusing on problems with a limited amount of available data. In particular, the main contribution of this work is a Hybrid learning algorithm able first to reliably reduce the search space and then to exhaustively explore it, by taking advantage of data-informed expedients as well. The proposed approach involves a parameterized Genetic Algorithm in order to pursue the task: this metaheuristic has been chosen because of its efficient global search capabilities even across a very large search space and because of its flexibility and adaptability; on the other hand it consistently suffers from time and space complexity relatively to other search methods in the literature, and moreover its performance is heavily influenced by the choice of a large set of parameters.

The research covered in this work is concerned with designing a series of hybrid methods on a build-up basis: they are provided with enhancements already existing in the literature but not yet applied to the Bayesian network structure learning topic and also with novel improvements able to further restrict the search space during the evolutionary process, in a data-driven manner. In the experimental chapter of this thesis it is possible to ascertain how presented algorithms allow to better address time and space complexity, sensitivity to parameters setting issues as well as the problem of data fragmentation, with the advantage of higher performances in some cases.

CHAPTER 1

INTRODUCTION

In the part of world accessible by man's senses or existing instruments, knowledge that can be empirically inferred from any experience or experiment is in general affected by uncertainty and nondeterminism. In several domains, such as medicine, law or business, any assertion is always characterized by a certain *degree of belief.* For example, a robot needs to know about the possible outcomes of its actions, and a medical expert system needs to know which causes lead to which effects.

The final goal in this context is the availability of a *general framework* to enable probabilistic reasoning in any new application without reinventing everything from scratch. Bayesian networks offer exactly such a domain-independent framework for probabilistic reasoning, but a problem still remains: since there does not exist a universal method for *Bayesian Network Structure Learning* yet, a human expert is still generally needed in order to design the network of interest.

This chapter first provides some background about probability theory, Bayesian networks fundamentals – including a brief overview on graph theory and Bayesian network structure learning – and evolutionary computation, so to supply the reader with the knowledge needed to understand what this thesis work deals with. The research motivation and the contribution of this work are then discussed. At the end of the chapter, an outline of the thesis is given.

1 Probability Theory

As stated by Russell and Norvig in [1], "probability provides a way of summarizing the uncertainty that comes from our *laziness* and *ignorance*." In fact, it is not the real world to be affected by nondeterminism: if we define the complex of means through which the phenomenon of interest is investigated as *the agent*, probability statements describe only the agent's view of the world, i.e. his *state of knowledge*, and not the actual world.

1.1 Fundamentals and Basic Notation

In probability theory a *possible world* ω is defined to be an assignment of values to all of the random variables under consideration.

The set of all possible worlds is called the *sample space*, denoted as Ω; in this space all possible worlds are *mutually exclusive* and *exhaustive.*

A fully specified *probability model* associates a numerical probability $\Pr(\omega)$ with each possible world, so that:

$$0 \leq \Pr(\omega) \leq 1 \quad \forall\omega \qquad\qquad \sum_{\omega\in\Omega} \Pr(\omega) = 1$$

Probabilities that refer to degrees of belief in propositions in the absence of any other information are called *priors* or *unconditional probabilities.* On the other hand, when we have some information to exploit (called *evidence*) we can make use of a *posterior* or *conditional probability*:

$$\Pr(a|b) = \frac{\Pr(a \wedge b)}{\Pr(b)}, \quad \Pr(b) \geq 0$$

Variables in probability theory are called *random variables*; every random variable has a *domain*, i.e. the set of possible values it can take on.

Given a set of random variables, their *joint probability distribution* is a probability distribution containing the probabilities of all combinations of variable values.

A *probability model* is fully determined by the *full joint probability distribution.*

Probability theory is based on *Kolmogorov Axioms* [2]:

$$\begin{gathered} \Pr(\omega) \in \mathcal{R}, \quad \Pr(\omega) \geq 0 \qquad \forall \omega \in \Omega \\ \Pr(\Omega) = 1 \\ \Pr\left(\bigcup_{i=1}^{\infty} \omega_i \right) = \sum_{i=1}^{\infty} \Pr(\omega_i) \end{gathered} \tag{1.1}$$

Bayes' Rule describes the probability of an event, based on conditions that might be related to the event [3].

$$\Pr(b|a) = \frac{\Pr(a|b)\Pr(b)}{\Pr(a)} \tag{1.2}$$

If b is the *cause* and a the *effect*, $\Pr(a|b)$ quantifies the relationship in the *causal* direction, whereas $\Pr(b|a)$ describes the *diagnostic* direction.

Two events are said to be *independent* of each other when the probability that one event occurs in no way affects the probability of the other event occurring. Similarly, two events a and b are *conditionally independent given a third event* c if the occurrence or non-occurrence of a and the occurrence or non-occurrence of b are *independent events* in the portion of the sample space defined by knowledge on c occurrence.

$$\Pr(a, b|c) = \Pr(a|c)\Pr(b|c)$$

The general concept of independence is denoted in this work with the symbol $\perp\!\!\!\perp$. Therefore:

- independence between two events a and b can be denoted with the expression $a \perp\!\!\!\perp b$;
- conditional independence between two events a and b with respect to a third event c can be described with the expression $a \perp\!\!\!\perp b \mid c$.

If, given $n+1$ variables, we can represent them as *conditionally independent* with respect to one of them, the size of this new representation grows as $\mathcal{O}(n)$ instead of $\mathcal{O}(2^n)$ as in the case in which all variables are *dependent.* Indeed, conditional independence assertions may allow probabilistic systems to *scale up.*

2 Bayesian Networks

Nowadays, Bayesian Networks are more and more used for modeling knowledge in various domains such as computational biology, bioinformatics, medicine, information retrieval, semantic search, image processing and security.

A *Bayesian Network* (BN) is a data structure that represents *dependencies* among random variables.

$$B(\mathcal{G}, \Theta), \qquad \mathcal{G} = (V, E)$$

Each node X_i in V corresponds to a random variable, which may be *discrete* or *continuous.*

A set of directed links defined in E connects pairs of nodes, so that the resulting graph $\mathcal{G}$ is a *Directed Acyclic Graph* (DAG), i.e. a graph not containing any cycle.

Each *node* X_i is characterized by a *Conditional Probability Distribution* (CPD) $\Pr\left[X_i | \text{Parents}(X_i)\right]$, that quantifies parents' effect on it; Θ is a *set of parameters* that determines the graph edges by specifying the above-mentioned local conditional probabilities.

The *topology* of the network specifies the conditional independence relationships that hold in the domain – *causes* should be parents of *effects.*

The CPD for a *discrete* variable is represented as a *conditional probability table* (CPT), where each row contains the conditional probability of each node value for every *conditioning case*, i.e. every possible combination of values for the parent nodes [1].

A Bayesian Network can be seen as a representation of the *joint probability distribution*: a generic entry of it is the probability of a particular combination of assignments to all variables in the sample space.

In order to construct a Bayesian Network we can apply the *chain rule*:

$$\Pr(X_1, \ldots, X_n) = \prod_{i=1}^{n} \Pr(X_i | X_{i-1}, \ldots, X_1) = \Pr[X_i | \text{Parents}(X_i)]$$

Definition 1 (Latent Structure) [4] *A* latent structure *is a pair $L = (D, O)$, where D is a causal structure over V and where $O \subseteq V$ is a set of observed variables.*

Definition 2 (Consistency) [4] *A latent structure $L = (D, O)$ is consistent with a distribution $\hat{P}$ over O if D can accommodate some model that generates $\hat{P}$ – that is, if there exists a parameterization Θ_D such that $P_{[O]}(D, \Theta_D) = \hat{P}$*

2.1 Bayesian Networks: a Simple Example

In Figure 1 it is represented a simple, canonical BN originally introduced in [1]. The four variables in the system are discrete: this means that their CPDs are represented by means of CPTs; moreover each variable is *binary.*

Given that the variables are binary, the probability that each conditioning case happens sums up to one with the probability that it does not happen: indeed in the reported CPTs only the probability of the type $\Pr(X = T)$ is shown, because we can simply calculate $\Pr(X = F)$ as $1 - \Pr(X = T)$.

In this system we can identify a variable that has no parents and thus specifies a *prior*, “Cloudy,” two intermediate variables “Sprinkler” and “Rain,” and an observed variable “Wet-Grass.” It models an inference system in which, on the basis of a prior knowledge about the

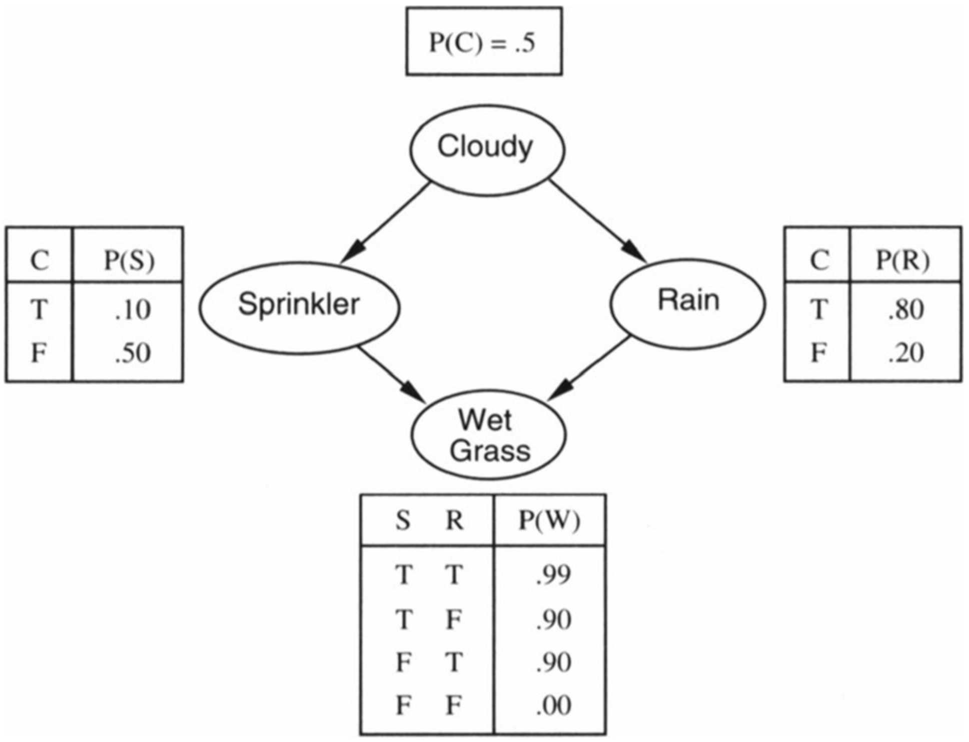


Figure 1: Simple Bayesian Network provided with each node's related Conditional Probability Distribution.
Source: RUSSELL, STUART; NORVIG, PETER, ARTIFICIAL INTELLIGENCE: A MODERN APPROACH, 3rd, ©2010, p. 529. Reprinted by permission of Pearson Education, Inc., New York, New York.

likelihood of a cloudy weather and the state of the grass (wet or not wet), we attempt to infer the likelihood of intermediate events occurrence, i.e. how much it is likely that grass has been watered by sprinkler action or by rain.

2.2 Graph Theory

In this subsection is provided a set of definitions and notions in the field of Graph Theory constituting the basis of any Bayesian network structure learning strategy presented in this thesis work.

Definition 3 (Adjacency) [5] *Two variables X and Y are* adjacent *if there is an edge between X and Y.*

Definition 4 (Partially directed acyclic graph) [6] *A* Partially Directed Acyclic Graph *(PDAG) is a graph where some edges are directed and some are undirected.*

A PDAG is also known as a *pattern* [7].

Definition 5 (v-structure) [8] *In a DAG, a* v-structure *is given by two converging arrows whose tails are not connected by an arrow.*

Definition 6 (Equivalence) [9] *Two DAGs $\mathcal{G}$ and $\mathcal{G}'$ are* distributionally equivalent *if for every Bayesian network $\mathcal{B} = (\mathcal{G}, \Theta)$ there exists a Bayesian network $\mathcal{B}' = (\mathcal{G}', \Theta')$ such that* B *and* B$'$ *define the same probability distribution, and vice versa.*

Definition 7 (Compelled edge) [9] *A directed edge $X \to Y$ is* compelled *if that edge exists in every DAG $\mathcal{G}'$ that is equivalent to $\mathcal{G}$.*

If an edge is not compelled in a graph, then it is *reversible* in the graph [9].

Definition 8 (Completed Partially Directed Acyclic Graph) [9] *A* Completed Partially Directed Acyclic Graph *(CPDAG) corresponding to an equivalence class is the PDAG consisting of a directed edge for every compelled edge in the equivalence class, and an undirected edge for every reversible edge in the equivalence class.*

A CPDAG is also known as an *essential graph* [10].

The essential graph of a DAG is its *skeleton*.

Theorem 1 (Verma and Pearl [8]) *Two DAGs are* equivalent *iff they have the same* skeleton *and the same* v-structures*.*

As stated by Pearl and Verma in [8], "the structural constraints that an underlying dag imposes upon the probability distribution are equivalent to a finite set of conditional independence relationships asserting that, given its parents, each variable is conditionally independent of all its non-descendents. Therefore two causal models are equivalent (i.e. they can mimic each other) if and only if they relay the same dependency information."
Theorem 1 is founded upon the dependency information.

Definition 9 (Perfect Map) [9] *$\mathcal{G}$ is a* perfect map *of a probability distribution P if every independence constraint in P is implied by the structure $\mathcal{G}$ and every independence implied by the structure $\mathcal{G}$ holds in P.*

If there exists some DAG that is a perfect map of a probability distribution P, we say that P is *DAG-perfect* (or perfect with respect to a DAG) [9].

Definition 10 (Vertex Indegree) [11] *The* indegree *of a vertex v in a directed graph, denoted as $deg^{-}(v)$, is the number of edges directed to it.*

A vertex with $deg^{-}(v) = 0$ is called a *source* [11].

Definition 11 (Vertex Outdegree) [11] *The* outdegree *of a vertex in a directed graph, denoted as $deg^{+}(v)$, is the number of edges directed out of it.*

A vertex with $deg^{+}(v) = 0$ is called a *sink* [11].

The *degree sum formula* [11] states that, for a directed graph,

$$\sum_{v \in V} \deg^{+}(v) = \sum_{v \in V} \deg^{-}(v) = |E|, \tag{1.3}$$

where E denotes the set of edges in the directed graph.

If $deg^{+}(v) = deg^{-}(v) \quad \forall v \in V$, the graph is called a *balanced directed graph* [11].

2.3 Bayesian Network Structure Learning

Bayesian Networks constitute a powerful tool for understanding the dynamics underlying a complex system behavior, including what concerns causal relationships among the single agents composing the system itself.

In order to use Bayesian Networks to model domain variables and their relationships from *empirical observations* or *sampled data*, one has to:

learn the underlying DAG structure;

estimate the conditional probability distributions.

We can create BNs manually by interviewing experts in the problem domain, but this process is time-consuming and error-prone, and the resulting problem not only would inherit any flaws in the experts' reasoning but would also be greatly affected by scalability issues; moreover, in some domains simply there is no expert with all the required knowledge.

2.3.1 Available Challenges

A goal of Bayesian Network Structure Learning is, given a phenomenon and a dataset directly sampled on its basis, to get an as complete as possible picture of inner interactions underlying it; in other words, if we can:

- identify the complete set of relevant entities that play a non-negligible role in the phenomenon, i.e. the set of agents;
- get some information from the agents by means of a source of knowledge so that a set of raw data is available directly from the phenomenon itself;

then it is possible to get insights on a subset of causal relationships possibly existing for each pair of agents, including which one of the two agents triggers a variation on the other one's behavior.

Structure learning given a set of variables and fully observed data constitutes the objective which this thesis work is focused on, but it is not the only possible formulation for the structure learning problem. Indeed we can deal with partially observed data, or even hidden variables;

for what concerns the model itself, we could already have available the structure and have to learn the parameters only.

All possible tasks in BN Structure Learning are represented in Table I [12].

TABLE I: DIFFERENT TASKS IN LEARNING BAYESIAN NETWORKS. N/A INDICATES A NON-EXISTING TECHNIQUE FOR THE RELATED TASK.

		Model constraints		
		Known structure. Learn parameters	Known set of variables. Learn structure + parameters	Known subset of variables. Learn hidden variables + structure + parameters
Data observability	Fully observed	closed-form MLE / closed-form Bayesian parameter estimation	constraint-based approaches / search-and-score approaches	N/A
	Partially observed	gradient ascent / expectation maximization (EM)	approximate score / structural EM	N/A
	Hidden variables	N/A	N/A	(here be dragons)

Source: C. Berzan, *An exploration of structure learning in Bayesian networks*, Doctoral Dissertation, Tufts University, 2012.

Another important distinction to do is the purpose we want to achieve: the standard goal is given by *knowledge discovery*, i.e. learn a model in order to gain knowledge about the domain; other objectives can be *classifier learning*, when we need to use the model as a classifier or *density estimation*, when we want to construct a (simpler) model representing a distribution that resembles as much as possible the true distribution, able to capture only relevant aspects of it.

In the rest of this thesis, we will focus on a single flavor of the learning problem: *learning BN structure from fully observed data given a known set of variables*, for the purpose of *knowledge discovery.*

2.3.2 Theoretical Foundations

Sometimes it is not easy to directly infer the causality role in a variable, because actual causality may occur at a *different level* than the one taken into account [4].

By quoting Whitehead's words in [13], "causality is an abstraction that indicates how the world progresses, so basic a concept that it is more apt as an explanation of other concepts of progression than as something to be explained by others more basic [i.e. it is such a basic concept that it is more suitable to clarify other progression concepts than any other abstraction, even a more primitive one]. [...] For this reason, a leap of intuition may be needed to grasp it."

In order to clarify what a different level means, let us have an explanatory example through a physics analogy. Let us imagine a body that is falling near Earth's surface: a derivation to Newton's Second Law of Motion tells us that $X \approx \frac{1}{2}gt^2$ represents the distance to the ground of the falling body in terms of time elapsed t. Even if X depends on t, it is not true that t

causes X: indeed Newton's Law ($F = ma$) states that *forces* cause the motion, and not time itself. However, a *covariation* analysis would make us deduce that a causal relationship exists between time and falling height.

As stated by Pearl in [4]: "an *autonomous intelligent system* attempting to build a workable model of its environment cannot rely exclusively on *preprogrammed causal knowledge*; rather, it must be able to translate direct observations to cause-and-effect relationships. However, given that statistical analysis is driven by *covariation*, not *causation*, and assuming that the bulk of human knowledge derives from uncontrolled observations, we must still identify the clues that prompt people to perceive *causal relationships* in the data. We must also find a computational model that emulates this perception."

Given a *learning algorithm* that has to generate a discrete-variable Bayesian Network from data and a *scoring criterion* that favors the simplest structure for which the model is as faithfully as possible able to represent the distribution, it results that, as Chickering et al. describes in [14]: "identifying *high-scoring structures* is NP-hard, even when any combination of one or more of the following hold:

- the generative distribution is perfect with respect to some DAG containing hidden variables;
- we are given an independence oracle;
- we are given an inference oracle;

- we are given an information oracle;
- we restrict potential solutions to structures in which each node has at most k parents, for all $k \geq 3$."

Indeed, the number of possible structures grows *super-exponentially* by the number of variables n, i.e. with a complexity $\mathcal{O}(n!2^{\binom{n}{2}})$: consequently, exact methods result to be *infeasible* for any domain [14, 15].

In particular, Murphy pointed out in [16] that "the number of DAGs as a function of the number of nodes, $G(n)$, is given by the following recurrence:

$$G(n) = \sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{k(n-k)} G(n-k)." \tag{1.4}$$

In Table II the first few values of the above-mentioned super-exponential correspondence are provided.

2.3.3 Approximate Structure Learning

Approximate structure learning algorithms are usually categorized into *Constraint-Based* (CB) and *Search and Score* (S&S) approaches [15].

Constraint-based algorithms rely on the application of a number of *Conditional Independence* (CI) tests to determine, given a conditioning set, whether two variables are independent.

TABLE II: SUPER-EXPONENTIAL CORRESPONDENCE BETWEEN THE NUMBER OF VARIABLES OF A BN AND THE NUMBER OF POSSIBLE RELATED DAGS REPORTED FOR THE FIRST 10 VALUES.

Number of Variables	**Number of Possible DAGs**
1	1
2	3
3	25
4	543
5	$2.9 * 10^4$
6	$3.8 * 10^6$
7	$1.1 * 10^9$
8	$7.8 * 10^{11}$
9	$1.2 * 10^{15}$
10	$4.2 * 10^{18}$

First they learn direct dependences between variables and build an *undirected graph structure*, then they direct edges by means of *orientation rules*.

A CB procedure requires in the simplest implementations a number of CI tests growing *exponentially* with the number of variables: as a consequence CB methods incorporate different assumptions to ensure independence correctness as well as to restrict *condition set size*; indeed, as described by Margaritis in [17], "the existence of these independences in the actual population depends on the extent to which these assumptions hold. These are:

Causal Sufficiency Assumption: there exist no common unobserved (also known as hidden or latent) variables in the domain that are parent of one or more observed variables of the domain.

Markov Assumption: Given a Bayesian network model B, any variable is independent of all its non-descendants in B, given its parents.

Faithfulness Assumption: a BN graph $\mathcal{G}$ and a probability distribution P are *faithful* to one another iff every one and all independence relations valid in P are those entailed by the Markov assumption on $\mathcal{G}$."

CB strategies are deterministic and characterized by a well-defined stopping criterion: a CB algorithm is iterated so that an edge per time is checked for CI with respect to a condition set of increasing size, until no variable has more than a certain number of adjacencies or when the maximum allowed size for the condition set has been explored in all cases.

As Kruse et al. stated in [18], "unfortunately, CI tests tend to be unreliable unless the volume of data is enormous," thing that makes CB methods sensitive to failures in these tests.

Search and Score approaches are based on a search within the space of possible structures, driven by the outcomes of a *scoring function*, that measures the fitness of each structure to the data.

The learning process operates by moving from one structure to another through the usage of some *variation operators* so to explore the possible structures and maximize the score at each step, until an optimal score is found or a stop-criterion is met.

Latest advances in Bayesian Network Structure Learning involve the adoption of an *Hybrid* approach that attempts to take the best of both CB and S&S worlds, in which first a CB method is applied in order to constrain and reduce the search space, and then a S&S method is used so to identify the possibly optimal structure in the remaining portion of the search space.

2.3.4 D-Separation

The notion of conditional independence by itself is not sufficient to fully determine whether two variables are dependent or not, in particular in presence of *evidence variables* or *colliders*.

As explained by Pearl in [19], "*d-separation* is a criterion for deciding, from a given causal graph, whether a set X of variables is independent of another set Y, given a third set Z."

A *collider* is a node characterized by a nonzero in-degree and a null out-degree, i.e. a node with all related edges directed to it.

We need to contemplate d-separation in our reasoning because only by means of this abstraction we can formalize and better understand complex dependence relations and associated complications among variables in a BN, such as *Berkson's Paradox* (explained below in this section).

As reported in [19], "*Dependence* is associated with *Directional Connectedness* [whereas the opposite concept of independence is associated with *Directional Separation*] and is based on three rules:

1. X and Y are *d-connected* if there is an unblocked path [i.e. a path not containing a collider] between them. [. . .]
2. X and Y are *d-connected*, conditioned on a set Z of nodes, if there is a collider-free path between X and Y that traverses no member of Z. If no such path exists, we say that X and y are *d-separated* by Z. We also say then that every path between X and Y is 'blocked' by Z. [. . .]
3. if a collider is a member of the conditioning set Z, or has a descendant in Z, then it *no longer blocks* any path that traces this collider."

Last statement finds an explanation in *Berkson's Paradox* [20]: "when we measure a common effect of two independent causes, these become dependent, because finding the truth of one makes the other less likely (or '*explained away*' [. . .]), and refuting one implies the truth of the other" [19].

As a consequence of first and second d-separation rules, we say that X and Y are d-separated given Z if there is no active trail between X and Y given Z [21]. We denote Z as the *separation set* (SepSet) of X and Y.

3 Evolutionary Computation

Evolutionary Computation uses computational models of natural evolutionary processes as key elements in the design and implementation of computer-based problem solving systems [22].

Evolutionary computation became an increasingly promising machine learning field during the last two decades only, although their theoretical foundations date back to late 1950's.

As pointed out by Vafaee in her PhD thesis [22], "compared to traditional formal approaches, evolutionary search techniques have the key advantages of being flexible and adaptable to the task in hand, in combination with robust performance and global search characteristics. Various evolutionary computational models have been thus far developed and studied, all of which are referred to as Evolutionary Algorithms (EAs)."

In nature, the *genotype* is the "source code" for an individual, encoded in its genome. The *phenotype* encompasses the individual's observable characteristics, acquired as the individual develops. The genotype and the environment both influence the way the phenotype develops.

When two individuals reproduce, their genotypes are copied and recombined to obtain the genotype of their *offspring*: the latter thus inherits characteristics described in the genotype of its parents.

Random mutations occur in the genotype as well: the blind variation caused by genotype recombination and mutation occasionally produces an individual with a superior phenotype;

this individual will be more fit for reproduction, and therefore will produce more offspring.

Any kind of optimization problem can be tackled by means of an evolutionary algorithm: if we have the possibility to encode the problem parameters in some way so to constitute an *individual* and to evaluate an individual's *fitness* when it is needed, then the problem can be addressed by pursuing the same dynamics underlying *natural evolution*.

More precisely, first a *population* of individuals, i.e. potential solutions of the problem, is initialized in a random way; then the population evolves through a number of iterations named *generations*, according to rules of selection and reproduction by means of *genetic operators*.

During *reproduction*, individuals are altered by genetic operators such as *mutation*, that allows the emergence of new genetic information, and *recombination* or *crossover*, that allows the actual reproduction of existing individuals. The *selection* procedure on the other hand favors the propagation of *high-fitness* individuals to later generations.

As the search progresses, an evolutionary algorithm population converges to fitter and fitter individuals: the algorithm stops after a specified number of iterations, when a sufficiently good solution is found, or after the fitness of the population stops improving.

EAs are useful for optimization problems where the search space is non-linear and multimodal, or where the fitness function cannot be differentiated: indeed their stochastic and population-based nature allow them to identify multiple local optima.

4 Research Motivation

Bayesian Network Structure Learning is a task that, if accomplished, would pave the path for Bayesian modeling and would therefore contribute to provide further insight into countless state-of-the-art research topics.

As pointed out by Uusitalo in [23],

> "Bayesian modelling [*sic*] techniques have several features that make them useful in many real-life data analysis and management questions. They provide a natural way to handle missing data, they allow combination of data with domain knowledge, they facilitate learning about causal relationships between variables, they provide a method for avoiding overfitting of data [24], they can show good prediction accuracy even with rather small sample sizes [25], and they can be easily combined with decision analytic tools to aid management [26–28]. On the other hand, their ability to deal with continuous data is limited [28], and such data generally needs to be discretized, which may cause certain difficulties. Bayesian networks are also a useful tool for expert elicitation and combining uncertain knowledge when used with care. Furthermore, building models forces us to think clearly about the subject, and articulate that thinking in the form of the model. This is often beneficial in and of

itself [27,29]."

For what concerns medical and biomedical contexts, available applications include:

- medical diagnosis [30,31];
- pathway modeling [32,33];
- integrative modeling and combinatorial control of RNA alternative splicing [34];
- cellular networks inference [35];
- genetics and phylogeny linkage analysis [36]

Learning the structure of Bayesian Networks may also contribute to conduct research on many topics related to several other domains, including:

- bayesian networks applied to Information Retrieval [37];
- environmental modeling and management [27,38–46];
- text analysis [47];
- evaluation of scientific evidence [48];
- image semantic retrieval and object recognition [49,50];
- setup of an internet security network [51].

The variety of possible applications and BNs generalization capability make the above-mentioned task as ambitious as complex: as a matter of fact, in the literature a plethora of methods for BN structure learning are already provided, although each of them results to be more suitable only for certain applications or only if we constrain the problem itself in terms of network or data sample sizes.

5 Contribution of This Work

Given a well-defined set of nodes and fully observed data, this work aims at providing a method able to learn the structure of the Bayesian network underlying a set of data samples, by focusing on problems with a limited amount of available data, that is often the case of biomedical applications.

In particular, in this thesis work we aim at designing a hybrid BN structure learning algorithm able first to reliably reduce the search space and then to exhaustively explore it as optimally as possible, by taking advantage of data-informed expedients as well.

Aside the first CB phase used to restrict the search landscape, the proposed approaches involve a parameterized Genetic Algorithm in order to pursue the task: this metaheuristic has been chosen because of its efficient global search capabilities even across a very large search space (in this case the space of DAGs extractable from a set of vertices, or rather a related super-structure) and because of its flexibility and adaptability; on the other hand Evolutionary Algorithms are generally characterized by a numerous set of parameters, that have to be carefully tailored with respect to the application in hand in order to make them efficiently explore the search space.

Another primary target we want to achieve is indeed to design a method that is as insensitive as possible to any parameter variation, that is conceivably capable of adapting to a wide range of problems in terms of size and density – for what concerns the Bayesian network itself – and in terms of sample size – for what concerns data, in an as unsupervised as possible fashion.

This work in the first place utilizes Vafaee et al.'s findings in [52, 53] in order to enhance the search phase performed by the genetic algorithm in two ways: by optimally steering search efforts towards some data-driven direction on the basis of [52] and by keeping the balance between exploration and exploitation on the basis of [53]: we will outline more in detail these improvements in Section 5.4.

Moreover, we propose a novel, additional enhancement suitable for any of previously presented genetic strategies in Section 5.5 and a further variation of it in Section 5.6, based on a dynamic, data-informed determination of the parents set of each node of any individual across the evolution, aimed to smartly limit the above-mentioned set and thus allow the application feasibility of presented methods to large networks, address data fragmentation issues and further reduce the search space throughout the evolutionary process.

6 Thesis Outline

The remainder of this thesis is organized as follows:

- Chapter 2 provides an extensive literature overview about Bayesian networks structure learning.
- Chapters 3 and 4 present the theoretical foundations of the two branches underlying the generic approaches to Bayesian network structure learning, i.e. Constraint-Based and Search and Score methods.
- Chapter 5 first gives a brief introduction on Hybrid methods, as well as an overview of the strategies and objectives taken into consideration in the design of our algorithms; then it provides a detour across the progressive enhancements that characterized the creative process underlying the implementation of our methods; in particular this additive process culminates with the research contribution offered in this work, covered by the two final strategies reported in this chapter.
- In Chapter 6 it is reported an exhaustive evaluation of the proposed methods within a rich benchmark composed of a plethora of test cases, jointly with a performance comparison with respect to a variety of other competitor strategies, as well as among the introduced algorithms themselves.
- Chapter 7 summarizes the proposed methods as well as related advantages and disadvantages emerged from experimentation.
- Finally, Chapter 8 outlines the future research directions under the thesis topic.

CHAPTER 2

A DETOUR INTO BAYESIAN NETWORK STRUCTURE LEARNING LITERATURE

This chapter offers an exhaustive literature survey on the topic of Bayesian network structure learning. The collection of methods included in this historical overview is here classified with respect to the standard Bayesian network structure learning taxonomy distinguishing among Constraint-Based (CB), Search and Score (S&S) and Hybrid methods.

1 Constraint-Based Structure Learning

Historically, the first developed methods for bayesian network structure learning were Constraint-Based. The idea is straightforward: an optimal structure among all the possible ones is found by progressively constraining the search space in some way.

1.1 Inductive Causation algorithm

Inductive Causation (IC) method by Pearl [4] is based on *Inferred Causation*, a definition derived from *Occam's Razor* [54]: "a variable X is said to have a *causal influence* on a variable Y if a directed path from X to Y exists in every minimal structure consistent with the data." [4]

At the beginning the graph is initialized as empty. Then, in a first loop, for each pair of vertices, it is driven a search for the minimal set (i.e. the minimum size set that satisfies a condition) S_{ab} so that $a \perp\!\!\!\perp b \mid S_{ab}$; if this set is empty, a and b can be connected with an edge.

In a second loop, for each non adjacent pair (a, b) with a common neighbor c, if $c \notin S_{ab}$ then this means that the compound (a, c, b) is a *v-structure*, i.e. a collider pointed by two other vertices; when a v-structure is identified, the procedure adds arrowheads to c.

As a last step, the built structure is explored in order to orient as many as undirected edges as possible so that vertices are not added nor cycles are created.

At the end of the procedure some edges may be not oriented: indeed this algorithm does not return a DAG but a *Partially Directed Acyclic Graph*, i.e. a PDAG or a *pattern*. Hence, a further method is needed to convert the PDAG in a DAG compatible with the BN.

1.2 Causality Search Algorithm

The *Causality Search* (PC) algorithm by Spirtes [7] is another traditional and basic CB method for BN structure learning.

It is based on four hypotheses, here directly reported from [7].

1. “The set of observed variables is *causally sufficient*.
2. Every unit in the population has the same causal relations among the variables;

3. The distribution of the observed variables is faithful to an acyclic directed graph of the causal structure (in the discrete case) or linearly faithful to such a graph (in the linear case).

4. The statistical decisions required by the algorithms are correct for the population."

The first step in this algorithm is to initialize the complete graph given the V set. Furthermore, for each pair (X, Y) an (initially empty) *Separation Set* $\mathit{SepSet}(X, Y)$ is initialized.

Next a sequence of loops starts, with each loop characterized by an increasing index n, initially 0: for each pair (X, Y) of adjacent nodes so that their neighbors are in equal or higher number to n, all possible combinations of neighboring sets as condition sets are evaluated with CI tests; as soon as a CI test results in independence between X and Y, the edge of the pair is deleted and the separating condition set is recorded in $\mathit{SepSet}(X, Y)$.

This sequence of loops stops when all the sets of neighbors of each pair of nodes is of cardinality lower than n. Then it begins the second part of the algorithm, where edges are directed on the basis of v-structures identification and induction rules.

V-structures identification works exactly as in the IC algorithm: given a triplet of nodes (X, Y, Z), if only one of them is adjacent to both the other two and if it does not belong to the separation set of the other two nodes, then the triplet results to be a v-structure with the node adjacent to the other two being the collider.

Induction rules are then applied until there exists an unoriented edge in the graph; as reported by Pearl in [4]:

1. "Orient $(B—C)$ into $(B \to C)$ whenever there is an arrow $(A \to B)$ such that A and C are nonadjacent.
2. Orient $(A—B)$ into $(A \to B)$ whenever there is a chain $(A \to C \to B)$."

1.3 Three-Phase Dependency Analysis algorithm

The *Three-Phase Dependency Analysis* (TPDA) algorithm, developed by Cheng et al. [55] is an information-theory based approach structured in three phases, i.e. drafting, thickening and thinning, followed by a final orientation for undirected edges.

First *drafting* phase consists in the *Chow-Liu algorithm* [56]: it starts with a null graph, then a mutual information based independence test is executed for each couple of vertices and subsequently all nodes pairs are ordered in a list with respect to the dependence value, in decreasing order; for each pair (X, Y) of nodes in the list, if there does not exist an adjacency path between X and Y then edge $(X - Y)$ is added to the graph and pair (X, Y) is removed from the list.

Second *thickening* phase checks whether an edge is needed by examining all remaining edges in the list and, if that is the case, it adds the edge.

Third *thinning* phase, given each edge in the built graph, first verifies whether there exists at least one path connecting the two vertices of the edge, besides the edge itself; if at least an alternative path exists, then the edge is temporarily removed and its need is checked: if the edge is not needed the temporary removal is confirmed, otherwise the edge is reinserted in the graph.

Lastly all edges are directed by means of standard orientation rules inherited by Pearl's method [4].

1.4 Grow-Shrink algorithm

The *Grow-Shrink* (GS) algorithm [57] operates in a local fashion: it first identifies the local neighborhood of each variable in the BN, then applies a series of conditional independence test by taking into account a vertex neighborhood per time.

The *Markov Blanket* of a node X, $Mb(X)$, is the set of parents, children and spouses (children's parents) of X [58]: it can be found by grouping all variables that result (one per time) in a dependence relationship with X, conditioned on all remaining vertices.

This algorithm is based on the *Total Conditioning* property: given a faithful causal graph, each parent, child or spouse of a node stores information about that node that cannot be obtained from any other variable.

First the Markov Blanket of each node is calculated, and the graph is converted to its *moral graph*, i.e. its undirected version with all spouses into each Markov Blanket linked together by additional edges.

One node per time, a CI test between it and each vertex in its Markov Blanket conditioned on all possible subsets of the remaining portion of the Markov Blanket is executed: in case it results independence, the edge between the two vertices taken into account is deleted.

What follows next is the orientation phase: for each vertex X and every vertex $Y \in Mb(X)$, all neighbors Z of X but not of Y are considered; for each neighbor the algorithm tries to orient Y towards X: if it results that there exists a Z so that Y and Z are conditionally dependent given all combinations of their markov blankets, then orientation is confirmed, otherwise it is removed.

1.5 Recursive Autonomy Identification algorithm

A more recent, recursive method is given by the *Recursive Autonomy Identification* algorithm, developed by Yehezkel et al. [59]; this approach attempts to tackle the problem when few data are available (although sample dataset should be large enough to ensure sufficiently reliable CI tests).

An important concept which this method deals with is the *d-separation resolution*: it is defined between a pair of two non-adjacent nodes as the size of the smallest condition set that d-separates them; furthermore, *d-separation resolution of a graph* is defined as the highest d-separation resolution within it [59].

The graph $\mathcal{G}$ is decomposed through the identification of substructures and exogenous causes; a node Y is an *exogenous cause* to a subgraph $\mathcal{G}' \in \mathcal{G}$ if Y is not within $\mathcal{G}'$ and if Y is a parent or a neighbor of X, for all vertices X in the substructure.

A substructure $\mathcal{G}^A(V^A, E^A) \in \mathcal{G}$ is defined to be *autonomous* in $\mathcal{G}$ given a set $V_ex \in V$ of exogenous causes to $\mathcal{G}^A$ if all vertices' parents in the substructures are in the substructure itself or in the exogenous causes set; moreover, if all parents are within the substructure, the latter is named a *completely autonomous* substructure [59].

A fundamental principle which this algorithm is based on is that if two variables are independent within a substructure, then they are independent in the whole graph as well.

This method needs also to assume two hypotheses, i.e. that a DAG can encode all the independences entailed from given data and also that data sample size is large enough for reliable CI tests.

Recursion occurs with respect to n, the d-separation resolution; at each recursive iteration three actors are involved: $\mathcal{G}_{start}$, with a d-separation resolution of $n-1$, $\mathcal{G}_{ex}$, constituting a set of structures, each having possible exogenous causes to $\mathcal{G}_{start}$, and $\mathcal{G}_{all}$, that contains $\mathcal{G}_{start}$, $\mathcal{G}_{ex}$ and edges connecting them.

At initialization phase n is set to 0 and $\mathcal{G}_{start}$ and $\mathcal{G}_{all}$ are initialized as the complete undirected graph.

As a first step, in the recursive function it checks the *exit condition*: if all nodes in V_{start} have a number of potential parents lower than $n + 1$ then the branch reached the end of its recursive path and $\mathcal{G}_{all}$ is returned as output graph.

If the exit condition is not undertaken, here it is executed the phase involving the *thinning* of the link between $\mathcal{G}_{ex}$ and $\mathcal{G}_{start}$: for every edge between $\mathcal{G}_{ex}$ and $\mathcal{G}_{start}$, if there exists a condition set sized n that d-separates the two vertices of the edge, then the latter can be removed from $\mathcal{G}_{all}$; then it follows a possible orientation of $\mathcal{G}_{start}$ edges by means of standard orientation rules.

A similar step to previous one is then executed: now the focus is on $\mathcal{G}_{start}$, that is *thinned, directed* and *decomposed.* Now all edges contained in $\mathcal{G}_{start}$ are taken into account, tested for CI and possibly removed; next edges that can be directed are oriented; finally the *decomposition* occurs: first lowest topological order nodes are grouped into a *descendant substructure* $\mathcal{G}_{\mathcal{D}}$, then all unconnected structures resulting from $\mathcal{G}_{all} \backslash \mathcal{G}_{\mathcal{D}}$ are defined as *ancestors* $\mathcal{G}_{A1}, \ldots, \mathcal{G}_{Ak}$.

Recursion is then triggered, first for all ancestor substructures (with $\mathcal{G}'_{ex} = \mathcal{G}_{ex}$) and then for the descendant one (with $\mathcal{G}'_{ex} = \{\mathcal{G}_{A1}, \ldots, \mathcal{G}_{Ak}\}$); then the recursive function returns.

An advantage of this method is that graph decomposition decreases dependence on *node ordering*, because it is not arbitrary as it happens for instance in *PC* algorithm.

On the other hand, in general the algorithm returns a partially directed acyclic graph, so an external procedure to orient remaining undirected edges is required.

Authors refer also to the possibility of an *interrupted learning* approach, where the stop condition is enhanced with the reaching of some d-separation resolution order: in this way what we get is a PDAG with more undirected edges with respect the base case, but also a *more reliable* graph.

1.6 Opt01SS algorithm

An efficient state-of-the-art CB-based method is the Opt01SS algorithm [60]: it learns super-structures using only $0^{th}-$ and $1^{st}-$order CI tests in a way that takes into account the presence of approximate-deterministic relationships and inconsistent CIs, commonly found in data scarcity contexts.

A *super-structure* is an undirected graph assumed to contain all true edges [of the target BN structure $\mathcal{G}$] [59].

A *sound* super-structure of $\mathcal{G}$ is any PDAG $\mathcal{S}$ that contains the skeleton of $\mathcal{G}$; a super-structure that is not sound is said to be *incomplete* [59].

Opt01SS aims at tackling two problems that are especially relevant when sample is *small*: presence of approximate deterministic relationships and inconsistency in CI testing.

An *Approximate Deterministic Relationship* (ADR) is a “fortuitous” strong association between two variables, related to the fact that a consistently large portion of data exhibits by accident a *deterministic relation* for those variables.

Inconsistent Conditional Independence and Dependence (CIDS) statements are occurrences "that cannot be simultaneously represented on a perfect map" [60]; this problem is caused by *false detected CIs*: they commonly lead to edges removal, even if they should not be removed.

Both presented problems may lead to a wrong choice for a pair of nodes' *separator*, i.e. the vertex that actually d-separates the two nodes in the pair: strength of this algorithm resides exactly in its ability to identify the correct separator, even after a wrong choice.

At initialization phase the super-structure, from now on named SS, is initialized as the complete undirected graph underlying the whole set of vertices; a cache $\mathcal{C}_{<X,Y>}$ containing a slot for each pair of nodes is also allocated; the cache, when a query is asked, returns the (last) separator node recorded for the pair of nodes taken into account.

First step is given by 0^{th}-*order CI tests*: for all edges in SS, if it results unconditional independence between the nodes of the edge then the latter is deleted from SS and the cache entry for the edge is updated with an empty set.

Next it begins the 1^{st}-*order CI tests* phase: the set of edges to check, i.e. $E2Check$ is initialized as the set of edges in SS remained after first step of the algorithm; then a loop starts: it will stop only when $E2Check = \emptyset$.

All edges in $E2Check$ so that both vertices in the edge have more than one only neighbor are then considered one per time in an inner loop, where three sets are built: first set contains neighbors of both vertices whereas the second and the third ones include only the neighbors to one vertex but not to the other one; moreover the vertices pair's separator Z (if any) is retrieved from the cache. The same looping procedure, given the edge under analysis, is then executed for each of the three above-mentioned sets: if the CI test, conditioned on each vertex in the set (but excluding the last separator Z retrieved from the cache), succeeds, then the edge is deleted from SS and inserted into the set of deleted edges, i.e. $EDel$, and the procedure continues with the next edge in $E2Check$.

After a preliminary check on all 1^{st}-order CI tests, the *separators robustness verification* phase starts: $E2Check$ is set as empty and all edges in $EDel$ are taken into account in a second inner loop; if the last separator recorded in the cache for the current edge is not a neighbor of any of the two vertices in the edge, then an ADR is identified: the edge is thus reinserted in the $E2Check$ set.

The final step is needed to *solve inconsistent non-edges*: here the *necessary path condition* [61] is exploited in order to restore inconsistently deleted edges. First the set *ERestored* is initialized as empty, then, for each pair of vertices not in SS so that their cache entry contains a separator, if there does not exist at least one path connecting each vertex with the connector that does not include the other vertex, then the absent edge is considered to be inconsistent and it is added to *ERestored*; finally all edges in *ERestored* are added to SS.

2 Search and Score Structure Learning

Score-based learning is another canonical technique useful to identify the optimal structure of a BN: its core resides in a scoring function used to estimate the goodness of fit of a structure to the data, whereas its goal is to find the highest fitting structure.

As Liu et al. pointed out in [62], "solving the learning problem exactly becomes impractical if the number of variables is too large. Consequently, much early work focused on approximate algorithms, such as greedy hill climbing approaches [63, 64], tabu search with random restarts [65], limiting the number of parents or parameters for each variable [66], searching in the space of equivalence classes of network structures [67] and the optimal reinsertion algorithm (OR) [68]. These algorithms use local search to find 'good' networks; however, they offer no guarantee to find the one that globally optimizes the scoring function.

Recently, exact algorithms for learning optimal BNs have been developed based on dynamic programming [69–73], branch and bound [74], linear and integer programming [75, 76] and heuristic search [77–79]."

Another recent S&S-based approach is given by evolutionary computation methods such as genetic algorithms.

In Section 1 it is provided a review on most known scoring methods in the literature.

2.1 K2 Algorithm

The popular *K2* algorithm by Cooper and Herskovits [63] uses a greedy search method and does not necessarily need an upper bound on the number of parents a node can have, even if a maximum indegree is requested as input.

The K2 search begins by assuming that a node has no parents and then greedily selects as its parents the variables from a given ordering whose addition best improves as much as possible the score of the resulting structure in an incremental fashion, until the score stops to increase.

The scoring function employed in this heuristic is the *K2 metric*, described more in detail in Section 1; one advantage of this approach is that K2 score prefers *simpler structures* [80].

On the other hand, a limitation to this method is that BN evaluation depends on the choice of a node ordering, needed as input to the algorithm.

2.2 Maximum Weight Spanning Tree algorithm

The *Chow-Liu [56] Maximum Weight Spanning Tree* (MWST) algorithm was among the first S&S methods for BN structure learning; its goal is to find a tree that maximizes the data likelihood.

First step is to compute a *weight* for each possible edge of the graph: in other words, each edge is provided with a score associated with data, defined by the *mutual information function.*

Then the algorithm attempts to find a maximum weight spanning tree, i.e. a tree with the greatest total weight that reaches all nodes; one common practice is to follow a greedy approach, i.e. to greedily add edges making sure the structure is a tree at every step. Standard approaches for tree construction are used, such as Kruskal or Prim algorithms.

Last step consists in edges orientation: it can be done after the scoring step because the weighting method assigns a score to each edge regardless of its orientation.

A limitation of this approach is the need to choose a root node.

2.3 Tree-Augmented Naïve Bayes algorithm

This algorithm, written by Friedman et al. [81] is an extension to the MWST algorithm, specialized in building a *tree augmented network* for a given *class node.*

The first step determines the weight of each edge of the network, on the basis of a conditional mutual independence scoring method, conditioned on the class node.

Remainder of the algorithm is the same as in MWST, with the difference that it will return a specialized tree with a class node as parent of all the remaining nodes.

A tree built by this method has one node with no parent (class node), one node with only the class node as parent (root) and all the other nodes with two parents (class node and some other node in the tree).

An advantage given by this algorithm is search space reduction: the set of parents of each variable is restricted to a small subset of candidates. On the other hand its behavior depends on the choice of equivalence class and root nodes.

2.4 Hill Climbing algorithm

The *Hill Climbing* (HC) algorithm, developed by Buntine in 1994 [82], is a *local* method for BN structure learning, in the sense that at each step the algorithm considers all available local operations and chooses the one that yields the best improvement.

This method starts by taking as input a dataset defined over V and a DAG defined over V, usually the empty graph. At each step the algorithm computes the differences in the overall score with respect to all possible local arc operations, i.e. *addition*, *deletion* and *reversal*, and chooses the one with the highest positive difference; the process is repeated until score improvement occurs.

2.5 Sparse-Candidate algorithm

In 1999 Friedman et al. [66] proposed the *Sparse-Candidate* algorithm: it constitutes a way to accelerate learning BN structures from data sets with many variables.

First the set of possible parents (candidates) for each nodes is restricted by means of several metrics, then a BN is learnt on the basis of these restrictions, through a traditional HC algorithm; after that the obtained network is used to update the sets of candidate parents, the entire procedure is iterated.

The main advantage of this algorithm with respect to traditional HC method is the speed enhancement on large data sets.

2.6 Optimal Reinsertion algorithm

The *Optimal Reinsertion* (OR) algorithm, introduced by Moore and Wong [68] works along the following lines: at each step a *target node* is chosen, all edges entering or leaving the target are deleted, and after that the optimal combination of in- and out-edges is found the node is re-inserted in the network with these edges.

The Sparse-Candidate enhanced version of this algorithm is characterized to be faster and more suitable to large networks. This algorithm is especially useful on large data sets and resulted to be faster and more performing than HC, in particular when the search space has many local minima.

2.7 Greedy Equivalent Search algorithm

The *Greedy Equivalent Search* was developed by Chickering [9] in 2002.

The graph is initialized with no edges, then it follows the forward phase and the backward phase.

During the *forward phase*, at each step it is considered the *essential graph* related to current PDAG instantiation, and it is added the edge that allows to get the highest possible score for the newly obtained essential graph; this process is repeated until it is not possible to improve the score anymore.

In the *backward phase* the essential graph related to current PDAG instantiation is considered once again: by taking into account all DAGs formed from the current PDAG after the edge deletion that leads to the highest possible scoring PDAG, if one of them has a higher score than all possible instantiations of current essential graph, then the deletion is confirmed; the process is repeated until score improves.

2.8 Ordering Search algorithm

In 2005, Teyssier and Koller [83] presented another learning method that searches over the space of node orderings.

It consists in a greedy hill climber with tabu lists and random restarts and resulted to be competitive with more complex algorithms; its efficiency is explained by the same main advantage of K2 algorithm, i.e. the best network can be found very efficiently, with a given node ordering. Indeed the orderings search space “is much smaller, makes more global search steps, has a lower branching factor and avoids costly acyclicity checks” [83].

2.9 Markov Chain Monte Carlo methods

Markov chain Monte Carlo (MCMC) methods are a class of algorithms for sampling from a probability distribution based on constructing a Markov chain that has the desired distribution as its equilibrium distribution [84]; they operate by generating each sample on the basis of a *random change* to the preceding sample.

A consistent amount of work in BN structure learning literature (e.g. [85–87]) involves the *Metropolis-Hastings* strategy, that belongs to the category of *Random Walk Monte Carlo* methods.

The Metropolis-Hastings algorithm allows to sample any probability distribution $P(x)$ given a known function $f(x)$ which is proportional to P density; in this context, the probability distribution is given, in the simplest implementation, by the space of DAGs constituting all possible instances of the target BN, whereas the known function is the scoring metric.

At each iteration, the algorithm uses some heuristic to select a candidate to sample next value on the basis of current one (Markov Property): if the candidate is more fit than starting sample then it is accepted, otherwise old sample is restored.

In particular, the candidate selection heuristic must be based on a *symmetric* distribution, i.e. the probability of transitioning from sample x to sample x' is the same of the opposite transition. A usual choice is to let the selection probability density be a *gaussian distribution* centered at current sample, making the sequence of samples into a *random walk*.

Moreover, Gibbs sampling methods can be used to implement algorithms able to work with incomplete data, as in Antal et al. [87].

2.10 Simulated Annealing methods

Several *Simulated Annealing* approaches are present in the literature for what concerns BN structure learning [85, 88–90]: they are all based on the same principle of simulated annealing, a general method for solving unconstrained and bound-constrained optimization problems; specifically, it is a metaheuristic useful to approximate a large search space.

Simulated annealing [91] interprets slow cooling as a slow decrease in the probability of accepting worse solutions as it explores the solution space, with the advantage of a more extensive search.

The algorithm presented by Carrillo et al. [85] constitutes an application of simulated annealing to BN structure learning with Bayesian score (described in Section 1) as a measure of goodness, focused on small datasets.

This method, although relatively faster with respect to other metaheuristics such as genetic algorithms, requires a predefined ordering on the nodes, and results sensitive to it as well.

First step consists in computing mutual information distribution for each pair of vertices following a particular pattern: each computed value $MI(x_i, x_j)$ is added at the corresponding position of a $n \times n$ matrix ($n = |V|$) to the sum of previous pair $MI(x_{i-1}, x_j)$: as a result, each matrix column will contain a distribution of MI of the variable X_j.

In the simulated annealing step, all possible parents for the current node are eligible to be chosen on the basis of their associated probability, defined as their MI value normalized with respect to all nodes. According to this distribution, nodes characterized by the highest MI with current node are more likely to be selected as its parents; when a parent node is selected, if the arc from it to the current node is not already included in the structure and if the node's fan-in is not already at the maximum allowed value, then it is added, otherwise it is not inserted.

Once the neighbor configuration of parents is obtained it is scored: the configuration is accepted and will replace previous one only if its evaluation yields a better score, on the basis of Metropolis-Hastings approach. The process is then repeated until no more improvement on the score can be obtained.

This procedure is iterated for each node: after all nodes have been analyzed, the constructed BN structure is deprived of redundant arcs and then it is returned as the best found structure.

2.11 Evolutionary Algorithms

Among the various application fields, genetic algorithms, categorized as a branch of evolutionary algorithms, have been also previously used for BN structure learning, in S&S as well as in Hybrid methods [15, 92–97].

2.11.1 Larrañaga algorithm

In 1996, Larrañaga et al. [92] proposed one of the first GAs for learning BN structure: individuals are given by adjacency matrices (a possible representation for graphs) and the LL score, presented in 1 has been used as fitness function.

Their mutation and crossover operators are able to always generate a valid DAG, with a given node ordering; they also tackled the learning problem without having a node ordering available, by providing the procedure with a *repair operator* which randomly removes edges within cycles until the DAG property is satisfied. They also restrict the maximum number of parents for any node to four.

2.11.2 K2GA algorithm

This method, developed by Larrañaga et al. [93], is basically a GA enhancement of the K2 algorithm.

The K2 method can generate a BN from a dataset once a node ordering is given: the GA is indeed used to search for a near-optimal ordering between the variables, with the K2 score (described in Section 1) serving as fitness function.

The authors compared numerous crossover and mutation operators that were previously used for the *Traveling-Salesman* problem. One limitation bounded to the employment of the

K2 algorithm is given to the need to set a maximum fan-in for each node.

2.11.3 Chain-Model Genetic algorithm

To reduce the time complexity of K2GA [93], Kabli et al. [95] proposed a *Chain-Model GA* which attempts to evaluate and thus select node orderings by relying on chain structures.

The base hypothesis in this approach is that node ordering mainly determines the score for a given network structure: in other words the final K2 score on the overall structure, given a node ordering, is assumed to be directly proportional to the K2 score evaluated on the simple *chain structure* obtained given the same ordering.

Given that each individual's node has at most one parent only, fitness evaluation during evolution is much faster with respect to standard K2GA.

Analogously to K2GA, also this method is dependent to the maximum fan-in parameter.

2.11.4 Carvalho's Cooperative Coevolution Genetic Algorithm

In 2011, Carvalho et al. [96] proposed a cooperative-coevolution GA for learning BN structures. Here two independent subpopulations are considered, the *permutation* species, representing a node ordering, and the *binary* species, that represent actual DAGs, or rather upper-triangular adjacency matrices constituting BN graphs, given a node ordering.

The authors used cycle crossover and swap mutation for the permutation subpopulation, and two-point crossover and bit-flip mutation for the binary subpopulation. This method allows no restrictions on the number of parents a node can have.

2.11.5 μGP algorithm

μGP is an EA software developed by the CAD Group of Politecnico di Torino [98]: provided with its capability of encoding individuals as tagged graphs, it was used by Tonda et al. [97] to learn BN structures as an S&S technique.

The individual representation has two parts: a DAG and a node ordering; arcs are generated only from one node to nodes that follow it in the order, so to avoid loops by construction.

This method deals with a maximum number of parents per variable.

3 Constraint-Based – Search and Score Hybrid Structure Learning

Latest advances in Bayesian Network Structure Learning involve the adoption of a *Hybrid* approach that attempts to take the best of both CB and S&S worlds, in which first a CB method is applied in order to constrain and reduce the search space, and then a S&S method is used so to identify the possibly optimal structure in the remaining portion of the search space.

3.1 CB algorithm

In 1995, Singh and Valtorta [99] presented their CB algorithm, an iterative method for learning BN structures.

It is articulated in several phases: first CI tests are used to restrict the search space and create an undirected graph of the variables; then some edges orienting heuristics are applied, so to obtain an ordering of the nodes; finally K2 is executed using the obtained node ordering. This process repeats until K2 score stops improving.

3.2 Wong's Cooperative Coevolution Genetic Algorithm

In 2004 Wong et al. [94] presented their Hybrid *Cooperative-Coevolution Genetic Algorithm* (CCGA) for learning BN structures.

Starting from the complete DAG, in the first phase they apply all possible 0^{th}- and 1^{st}-order CI tests so to possibly exclude from consideration certain edges, thereby reducing the search space.

In the second phase they use a collaborative-coevolution algorithm that splits the structure-learning problem into a set of subproblems, one per variable: each subproblem aims at learning a set of parents for each node, and constitutes a single, independent population to evolve, with each row of the network's adjacency matrix being the string representation of the individual.

In order to reduce the chance of creating cyclic structures they adopted an approximate ordering on the nodes.

Wong et al. reported that their algorithm outperforms the previous *Minimum Description Length and Evolutionary Programming* (MDLEP) algorithm of their own.

3.3 Hybrid Structure Learner using Genetic Algorithm

In 2014 Vafaee [15] proposed her *Hybrid Structure Learner using Genetic Algorithm* (HSL-GA), focused in particular on problems involving networks of medium to large size and a limited dataset.

This Hybrid strategy first builds a *super-structure*, i.e. an undirected *super-graph* that is possibly likely to contain the target DAG: 0^{th}-order CI tests are performed in a constraint-based fashion in order to get a reliable super-structure; this is done so to deal with a reduced search space.

At the second stage, the GA procedure attempts to pick the highest-scoring DAG among all possible subgraphs of the original super-structure. An individual is built as a ternary string of loci, where each locus represents an edge that was present in the starting super-graph and that can assume three possible states during evolution: orientation in one of the two possible directions or absence.

The used approach is *elitist*: indeed during selection the best individual is retained and will be propagated to next generation, in the case it will still be the most performing one. The best-fitting part of population is then selected to undergo the reproduction phase.

This method involves a conventional uniform mutation and single-point crossover; moreover a Minimum Feedback Arc Set removal function is applied to each individual after reproduction phase, so to ensure the validity of the reproduced DAGs population.

Since the goal of Vafaee's work was to get an as close as possible representation of reality she decided to evaluate performance by means of matching accuracy, i.e. F_1 score [100].

Vafaee reported that HSL-GA outperforms a set of other relevant BN structure learning methods (including MWST and K2) regardless of data size, for sample sizes varying from 30 to 100.

CHAPTER 3

CONSTRAINT-BASED STRUCTURE LEARNING

Constraint-Based (CB) algorithms rely on a number of Conditional Independence tests to determine, given a conditioning set, whether two variables are independent.

They usually construct DAGs in two stages: first by learning the direct dependences between variables so to produce an undirected graph structure, then by directing the edges through the employment of orientation rules. The first stage requires a number of CI tests that grows exponentially with the number of variables, thing that makes CB methods adopt some heuristics to restrict the size of the condition set [15].

Possible constraints may be conditional independence or structure-based statements, but the latter case is applicable only in certain cases where *latent variables* are taken into account [8]; thus, since in this thesis work we are only interested in domains without missing values or latent variables, the constraints we take into account are only *conditional independence* statements.

1 Conditional Independence Test

CI testing is commonly implemented through a metric estimating statistical independence between variables, such as Pearson's χ^2, likelihood-ratio G^2-test or by thresholding Conditional

Mutual Information.

1.1 Mutual Information

Mutual Information between variables X and Y is described by the *Kullback-Leibner Divergence* between the joint distribution $\Pr(X, Y)$ and the product of single distributions $\Pr(X)\Pr(Y)$ [101]:

$$\mathcal{MI}(X;Y) = \sum_{x \in X, y \in Y} \Pr(x,y) \log \frac{\Pr(x,y)}{\Pr(x)\Pr(y)} \tag{3.1}$$

It measures the amount of information shared between the variables X and Y: if this quantity is considered to be negligible, i.e. below some threshold, then the two variables can be considered independent [102].

As reported in [103], the Conditional Mutual Information between X and Y measures the information flow between X and Y given a *conditioning set* $\boldsymbol{S}$:

$$\mathcal{CMI}(X;Y|\boldsymbol{S}) = \sum_{x \in X, y \in Y, s \in \boldsymbol{S}} \Pr(x,y,s) \log \frac{\Pr(x,y|s)}{\Pr(x|s)\Pr(y|s)} \tag{3.2}$$

1.2 Pearson's χ^2

The standard, most popular CI test is *Pearson's* χ^2 *test for statistical significance* [104]: it is a *statistical hypothesis test*, i.e. a method that proves a hypothesis by observing a process modeled via a set of random variables, where the *sampling distribution* of the test statistic is a χ^2 *distribution* when the null hypothesis is true.

As stated by Pearson in [104], “the quantity

$$\chi = \sqrt{S\Big(\frac{e^2}{m}\Big)} \tag{3.3}$$

is a measure of the goodness of fit [of data to the model given by the distribution, where m is the vector, with each element related to a random variable, of the theoretical frequencies supposed known a priori, e is the error vector, with each element related to a random variable, between the observed frequencies and the theoretical frequencies and with S indicating a sum over all the elements of the resulting vector].”

In this case this test is used to reject the hypothesis that data and related random variables are *independent.*

1.3 G^2 Likelihood-ratio

A theoretical generalization to Pearson’s χ^2 test is given by the G^2 *likelihood-ratio test* [105], that formally correlates likelihood ratio to Pearson’s test.

By quoting the words of McDonald in [105], “the G-test uses the log of the ratio of two likelihoods as the test statistic, which is why it is also called a likelihood ratio test or log-likelihood ratio test. [...] The equation is

$$2\sum\left(O \cdot \ln\Big(\frac{O}{E}\Big)\right), \tag{3.4}$$

[where O constitutes the vector of observed values and E the vector of expected values relatively to the set of random variables.]"

As MacKay and Sokal et al. reported in [106, 107]: "in general, with smaller amounts of data, the chi-squared test will sometimes give incorrect answers, whereas the G-test will not, and so is the recommended test."

Both χ^2 and G^2 statistical tests return a value: if it is lower than the critical value, i.e. some predefined threshold, then the null hypothesis cannot be rejected and involved variables are considered to be (conditionally) independent.

CHAPTER 4

SEARCH AND SCORE STRUCTURE LEARNING

Search and Score (S&S) methods involve a strategy for searching through the space of possible structures and a scoring function able to estimate the fitness of each structure to the data. It works as any test-and-set approach: in this particular case the goal is to maximize the score at each step by transitioning from one structure to another through some local variation operators (e.g., edge deletion or addition), and iterating this process until score stops improving or a stop condition is met [15].

When tackled in this way, the task can be formalized as a *static optimization problem* of the type

$$\max f(x) | x \in \{\ \rightarrow,\ \leftarrow,\ \not\rightarrow\ \}^l, \tag{4.1}$$

where x is a ternary string of length l (i.e., a solution) and $f(x) \in \mathbb{R}^+$ is its score.

1 Scoring Function

The core of a S&S method is given by the employed *scoring metrics*, usually categorized into Bayesian and information-theoretic scoring functions.

A *score* is said to be *decomposable* if it can be written as the sum or the product of functions that depend only of one vertex and its parents [108], as in Equation 4.2.

$$S(B) = \sum_{i=1}^{n} s(X_i, Pa(X_i)) \qquad \text{or} \qquad S(B) = \prod_{i=1}^{n} s(X_i, Pa(X_i)) \tag{4.2}$$

An interesting property for scoring functions is score equivalence: a scoring function ϕ is said to be *score equivalent* if it assigns the same score to all DAGs that are represented by the same essential graph [109].

1.1 Bayesian Scoring Functions

The general idea of Bayesian scoring functions is to compute the posterior probability distribution, starting from a prior probability distribution on the possible networks, conditioned to data D, i.e. $\Pr(\mathcal{B}|D)$. The best network is the one that maximizes the posterior probability.

$$\Pr(\mathcal{B}|D) = \frac{\Pr(D|\mathcal{B})}{\Pr(D)} \tag{4.3}$$

Since the term $\Pr(D)$ is the same for all possible networks, in practice, given that we deal with comparisons, computing $\Pr(\mathcal{B}|D)$ is sufficient. Moreover, as it is easier to work in the logarithmic space, the scoring functions use the value $\log \Pr(\mathcal{B}|D)$ instead of $\Pr(\mathcal{B}|D)$ [109].

1.1.1 Bayesian Dirichlet score

Heckerman et al. [24] proposed this score function on the basis of four assumptions on $\Pr(\mathcal{B}|D)$:

1. data D is *exchangeable*, i.e. if an instance of the data is replaced with another instance, exchanged data has the same probability as the original one;
2. parameters Θ related to $\mathcal{B}$ have a Dirichlet distribution, i.e. the probability density function for Θ_{ij} is given by
$$\rho(\Theta_{ij}|\mathcal{G}) = c\prod_{k=1}^{r_i}\theta_{ijk}^{N'_{ijk}-1}$$
with $N'_{ijk} > 0$, where $N'_{ijk_{k=1\ldots r_i}}$ are the hyperparameters of the Dirichlet distribution;
3. the parameters associated with each variable in the network are independent;
4. the parameters associated with each possible parents combination of a variable are also independent.

The *Bayesian Dirichlet* (BD) score, derived from Heckerman, Geiger and Chickering theorem [24] is defined as

$$\Pr(\mathcal{B}, D) = \log(\Pr(B)) + \sum_{i=1}^{n}\sum_{j=1}^{q_i}\left(\log\left(\frac{\Gamma(N'_{ij})}{\Gamma(N_{ij}+N'_{ij})}\right) + \sum_{k=1}^{r_i}\log\left(\frac{\Gamma(N_{ijk}+N'_{ijk})}{\Gamma(N'_{ijk})}\right)\right) \tag{4.4}$$

Unfortunately, as Heckerman et al. recognized, specifying all N'_{ijk} for all i, j and k is unmanageable in realistic situations: this makes the BD score unusable in practice.

Regarding the term $\log(\Pr(\mathcal{B}))$, which appears in BD and all expressions derived from it (K2, BDe, BDeu) in general it is assumed a *uniform distribution* except if, for some reason, some structure is strongly preferred. In case of a uniform distribution, this term becomes a constant and can be removed.

1.1.2 K2 score

Cooper and Herskovits [63] proposed one of the first usable Bayesian scoring functions. The *K2* metric is a particular, simplified case of the BD score, where it is simply applied the assignment ($N'_{ijk} = 1$), corresponding to zero pseudo-counts.

1.1.3 BDe score

The BD metric involves a parameter that depends on all possible combinations among a variable and its parents; furthermore it is not *score equivalent.*

Heckerman et al. [24] addressed this issue by developing a score equivalent version of BD, called *likelihood-equivalence Bayesian Dirichlet* (BDe): given a single hyperparameter called the *equivalent sample size*, referred to as α, and a prior distribution over network structures, the BDe metric is able to score a DAG with respect to a sample dataset; its expression is identical to the BD equation.

This scoring approach is based on two further assumptions, besides BD hypotheses:

Likelihood equivalence two equivalent DAGs cannot be discriminated by means of the set of parameters Θ_D extracted from data D;

Structure possibility the probability of any complete DAG is *nonzero.*

Similarly to the BD score, the BDe metric requires knowing $\Pr(X_i = x_{ik}, \Pi_{X_i} = w_{ij}|G)$ for all i,j and k: since this knowledge might not be elementary to find, this score is of little practical interest [109].

1.1.4 BDeu score

A particular case of BDe is given when the prior network assigns a uniform probability to each configuration of the set of variables and priors distributions:

$$\Pr(X_i = x_{ik}, \Pi_{X_i} = w_{ij}|G) = \frac{1}{r_i q_i}. \tag{4.5}$$

The resulting score is called *uniform joint distribution likelihood-equivalence bayesian Dirichlet*, BDeu, and was originally proposed by Buntine [110]:

$$\Pr(\mathcal{B}, D) = \log(\Pr(B)) + \sum_{i=1}^{n}\sum_{j=1}^{q_i}\left(\log\left(\frac{\Gamma(\frac{\alpha}{q_i})}{\Gamma(N_{ij} + \frac{\alpha}{q_i})}\right) + \sum_{k=1}^{r_i}\log\left(\frac{\Gamma(N_{ijk} + \frac{\alpha}{r_i q_i})}{\Gamma(\frac{\alpha}{r_i q_i})}\right)\right) \tag{4.6}$$

This score only depends on one parameter, the *equivalent sample size* α: it expresses the strength of our prior belief in the uniformity of the conditional distributions of the network [109].

Since this score is very sensitive with respect to the above-mentioned hyperparameter, that is directly related to the *density of the network to be learnt* [62], several values are commonly attempted. As Liu et al. claimed in [62], "if the density of the network to be learned is unknown, selecting an appropriate α is difficult."

1.2 Information-Theoretic Scoring Functions

Information-theoretic metrics are based on compression. In this context, the score of a Bayesian network B is related to the compression that can be achieved when we try to describe data D with $\mathcal{B}$.

Shannon's source coding theorem establishes a theoretical lower bound to lossless data compression, as well as Shannon entropy operational meaning.

Theorem 2 (Shannon's Source Coding Theorem [111]) *As the number of instances of an i.i.d. (independent and identically-distributed) data stream tends to infinity, no compression of the data is possible into a shorter message length than the total Shannon entropy, without losing information.*

Given data D, it is possible to score a BN $\mathcal{B}$ by the size of an optimal code, induced by $\mathcal{B}$, when encoding D. This value is the *information content of D by $\mathcal{B}$* and is given by:

$$L(D|\mathcal{B}) = -\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log(\theta_{ijk}) \tag{4.7}$$

On the basis of *Gibbs inequality*, the equation above is minimized when the Bayesian network that induces a code that compresses D the most is precisely the Bayesian network that maximizes the probability of observing D.

1.2.1 Log-Likelihood score

By applying a logarithm to $L(D|\mathcal{B})$ we obtain the *log-likelihood* (LL) of D given $\mathcal{B}$: maximizing the log-likelihood is equivalent to minimizing the information content of D by $\mathcal{B}$ [109].

$$LL(\mathcal{B}|D) = -\sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log(\frac{N_{ijk}}{N_{ij}}) \tag{4.8}$$

This score cannot be used as it is, since it does not follow the basic rule of *Occam's razor*: indeed it tends to favor complete network structures without taking into account independence assumptions of the learned network.

This problem is commonly overcome by using some structure-related penalization function $f(N)$ over the LL score, of the type:

$$\phi(\mathcal{B}|D) = LL(\mathcal{B}|D) - f(N)|B|, \tag{4.9}$$

where $|B|$ denotes the *network complexity* [112], that is, the number of parameters in Θ for the network $\mathcal{B}$.

1.2.2 Minimum Description Length score

Minimum Description Length is an *Occam's razor* approach (the simplest explanation is the best) to fitting, rigorously defined as:

$$MDL(\mathcal{B}|D) = LL(B|D) - \frac{1}{2}\log(N)\sum_{i=1}^{n}(r_i - 1)q_i \tag{4.10}$$

where the second term represents the *network complexity*, i.e. the numbers of parameters in Θ for the network $\mathcal{B}$; each parameter is weighted by $\frac{1}{2}\log(N)$ bits. This score coincides with the *Bayesian Information Criterion* (BIC) score based on *Schwarz Information Criterion*, with the difference that the latter "is derived based on the asymptotic behavior of the models, that is, BIC is based on having a sufficiently large amount of data" [62].

1.2.3 Akaike Information Criterion score

The AIC metric [113] is a derivation of MDL score: here we have that $f(N) = 1$.

$$AIC(\mathcal{B}|D) = LL(\mathcal{B}|D) - |D| \tag{4.11}$$

1.2.4 Normalized Maximum Likelihood score

The *Normalized Maximum Likelihood* (NML) score, recently proposed by Kontkanen et al. [114], is another approach attempting to formalize the penalizing function appearing in the LL equation.

The idea behind NML codes is the same of *universal coding*, recasted in a stochastic context [109]. Given a set of probability distributions $\mathcal{H}$ the encoder relies on the *best-fitting hypothesis* H, i.e. one distribution $H \in \mathcal{H}$ that will assign high likelihood (low code length) to the incoming data D; therefore, we will like to design a code, related to a distribution H', that for all D it will compress D as close as possible to the best-fitting hypothesis H.

In order to compare the *performance of a distribution H with respect to H' of modeling D* we can compute:

$$-\log\big(\Pr(D|H)\big) + \log\big(\Pr(D|H')\big). \tag{4.12}$$

In the NML metric the penalty term is defined as *regret*: it is derived from the equation above in a way that allows a comparison in performance between the best-fitting hypothesis in a *set of probability distributions $\mathcal{H}$* and a distribution $\bar{H}$ not necessarily in $\mathcal{H}$; the *regret of $\bar{H}$ relative to $\mathcal{H}$ for D* is:

$$-\log\big(\Pr(D|\bar{H})\big) - \min_{H\in\mathcal{H}}\Big(-\log\big(\Pr(D|H)\big)\Big). \tag{4.13}$$

We can define $H_{\mathcal{H}}(D)$ as the distribution that minimizes $-\log(\Pr(D|H))$.

The *universal distribution relative to $\mathcal{H}$ for data of size N* is one that minimizes the worst-case regret:

$$H_{\mathcal{H}}(N) = \min_{\bar{H}} \max_{D:|D|=N}\Big(-\log\big(\Pr(D|\bar{H})\big) + \log\big(\Pr(D|H_{\mathcal{H}}(D))\big)\Big) \tag{4.14}$$

The penalizing term in this scoring function is given by the *parametric complexity of $\mathcal{H}$ for data of size N*:

$$C_N(\mathcal{H}) = \log\left(\sum_{D:|D|=N} \Pr(D|H_{\mathcal{H}}(D))\right) \tag{4.15}$$

Thus, for a fixed network structure $\mathcal{G}$, we have:

$$NML(B|D) = LL(B|D) - C_N(\mathcal{B}_{\mathcal{G}}). \tag{4.16}$$

Unfortunately, it is not possible to compute $C_N(\mathcal{B}_{\mathcal{G}})$ efficiently, since it involves an exponential sum over all possible data of size N. Moreover, this score is not decomposable.

1.2.5 Factorized Normalized Maximum Likelihood score

This scoring function is an heuristic enhancement of NML developed by Roos et al. [115] to make NML method usable in practice.

Factorized Normalized Maximum Likelihood (fNML) idea is to approximate $C_N(\mathcal{B}_{\mathcal{G}})$ expression by considering only the contribution to the parametric complexity of the multinomial distributions associated to each variable given a parent configuration.

This makes the score *decomposable* as well.

On the other hand this metric is not *score equivalent.*

1.3 Scoring Function Selection

There is no scoring function that is suitable for any situation: each of them is characterized by advantages and disadvantages. As pointed out by Liu et al. [62], by averaging on the sample size, BDeu is able to converge to the target network but with an appropriate α value, fNML can converge very quickly but sometimes it converges to a different network, whereas MDL presents the best behavior because it often converges to the target network; moreover, with little data, fNML seems to be a good scoring function and performs better than MDL, that in general still converges but does it more slowly.

1.4 Indegree and Data Fragmentation

In all S&S methods, if we have not available an extremely large training set, it is reasonable to put a restriction on the maximum indegree of a variable, i.e. the maximum allowed number of parents.

Berzan explained the rationality of this supposition in [12] by means of the following example:

> "Consider a BN where all variables are binary. If a variable X has no parents, then to compute the Bayesian score we only need two counts from the training set: $D(X = 0)$ and $D(X = 1)$. If X has one parent Y, then we need four counts: $D(X = 0, Y = 0)$, $D(X = 0, Y = 1)$, $D(X = 1, Y = 0)$ and $D(X = 1, Y = 1)$. If X has two parents we need eight counts, and so on. In general, if X has k parents, we need a total of 2^{k+1} counts: as the number of parents Pa_X increases, it becomes less

and less likely to find representative [and reliable] counts for all possible assignments $D(Pa_X = pa_X, X = x)$ in the training set. This phenomenon of *data fragmentation* means that we can only learn networks where each node has a small number of parents. Thus, setting the maximum indegree k to a small value is justified."

CHAPTER 5

HYBRID CONSTRAINT-BASED – SEARCH AND SCORE STRUCTURE LEARNING WITH GENETIC ALGORITHMS

In this chapter a new collection of Hybrid methods for BN structure learning is proposed: each of them consists of a first Constraint-Based part – useful to optimally reduce the search space – and of a second Search and Score part – responsible of finding the best-performing structure – given by a Genetic Algorithm.

1 The Logic Underlying Hybrid Strategies

As explained in Chapter 3 and Chapter 4, we can tackle the BN structure learning problem by progressively reducing the number of possible structures belonging to the search landscape until one only, hypothetically the best one, remains (CB strategy) or by moving across the search space on the basis of some guiding criterion until some suitable stop condition occurs (S&S strategy).

Both above-mentioned strategies are characterized by insidious disadvantages, more and more relevant as the search space increases in size:

- CB methods need a great amount of data to be considered sufficiently trustworthy, and even in such a case, they lose in reliability as soon as they make use of CI tests characterized by too high orders [7] (a common phenomenon with large networks);

- S&S approaches are always limited by bounds such as the maximum number of iterations: even if one generic S&S algorithm is able to efficiently explore the landscape related to one particular network, the more we increase the network size (and thus search space size) the less the search landscape will be exhaustively explored by the algorithm.

What Hybrid methods attempt to do is try to *achieve the best of both CB and S&S worlds*, so that they can better deal with reasonably small datasets (a serious problem for CB-only strategies) as well as reasonably large networks (a major issue to S&S strategies).

A consistent amount of past work focused on tackling the task by means of Hybrid procedures, as we reported in Section 3.

2 An Overview of Proposed Methods

In this thesis work we aim at designing a Hybrid BN structure learning algorithm characterized by the following claims:

1. it should be able in a first moment to reliably reduce the search space, in a way that a larger but safer search landscape is preferred to a more restricted one; in other words the priority in the first phase is to get a super-structure that is as *sound* as possible, i.e. it should include the largest possible portion of the BN to learn;
2. in the second phase it should search as homogeneously as possible across the resulting landscape, and at the same time by optimally steering search efforts towards some knowledge-guided direction;

3. during the second phase it should also be able to further reduce the search landscape on a test-and-set basis without affecting exploration (reliability is still the main priority) so to allow feasible execution time and space requirements for large networks and address data fragmentation issues.

In order to justify the first two claims, we decided to make use of an algorithm similar to Vafaee's HSL-GA [15], enhanced with Vafaee et al.'s ultimate findings on GA methods [52, 53]: indeed we will make use of the most reliable CB method for search space reduction among those taken into account (it experimentally resulted to be the same employed by Vafaee in [22]) and of a GA (able to operate a global search across the landscape) for the second phase, specifically characterized by an evolution process informed by data – based on Vafaee et al.'s contribution in [52] – and able to *keep the balance between exploration and exploitation* – on the basis of Vafaee et al.'s contribution in [53].

GAs are powerful tools, very suitable to globally explore huge search landscapes, and this is why we suppose they are the best choice to operate the search.

Unfortunately, nothing ever comes for free: aside from relevant computational time and space amounts, that heavily depend on network size and that are relatively larger with respect to other S&S techniques, an important limitation in GAs is the strong dependence on parameters values.

In general, evolutionary algorithms are characterized by several parameters, such as population size, number of iterations, genetic operator (mutation and crossover) rates, and often many others: they all require to be carefully adjusted and tailored not only with respect to the general problem in hand, but also to each particular instance of the problem. For example, in this context we can consider BN Structure Learning as the general problem and the ALARM network [116,117] as a particular instance of the problem; furthermore, several related datasets, characterized by different sizes, constitute each a distinct sub-instance of the problem, to be tackled with a different set of parameters.

Consequently, besides above-mentioned claims, these methods are implemented with another clear objective in mind, that is to make them as insensitive as possible to the various adjustable parameters. We will find out in Sections 5.4 and 5.4.2 and experimentally ascertain in Section 4.3 how Vafaee et al.'s findings in [52, 53] will help with this further issue as well.

Finally, we propose another procedure characterized by a new strategy based on parent reduction in order to fulfill our above-mentioned third claim: it attempts to reduce the search space across the evolution by selecting at each iteration a subset of current parents set for each node, so to restrict all possible DAGs contained by the super-structure to a subset of them, that is as smaller as the lower is the maximum size for the parents set.

Since this novel enhancement introduces an additional parameter (i.e. the maximum number of parents threshold for each node), we also propose a further variation of this new method, that attempts to achieve an improved insensitivity with respect to this parameter.

In Figure 2 it is represented the general structure of our hybrid methods.

First, we can observe that the CB part is the same for all of them: the Opt0SS algorithm, on the basis of provided dataset D and set of vertices V, returns a super-structure SS; then, each of our hybrid methods involves a specific genetic algorithm for the S&S phase: each of them – namely Standard GA, SiRGA, DiG-SiRGA, PaRe-DiG-SiRGA and SPaRe-DiG-SiRGA – takes as input the super-structure (denoted by SS) and returns the DAG constituting the structure of the learnt BN.

All mentioned algorithms are described in details in Sections 4 and 5.

In the schematics depicted in Figure 2 we can also observe that presented GAs blocks are portrayed in a superposed arrangement: we will see in Section 5 how each S&S method, starting from Vafaee et al.'s Site-specific Rate GA [52] can be considered as an extension to the previous one, constituting its basis.

The presented methods have been implemented within MATLAB by means of the *Bayesian Network Toolbox – Structure Learning Package* (BNT-SLP) [108], developed on top of the *Bayesian Network Toolbox* (BNT) [16].

3 Graph Representation

The required representation for this Hybrid approach should encompass graphs constituted of directed as well as undirected edges: the standard *adjacency matrix* representation suits our needs.

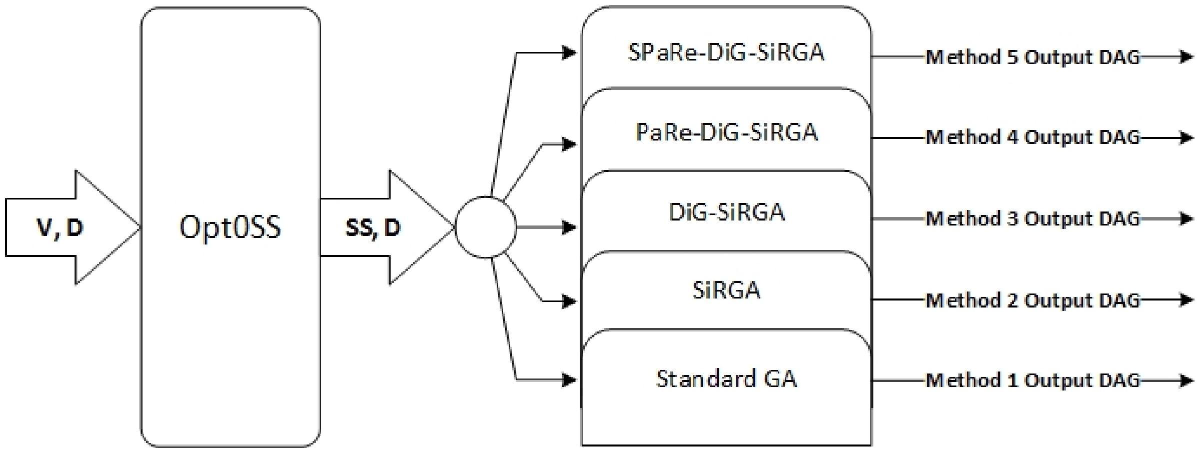


Figure 2: General structure of our presented hybrid methods.

Given $|V|$ variables, a generic graph $\mathcal{G}(V,E)$ can be represented as a $|V|\times|V|$ binary matrix, where a position (x,y) with a 1 denotes the presence of a directed edge $X \rightarrow Y$ and a position (x,y) with a 0 denotes the absence of that directed edge. In other words:

- $(x,y) = (0,0) \qquad \Longleftrightarrow \qquad (X \not\rightarrow Y)$
- $(x,y) = (0,1) \qquad \Longleftrightarrow \qquad (X \rightarrow Y)$
- $(x,y) = (1,0) \qquad \Longleftrightarrow \qquad (X \leftarrow Y)$
- $(x,y) = (1,1) \qquad \Longleftrightarrow \qquad (X - Y)$

In particular, since the target DAG cannot contain cycles by definition, the matrix main diagonal will be kept null during the whole algorithm, in the original super-structure as well as for any individual in the population.

Even if this representation requires a double access to the matrix to know the state of an edge (because we have to check both edge extremes), we chose it to get input compatibility with BNT-SLP toolbox functions without the need of any conversion procedure. Used toolbox routines are the `cond_indep_chisquare` function, needed for CI tests, and the `score_dags` function, i.e. the scoring function.

4 Super-Structure Construction

We tried to use the Opt01SS algorithm, described in Section 1.6, for the CB phase of the algorithm: given a set of variables and related fully observed data, it attempts, by primarily focusing on reliability, to provide a *super-structure* containing the target DAG, on the basis of a test-and-set procedure driven by CI tests of the type that resulted to be the most performing experimentally, as we will report in Section 3; this method is carefully tailored to address issues that heavily occur with data scarcity, i.e. *Approximate Deterministic Relationship* (ADR) and *Inconsistent Conditional Independence and Dependence* (CIDS) statements.

This first stage is required in order to limit the DAGs space that will be explored by the GA, so to focus its search on a reduced number of structures and simplify its job.

As we will see in Section 3, since we want to make the algorithm suitable especially in situations characterized by extreme data scarcity, a simpler, basic and more reliable approach will be preferred: starting from the complete graph, 0^{th}-order CI tests will be applied to all node pairs; whenever two nodes in a pair result to be unconditionally independent then the edge between them is removed. Its reliability resides in the fact that only 0^{th}-order tests are used: indeed the higher the order, the less reliable the CI test [7].

We will name this method *Opt0SS*, as opposed to Opt01SS.

5 DAGs Evolution

Provided with a reliable super-structure containing the final DAG, the GA is at this stage responsible of efficiently exploring the search space driven by a scoring metric so to finally produce an highly-scoring and possibly highly-performing DAG.

Five different implementations of this stage are proposed: a first simple routine shaped as any standard GA, a Site-specific Rate GA based on Vafaee's algorithm [52], its adaptive enhancement reported in [53] defined as DiG-SiRG algorithm, a novel method where parent reduction is performed based on Elite guidance and another new method based on previous one, where the threshold defining maximum number of parents per each node is automatically adjusted on the basis of current Elite population.

First we will present a brief introduction on Evolutionary Computation history, then a review of main Evolutionary Computation constituents is proposed, and finally we will go through our genetic methods.

5.1 Evolutionary Computation in the Literature

In 1950, Turing [118] proposed a "learning machine" which would parallel the principles of evolution. After some years, in 1957 Fraser [119] published a series of papers on simulation of organisms artificial selection. Rechenberg et al. in 1973 [120] presented an efficient optimization method based on *evolution strategies* able to solve complex engineering problems.

Another method was Fogel's evolutionary programming technique, which was proposed for generating artificial intelligence [121].

Genetic algorithms (GAs) were introduced by Holland in 1975 [122].

Research in GAs remained largely theoretical until The First International Conference on Genetic Algorithms in 1985.

5.2 Evolutionary Computation Fundamentals

In this section the main components of evolutionary algorithms are presented in their variations, from the perspective of the employed method, i.e. Genetic Algorithms (GAs).

5.2.1 Individual Representation

Canonical GAs use a binary representation of individuals as fixed-length strings over the alphabet $0, 1$ [122]. In this particular context, each individual is represented by a ternary string, where each ternary digit describes the state of a specific edge that was included in the original super-structure as oriented in one of the two available directions or as absent. Length l of the string is given by the number of undirected edges in the original super-structure.

From an implementative point of view, each individual is actually given by an adjacency matrix: it is manipulated with respect to a list of index pairs that include relevant matrix positions, i.e. edges that were originally present in the input super-structure; all other positions are kept null during the whole evolution process.

5.2.2 Initialization

With the *Initialization* procedure, we can generate a population of N individuals to evolve, possibly provided with some input. In very large search spaces (i.e. with large network sizes) it would be a good idea to spread the population across the search space so to cover it as uniformly as possible: indeed all methods seen in the literature generate the initial population in a random fashion.

A full genetic S&S method would initialize the population without any constraint, i.e. by generating a set of structures over the whole DAGs space defined by the V set. As previously explained, the above-mentioned space becomes very large even with a few nodes only: indeed the number of possible structures grows *super-exponentially* by the number of variables $|V|$, i.e. with a complexity $\mathcal{O}(|V|!2^{\binom{|V|}{2}})$ [14, 15]. This means that, even provided with a powerful GA, in order to keep the same learning performance with respect to an increase in the number of variables, we should raise the value of some parameter (such as population size or number of iterations) with a relatively larger rate: in this way, we will eventually deal with parameters involving very high computational power, which is what makes a basic method inapplicable to real cases with many variables.

Moreover, network edges density may be an issue, because this (commonly unknown) characteristic of the network may increase the sensitivity of the method with respect to the individual density in the initial population.

The first presented problem can be addressed by reducing the search space before the evolution: the CB stage of the Hybrid approach allows us to ascertain the absence of a set of edges in the complete graph, so to make the GA deal with a super-structure bounded to a greatly reduced DAGs space (quantitative experimental results about this reduction are provided in Section 3). Indeed, all individuals in the population will be particular instances of the original super-graph, and not of the complete undirected graph.

In 5.4 we will see how to overcome the second problem: Vafaee et al. SiRG method [52] is capable of making GA performance independent of initial individual's density.

5.2.3 Mutation

Mutation in canonical GAs works through the *bit-flip* technique: each bit is flipped with probability μ where $\mu \in (0,1)$ is a control parameter referred to as *mutation rate*.

Accordingly, the mutation probability for a single individual is $\Pr(\text{mutation}) = 1 - (1-\mu)^l$.

By following a naïve approach, mutation operator in GAs has been set in many works in the literature as equal to $1/L$, where L is the individual's length.

Mutation allows an individual to perform with some chance a little step in the search space, along any of its dimensions: it is indeed useful to *explore* the space. In the standard approach, this step is taken in a random way, with the possibility of undertaking disadvantageous choices; Vafaee et al. [52] proposed a smart way to focus mutation on more convenient directions, consisting in their *Site-specific Rate* mutation scheme. It will be analyzed in-depth in Section 5.4.

5.2.4 Crossover

Crossover or *recombination* is a binary genetic operator responsible of generating some offspring from two parents. Once that two individuals are selected for reproduction, it actually occurs if an independent random experiment yields a positive outcome: the *crossover rate* χ indicates the probability that the above-mentioned experiment succeeds.

In the literature there are two forms of crossover operators:

n-point crossover : the two parent strings are cut into $n + 1$ segments along the randomly chosen crossover points. The first offspring is obtained by concatenating odd segments from the first parent and even segments from the second one, whereas the second child is given by combining odd segments from the second parent and even segments from the first one.

n-point crossover for $n > 2$ is rarely used in GA applications [22].

An example of this crossover method is given by [123].

uniform crossover : each bit of the first child is randomly picked from one of the two parents, whereas the second offspring will be constituted by bits taken by means of opposite decisions.

In [124] is reported an example of uniform crossover.

We employed uniform crossover strategy in this thesis work, in a slightly modified approach: each individual of the population constitutes the first parent; the second parent is randomly chosen in the same population after a series of random experiments, in which an individual $\mathbf{x_i}$ is eligible for mating if $f(\mathbf{x_i}) > \mathcal{U}(0,1)$, where $\mathcal{U}(0,1)$ represents a real number randomly picked in the interval $(0,1)$; a single offspring is finally *added* (without replacing any existing individual) to the population, by following the standard uniform crossover operation.

5.2.5 Selection

Any selection operator operates by considering fitness values of individuals so to let best-performing ones keep evolving and to remove bad individuals from population.

Various selection schemes have been applied to GAs in the past:

Proportional selection : this scheme uses the *relative fitness*

$$\Pr(\mathbf{x_i}) = \frac{f(\mathbf{x_i})}{\sum_{j=1}^{n} f(\mathbf{x_j})} \tag{5.1}$$

to determine the selection probability of an individual $\mathbf{x_i}$.

An example of this method is given by [125].

This is the selection strategy we employed in this work.

Rank-based selection : individuals are ordered with respect to their fitness, then their *rank* (order position) is used as selection probability after being normalized with respect to the sum of the ranks.

[126] constitutes an example of a linear mapping technique following this idea.

Tournament selection : as explained by Vafaee in [22], "this method works by first taking a random uniform sample of a certain size (called *tournament size*) from the population. It then selects the best of these individuals to survive for the next generation, and repeats the process until the new population is filled."

[127] is an example of this selection method.

5.3 A Simple Genetic Algorithm

In this section the base version of our method, directly inspired by Vafaee's method in [22], is described: it will be explained in detail line-by-line by referring to Algorithm 1.

Algorithm 1: Standard genetic algorithm.

```
Input:  $SS$ – complete super-structure
        $data$ – dataset
        $N$ – population size
        $M$ – maximum number of generations
        $MP$ – maximum fan-in
        $scoring\_fn$ – scoring function
Output: $\mathbf{x}^*$ – elite individual
1  $P \longleftarrow$ INIT-POPULATION($SS$, $2N$)
2  $P_{dag} \longleftarrow$ MAKE-DAG($P$)
3  $P_{dag} \longleftarrow$ LIMIT-PARENTS($P_{dag}$, $MP$)
4  $score \longleftarrow$ SCORE-DAG($data$, $P_{dag}$, $scoring\_fn$)
5  for $i = 1 : M$ do
6      $\mathbf{x}^* \longleftarrow$ GET-ELITE($score$, $P_{dag}$)
7      [$P_{dag,1}$, $score_1$] $\longleftarrow$ SELECTION($P_{dag}$, $score$)
8      $P_2 \longleftarrow$ CROSSOVER($P_{dag,1}$)
9      $P_{dag,2} \longleftarrow$ MAKE-DAG($P_2$)
10     $P_{dag,2} \longleftarrow$ LIMIT-PARENTS($P_{dag,2}$, $MP$)
11     $P'_{dag} \longleftarrow$ [$P_{dag,1}$ $P_{dag,2}$]
12     $P'' \longleftarrow$ MUTATION($P'_{dag}$)
13     $P''_{dag} \longleftarrow$ MAKE-DAG($P''$)
14     $P''_{dag} \longleftarrow$ LIMIT-PARENTS($P''_{dag}$, $MP$)
15     $score'' \longleftarrow$ SCORE-DAG($data$, $P''_{dag}$, $scoring\_fn$)
16     [$P_{dag}$, $\mathbf{x}^*$] $\longleftarrow$ PLACE-ELITE($score''$, $P''_{dag}$, $\mathbf{x}^*$)
17 end
```

5.3.1 Line 1: Population Initialization

The INIT-POPULATION routine is responsible of initializing a population of individuals starting from the super-structure, by producing $2N$ directed instances of it, where N is an input parameter denoting the size of selected population at each generation.

Each individual is built as in the following: only edges existing in the original super-structure are taken into account, whereas edges that were absent in the super-structure won't be considered and will be absent in every individual, till the end of the evolution process; for any individual, given each super-structure edge, a random choice decides whether setting it to one of the two orientations or whether removing it.

This choice can be calibrated so to produce a population of denser or sparser individuals, i.e. structures characterized respectively by a high or low number of edges with respect to the total number of undirected edges in the super-structure: for instance, by setting the *probability of orientation at initialization*, denoted as POI, to 0.2, the initial population will be constituted of individuals having in the average a number of directed edges approximately equal to 1/5 of the number of edges in the original super-structure.

This possibility to calibrate the starting number of individuals' edges will allow us to estimate the improved insensitivity of our Site-specific Rate method's performance with respect to variations in individuals' initial density.

5.3.2 Lines 2, 9, 13: Directed Structure to DAG conversion

Since any individual – being an adjacency matrix – can represent any possible directed or undirected structure, it may be provided with cycles during the crossover or mutation processes. The Make-DAG routine is indeed needed to convert a generic directed graph to its as similar as possible DAG, i.e. with the minimum number of changes.

Its core consists in a slightly modified version of a heuristic method developed by Eades et al. [128] called *GR* algorithm, capable of solving the *minimum Feedback Arc Set* problem with a fast heuristic approach and with close-to-optimal results.

The slightly modified, improved version is taken directly from Vafaee's contribution on the matter [15], where in particular the *vertex degree* is defined so to give precedence to nodes with more children and less parents, in a way that further minimizes the Feedback Arc Set; finally at the end of this procedure the Feedback Arc Set is deleted, so to yield a DAG as output.

5.3.3 Lines 3, 10, 14: Parents Limitation

The Limit-Parents function constrains the maximum fan-in for each node in the input DAG(s) to the value MP (Maximum number of Parents): in this basic GA its usefulness is to only solve a logistic problem, indeed it is used to avoid high computational requirements, caused by the large size of Conditional Probability Tables related to nodes with too many parents.

In this basic implementation, as well as in its variations inspired on Vafaee et al.'s work in [52, 53], we assume that structure evolution is practically unconstrained with respect to the maximum number of parents a node can have at each generation. Indeed, given that in the experiments we will consider a sufficiently high value for the maximum number of parents so that:

- computational requirements are bounded in a way that allows the used machine to execute within a reasonable time frame and without running out of memory,
- parents reduction action influences the evolution process in a negligible way,

we decided to adopt a very simple strategy to perform parent reduction.

Given the parents set of a vertex of exceeding size with respect to the maximum fan-in threshold value MP, a parent per time is randomly picked from the set and detached from the node, until the parents set size is reduced to MP. We will see how we can actually exploit parent reduction to further reduce the search landscape and better steer the evolution process to convergence in sections 5.5 and 5.6; in this context we suppose to deal with a sufficiently high constant value for MP just to solve our computational problems.

5.3.4 Lines 5, 15: Fitness Computation

A GA should be driven by some metric, a way to score each individual with some fitness value so to consequently be able to perform the selection procedure: the Score-DAG routine evaluates such score when needed.

As reported in Section 1 of this thesis work, a plethora of methods for scoring BNs with respect to their fitness to some related data have been introduced in the past; however, given that in real situations we are not provided with the necessary elements of information (such as network density or prior distribution) to fully rely on any of the available scoring functions, we should always consider them as part of an informative but also deceptive environment. In other words, scoring functions can help but don't constitute the perfect source of knowledge, especially in data scarcity conditions.

In this work we chose to use the BDeu score with equivalent sample size $\alpha = 1$ as fitness function; its cached implementation is already provided in the BNT-SLP toolbox.

5.3.5 Lines 6, 16: Elite Propagation

This method is *elitist*: it means that the highest-fitting member is replicated from the population at the current generation and reinserted in the next generation population, if it still results to be the best individual.

This operation is executed by Get-Elite and Place-Elite routines: at the beginning of each generation the highest-scoring DAG is stored in $\mathbf{x}^*$; at the end of the current generation the $\mathbf{x}^*$ score is compared with the fitness of the best individual of the current population after crossover and mutation: if the original $\mathbf{x}^*$ is still the highest-scoring individual then it replaces the worst-fitting individual in the population, otherwise $\mathbf{x}^*$ is updated with the new, highest-scoring individual.

5.3.6 Line 7: Selection

At the beginning of each generation the population of $2N$ individuals goes through the Selection procedure, along with their scores.

For a reason of computational convenience useful in the crossover phase, fitness values are normalized with respect to best and worst scores in the current population, then N individuals are selected on the basis of the *proportional selection* scheme: a loop cycles on all $2N$ individuals so to compare each individual's fitness to a random value extracted from the uniform distribution and to possibly admit it in the selected population, until the latter is filled with N individuals.

5.3.7 Line 8: Crossover

A modified uniform crossover strategy is applied in this method with the CROSSOVER procedure so to produce a population of $2N$ individuals starting from N individuals, as described in Section 5.2.

5.3.8 Line 12: Mutation

After recombination, each individual undergoes the MUTATION procedure: one edge per time, among those ones allowed to mutate (i.e. edges that existed in the original superstructure), mutates with probability $1/L$, where L is the individual length, or rather the number of its mutable edges; if the random experiment results in a mutation for the edge, then its state is randomly changed to one between the other two available (e.g., if edge before mutation is $(0,1)$ / $\rightarrow$, then after mutation it becomes $(1,0)$ / $\leftarrow$ or $(0,0)$ / $\not\rightarrow$ with equal likelihood).

5.4 A Site-specific Rate Genetic Algorithm

The second method we chose to use is directly derived from one of Vafaee et al.'s contributions [52] and adapted for the BN structure learning task. It consists in a *Site-specific Rate Genetic* (SiRG) algorithm, i.e. a GA that embeds an adaptive mutation scheme able to make *exploration* and *exploitation* search mechanisms work together, whereas in standard GAs these two dynamics behave as opposite forces, as it is commonly known [129].

As stated by Vafaee et al. in [52], "*Exploration* is the ability of a search algorithm to discover unseen regions of the search space, and to avoid convergence to local optima. On the other hand, *exploitation* is defined as 'the ability of an algorithm to step into the direction of the desired improvement' [130], or 'the capability of the good use of good information' [131]."

The proposed method aims at retaining the balance of exploration and exploitation by developing them where it is needed: the mutation operator is dynamically adapted to each individual with respect to its fitness and to each edge with respect to its profusion among the best individuals. Mutation rates are derived in order to preserve good individuals and to freely disrupt bad individuals, so to make them further explore the search space; furthermore, these mutation rates are defined to be dynamic in the sense that they are *adaptively* controlled throughout the evolution.

This SiRG approach was originally derived from the biological theory of *motif representation.* As Vafaee et al. pointed out in [52], "in genetics, a *DNA motif* is defined as a nucleotide acid sequence pattern that is widespread and has some biological significance [132]. Given a set of m DNA sequences, the problem of *motif discovery* can be roughly defined as finding a set of m

subsequences $X = \{x_1.x_2, \ldots, x_m\}$, one from each input sequence, that maximizes a predefined scoring criterion."

The set of discovered patterns can be represented in a complete *matrix form.* Given that a subsequence is supposed to lie in one among l possible specific sites in a generic sequence, the above-mentioned matrix, denoted as *position frequency matrix* (PFM), stores the frequency of each possible subsequence residing in a specific site over all sequences, for all sites.

The method is provided in Algorithm 2: structurally it is the same as the first one, excepting for the Form-Elite-Group, Construct-PWM and SiRG-Mutation routines.

As previously done with the standard GA, new functions will be explained in detail below by referring to their line numbers in Algorithm 2. For the reader's convenience, newly introduced procedures, with respect to the standard GA, are indicated in bold in the pseudocode.

Algorithm 2: Site-specific rate genetic algorithm.

```
Input:  $SS$ – complete super-structure
        $data$ – dataset
        $N$ – population size
        $M$ – maximum number of generations
        $MP$ – maximum fan-in
        $scoring\_fn$ – scoring function
        $\alpha$ – elite eligibility threshold
        $\epsilon$ – small positive number
Output: $\mathbf{x}^*$ – elite individual
1  $P \longleftarrow$ Init-Population($SS$, $2N$)
2  $P_{dag} \longleftarrow$ Make-DAG($P$)
3  $P_{dag} \longleftarrow$ Limit-Parents($P_{dag}$, $MP$)
4  $score \longleftarrow$ Score-DAG($data$, $P_{dag}$, $scoring\_fn$)
5  for $i = 1 : M$ do
6      $\mathbf{x}^* \longleftarrow$ Get-Elite($score$, $P_{dag}$)
7      $[P_{dag,1}, score_1] \longleftarrow$ Selection($P_{dag}$, $score$)
8      $P_2 \longleftarrow$ Crossover($P_{dag,1}$)
9      $P_{dag,2} \longleftarrow$ Make-DAG($P_2$)
10     $P_{dag,2} \longleftarrow$ Limit-Parents($P_{dag,2}$, $MP$)
11     $P'_{dag} \longleftarrow [P_{dag,1}\ P_{dag,2}]$
12     $\mathcal{E} \longleftarrow$ **Form-Elite-Group**($P'_{dag}$, $\alpha$)
13     $PWM \longleftarrow$ **Construct-PWM**($\mathcal{E}$)
14     $P'' \longleftarrow$ **SiRG-Mutation**($P'_{dag}$, $PWM$, $\epsilon$)
15     $P''_{dag} \longleftarrow$ Make-DAG($P''$)
16     $P''_{dag} \longleftarrow$ Limit-Parents($P''_{dag}$, $MP$)
17     $score'' \longleftarrow$ Score-DAG($data$, $P''_{dag}$, $scoring\_fn$)
18     $[P_{dag}, \mathbf{x}^*] \longleftarrow$ Place-Elite($score''$, $P''_{dag}$, $\mathbf{x}^*$)
19 end
```

5.4.1 Lines 12, 13, 14: Site-specific Rate Mutation Scheme

The *Site-specific Rate Mutation* procedure starts by recruiting best individuals in the population by means of the Form-Elite-Group routine (line 12), in order to constitute the *Elite set*: each individual's fitness is compared with respect to the *Elite Eligibility Threshold*, denoted by α (with $\alpha \in [0.5, 1]$) and, if the score is above α, the individual is admitted in the Elite set; the comparison consists in the following inequality:

$$f(x_k) \geq \alpha f_{max}. \tag{5.2}$$

In this context we define a mutable sequence of the individual, i.e. an edge with a mutable state, as a *locus*; moreover each component in the set of values that a locus can assume is defined to be an *allele*. Individuals in the Elite set can be thought as leaders guiding the people towards higher peaks in the search landscape, at least for what concerns the *loci states*. Indeed their overall structural information is employed in order to consequently yield a high mutation chance for those loci whose state is not spread among the elitist population, and conversely, to reduce the mutation probability for "good" loci.

The presented mechanism is implemented through the *Position Weight Matrix*, similarly to the motif discovery context described in Section 5.4; by means of the Construct-PWM function (line 13) we can indeed construct a $PWM = (w_{i,j})$, in which each element $w_{i,j}$ corresponds to the *probability* of occurrence of allele $i \in \{\not\rightarrow, \leftarrow, \rightarrow\}$ (encoded as $1, 2, 3$) at locus $j \in [1, L]$ (with L equal to the individual length), weighted by individual's fitness. More formally, let $\chi = \{x_1, x_2, \ldots, x_m\}$ be the set of selected individuals and let $w_{i,j}$, derived by Equation 5.3, be the probability of occurrence of allele i at position j within the elite set; in Equation 5.3 $f(x_k)$

is the fitness of individual x_k, while the delta function $\delta(x_{k,j} = i)$ goes to 1 when the allele of the individual x_k at locus j is equal to i.

$$w_{i,j} = \frac{\sum_{k=1}^{m} f(x_k) \cdot \delta(x_{k,j} = i)}{\sum_{k=1}^{m} f(x_k)} \tag{5.3}$$

Provided with a PWM embodying the underlying pattern of the promising solutions discovered so far, the SiRG-Mutation procedure (line 14) computes a distinct mutation rate $\mu_{k,j}$ for each site $j = 0, \ldots, l-1$ of every individual $x_k \in P'_{dag}$ and possibly performs mutation on that specific site; if mutation, driven by a random experiment, succeeds, then the state of current individual's related site is randomly changed to one between the remaining two possible alleles.

$\mu_{k,j}$ is computed according to Equation 5.4, where $w_{i,j}$ is the $(i,j)^{\text{th}}$ element of PWM, $f(x_k)$ is the current individual's fitness, f_{max} is the fittest-individual's fitness and ϵ is a small positive number to avoid zero probabilities:

$$\mu_{k,j} = \left[\epsilon + \left(1 - \epsilon\right)\left(1 - \frac{f(x_k)}{f_{max}}\right)\right] * \left[\epsilon + \left(1 - \epsilon\right)\left(1 - w_{i,j}\right)\right]. \tag{5.4}$$

Equation 5.4 meaning is explained here by quoting Vafaee et al. words in [52]:

"the mutation rate at each site is gauged based on the following two criteria:

1. The probability of having the corresponding allele at that site based on the PWM constructed out of high-quality individuals. Since the pattern of good

solutions is somehow represented by a PWM, being in conformity with this pattern should give rise to lower rates of mutation and vice versa.

2. The fitness of the individual which is about to get mutated; in general better individuals are more probable to carry more valuable information; accordingly, they need to be modified more carefully. In other words, assigning them high mutation rates is rather risky and may cause destruction of useful information."

5.4.2 Diversity Guide Enhancement

The SiRG scheme allows the GA to retain a good ratio between exploration and exploitation, but there is still the possibility that this adaptive mechanism, led by the Elite set, eventually goes out of control.

The proposed method can be augmented with a diversity control process so to more directly monitor the balance between exploration and exploitation and to ensure its retention across the search procedure.

Diversity refers to differences among individuals: "loss of diversity corresponds to the lack of exploration, resulting in premature convergence and trapping into local optima. Excessive diversity, on the other hand, is counterproductive when high exploitation is required" [53].

In a large, complex and strongly multimodal landscape such as any DAGs space, a diversity monitoring feature in the Elite set can be a good enhancement, especially if we want to make the algorithm suitable to networks characterized by different sizes. E.g., when we deal with a small-sized network, the search space can be explored efficiently even with an Elite set that is little with respect to the remainder of the population, because such a landscape would be generally characterized by a small number of peaks (where the few elitist people can still distribute): in this context the non-elitist individuals can focus on the few directions given by the little Elite set, thus contributing more to exploitation; on the other hand, if we employ the same little Elite set on a larger landscape (associated to a larger network), then it is likely that the few elitist individuals cannot distribute uniformly across a reasonable portion of peaks in the search space: the population will probably evolve in the direction of small, sub-optimal peaks; in this second case it is clearly needed a relatively larger Elite set, more prone to explore the landscape instead of directly converging to nearest peaks.

The entire mechanism described above depends on individuals' diversity, that can be more concretely defined as the individuals' strings reciprocal *Hamming distances* and, consequently, on the distribution of alleles probabilities on each row of the PWM matrix. As explained by Vafaee et al. in [53], "if $\mathcal{E}$ happens to contain genotypically identical individuals – i.e., no diversity – all the allele probabilities ($w_{i,j}$'s) of the corresponding PWM matrix drop off to 0 or raise to 1. Such PWM will possibly lead the evolution towards trapping into local optima and premature convergence resulted from the loss of enough exploration. Conversely, by an excessively diverse $\mathcal{E}$, the PWM loses the information content required for an effective exploitative search."

Hence, the key to solve our issue is to find a diversity-driven way to adjust α, i.e. the Elite eligibility threshold: when we need to expand the Elite set so to promote diversity and exploration then it is sufficient to reduce this parameter, on the other hand when more exploitative power is required we can restrict the access to the Elite set by increasing the threshold.

In this work we make use of an *entropy-based* diversity measure in order to monitor the exploration/exploitation equilibrium, accordingly to the same method described by Vafaee et al. in [53]. The amount of *elitist population disorder* can indeed be used as a measure of its diversity.

In information theory, entropy is a measure of uncertainty or disorder in a signal or a random event. Shannon [111] defines the *entropy of a random event X*, with possible states $\{X_1, \ldots, X_n\}$ as:

$$H(X) = -\sum_{i}^{n} p_i \log_2(p_i), \tag{5.5}$$

where p_i is the probability of observing the i^{th} outcome, X_i.

By quoting Vafaee et al.'s words in [53], "when more states are available to a system, the systems' unpredictability and disorder/diversity increase, and thus, entropy rises. When only one outcome is observed, there is no uncertainty, and therefore, the entropy of the system is zero. Conversely, entropy becomes maximal if all the outcomes are equally likely – i.e., if the system has n states which are equiprobable ($p_i = 1/n$), the entropy is maximum":

$$H_{max} = -\sum_{i}^{n} \frac{1}{n} \log_2 \left(\frac{1}{n}\right) = \log_2(n) \tag{5.6}$$

In this context, we define the entropy for a set of selected population $\mathcal{E}$ of size E by first placing each individual x_i into a genotype class C_i. $C_0, C_1, \ldots, C_l$ are possible genotype classes where l is the individual length and C_i comprises individuals whose *hamming distance*[1] with the best individual in the current population, $x^*(P_{dag})$ is i.

p_i in Equation 5.6 constitutes the proportion of the population related to partition C_i. More formally, p_i is derived by Equation 5.7 where $h^*(x_k)$ is the hamming distance between the best individual $x^*(P)$ and the current individual x_k, E is the Elite set size and $\delta(c)$ returns 1 if its boolean argument (i.e. $h^*(x_k) = i$) holds and 0 otherwise.

$$p_i = \frac{1}{E} \sum_{k=1}^{m} \delta\Big(h^*(x_k) = i\Big) \tag{5.7}$$

The procedure is provided in Algorithm 3: structurally it is the same as the second one (i.e. the SiRG algorithm), excepting for the addition of the Update-Alpha procedure.

As previously done with the SiRG algorithm, the new function will be explained in detail below by referring to its line number in Algorithm 2. For the reader's convenience, the newly

[1]Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different.

introduced procedure with respect to its basic method (i.e. the SiRG algorithm) is indicated in bold in the pseudocode.

Algorithm 3: Diversity guided site-specific rate genetic algorithm.

```
Input:  $SS$ – complete super-structure
        $data$ – dataset
        $N$ – population size
        $M$ – maximum number of generations
        $MP$ – maximum fan-in
        $scoring\_fn$ – scoring function
        $\alpha$ – elite eligibility threshold
        $\epsilon$ – small positive number
        $\mathbf{d_1}$ – healthy diversity interval at generation 1
        $\mathbf{d_M}$ – healthy diversity interval at generation $M$
Output: $\mathbf{x}^*$ – elite individual
1  $P \longleftarrow$ Init-Population($SS$, $2N$)
2  $P_{dag} \longleftarrow$ Make-DAG($P$)
3  $P_{dag} \longleftarrow$ Limit-Parents($P_{dag}$, $MP$)
4  $score \longleftarrow$ Score-DAG($data$, $P_{dag}$, $scoring\_fn$)
5  for $i = 1 : M$ do
6      $\mathbf{x}^* \longleftarrow$ Get-Elite($score$, $P_{dag}$)
7      $[P_{dag,1}, score_1] \longleftarrow$ Selection($P_{dag}$, $score$)
8      $P_2 \longleftarrow$ Crossover($P_{dag,1}$)
9      $P_{dag,2} \longleftarrow$ Make-DAG($P_2$)
10     $P_{dag,2} \longleftarrow$ Limit-Parents($P_{dag,2}$, $MP$)
11     $P'_{dag} \longleftarrow [P_{dag,1}\ P_{dag,2}]$
12     $\mathcal{E} \longleftarrow$ Form-Elite-Group($P'_{dag}$, $\alpha$)
13     $\alpha \longleftarrow$ **Update-Alpha**($\mathcal{E}$, $\alpha$, $\mathbf{d_1}$, $\mathbf{d_M}$)
14     $PWM \longleftarrow$ Construct-PWM($\mathcal{E}$)
15     $P'' \longleftarrow$ SiRG-Mutation($P'_{dag}$, $PWM$, $\epsilon$)
16     $P''_{dag} \longleftarrow$ Make-DAG($P''$)
17     $P''_{dag} \longleftarrow$ Limit-Parents($P''_{dag}$, $MP$)
18     $score'' \longleftarrow$ Score-DAG($data$, $P''_{dag}$, $scoring\_fn$)
19     $[P_{dag}, \mathbf{x}^*] \longleftarrow$ Place-Elite($score''$, $P''_{dag}$, $\mathbf{x}^*$)
20 end
```

The UPDATE-ALPHA procedure (line 13) is used to update α value so to guarantee a sufficient and constant amount of diversity in the Elite set across the evolution process in its entirety: at each step α can be incremented, decremented or kept as it is, depending on current elitist diversity measure.

Let $H(\mathcal{E})$ be the *entropy diversity measure of Elite set* $\mathcal{E}$, as defined by Equation 5.6 and Equation 5.7. By directly quoting Vafaee et al. words in [53]: "we can define $[d_{min}, d_{max}] \subset [0, H_{max}]$ to be a *healthy diversity* interval such that if $H(\mathcal{E})$ falls within this interval, then exploration and exploitation are assumed to be in good ratio, and thus the process is proceeded without concern; otherwise, $H(\mathcal{E}) < d_{min}$ implies that evolution is starved of exploration, while $H(\mathcal{E}) > d_{max}$ means that the process falls into excessive exploration."

α update procedure is enclosed in Equation 5.8.

$$\alpha_t = \begin{cases} \alpha_{t-1} - 0.01, & \text{if } H(\mathcal{E}) < d_{min} \\ \alpha_{t-1} + 0.01, & \text{if } H(\mathcal{E}) > d_{max} \\ \alpha_{t-1}, & \text{otherwise} \end{cases} \tag{5.8}$$

Two possible methods are proposed by Vafaee et al. for what concerns the healthy diversity interval (denoted as $\mathbf{d} = [d_{min}, d_{max}]$) definition: we can assign constant values $a \cdot H_{max}$ and $b \cdot H_{max}$ respectively to d_{min} and d_{max}, with $a \in (0, 1)$, $b \in (0, 1)$ and $a < b$; another preferable practice, as stated by Vafaee et al. in [53], is "to make the healthy diversity interval time-sensitive based on the intuition that at the earlier stages of evolution more exploration is required while towards the end of the process exploitation is mode demanding": this results in a time-sensitive interval $\mathbf{d_t}$ based on the definition of two interval pairs, i.e. one identifying the interval values at the beginning of the evolutionary process (denoted as $\mathbf{d_1}$) and one referring to interval values applied at the last generation of the process (denoted as $\mathbf{d_M}$).

5.5 Towards Automatic Parents Reduction: an Elite-Guided Parents Limitation Approach

All of the methods taken into account up to now, as well as related experiments in Chapter 6, assume a maximum fan-in MP, i.e. a *Maximum number of Parents*, per node: even if we keep this maximum bound MP to a sufficiently large value so to make its influence negligible to GAs behavior, we still need it to avoid the need of huge computational power as soon as any individual randomly and eventually gets too many parents for one node, a recurring issue peculiarly with a large number of vertices.

Another important problem we can address with parents reduction, especially in the perspective of using a GA to learn large BNs, is *data fragmentation*, as pointed out in Section 1.4.

Furthermore, it is worth to say that parent reduction can also be an advantage: if we know a sufficiently reliable value for the maximum fan-in per each node in a network to learn, then we can bound any evolving DAG to keep the number of parents per node below that threshold: as a result, the population will be eligible to explore only a portion of the overall search landscape related to the complete input super-structure, and this means that the efforts of each individual will be focused on a more restricted and direct search space.

5.5.1 Parents Limitation in the Literature

Parents Reduction is a practice typical of S&S approaches: indeed in CB methods we only deal with CI tests, where a node indegree does not contribute to algorithm complexity. On the other hand S&S algorithms always use a scoring metric: its computational requirements depend on the conditional probability table size per each node, and they are likely to not be applicable to real-life contexts when a node has too many parents.

The *K2* algorithm [63] assumes an upper bound on the number of parents per node: even if this upper bound is there only to prevent high computational requirements, the inner heuristic method on which parent reduction is based is oriented to prefer the minimum number of parents possible, per each node. Indeed the K2 heuristic greedily adds one parent per time to each node until the score of the substructure does not increase: if a substructure with more additional parents than one only yields a better score, then the K2 algorithm would never know.

The *CB* [99] and *Ordering Search* [83] algorithms, as well as the *K2* [93] and *Chain-Model* [95] genetic algorithms makes use of parent reduction through the K2 method, embedded in them.

The *Chow-Liu MWST* algorithm [56] is limited by the need to choose a root node and the greedy approach: at each step the structure adds the maximum scoring edge among those that make the structure keep being a tree. This does not bound a number of maximum parents to any node, but establishes an ordering among nodes for what concerns parenting: in other words only nodes that are already part of the tree have the possibility to be parents of the nodes that have still to be included.

As an extension to the MWST approach, the TAN algorithm [81] is affected by the same limitations of its core component.

The *HC algorithm* [82] is not characterized by any limitation on the number of parents, but its computational needs are very likely to become inapplicable to real-life situations with large networks throughout the execution, because the greedy process of structure construction is not controlled, in the sense that it allows any substructure of the complete graph; as we already said, with large networks this is likely to result in huge computational needs and a long time to complete the execution.

The *Sparse-Candidate* algorithm [66] focuses its method exactly on the choice of a good set of parents – here named *candidates* – for each node: the main claim of this strategy is that it can optimally reduce the search space so to better steer the exploration process operated by HC algorithm. The candidate set construction is based on the mutual information existing between the node taken into account and its potential parents. Limitations to this method are the need to choose a candidate selection measure and a size for the candidate set, together with the relevant sensitivity of method's performance to these parameters.

In particular, proposed candidate selection measures are the *discrepancy*, *shielding* and *score based* metrics. The *discrepancy* measure is analogous to an independence test with the *Kullback-Leibner divergence*, described in Section 1; on the other hand, the *shielding* measure estimates how strongly a conditional independence statement is violated with respect to the parents set of the node; finally the *score based* metric is based on the score improvement we get by adding the potential candidate to the parents set of the node. The *candidate set size* is the maximum number of parents allowed per each node.

The basic version of *Optimal Reinsertion* algorithm [68] does not involve a limit for the in-degree of each node, and so its computational needs could not be feasible especially with large networks. Anyway the authors developed an OR version enhanced with the Sparse-Candidate strategy, characterized by same properties and conclusions related to it.

The *Greedy Equivalent Search* algorithm [9] contains in its procedure a routine that converts a DAG to its essential graph; since this procedure needs to be set with a maximum number of parents per node, we can say that this can be a limitation if we don't know exactly the maximum fan-in, even more than other methods if we consider that in an essential graph an undirected edge between two nodes denotes one parent more in the parents set of both nodes, and not only one of them.

Carrillo's Simulated Annealing [85] algorithm adopts a parent limitation strategy with fixed threshold based on mutual information between current node and its parent.

In *Larrañaga*'s genetic algorithm [92], for each individual, edges within cycles and directed to nodes having more than k parents are randomly removed until DAG property and maximum fan-in equal to k property are satisfied.

Carvalho's *Cooperative Coevolutionary* genetic algorithm [96] does not restrict the number of parents of each node, but still needs a huge computational power when too many parents are added to a node during the evolution process.

In the μGP genetic algorithm [97] a maximum number of parents per node has to be given as input: when a node reaches this parents limit, no more arcs can be oriented towards it until at least one among other edges directed to it is removed.

Wong's *Cooperative Coevolution* genetic algorithm [94] sets a maximum parents threshold to each node and attempts to tackle the learning problem by decomposing it in n subproblems (with n denoting the number of vertices), where each of them consists in finding, per each node, its actual parents set, on the basis of mutual information metrics between a node and its potential parents.

In particular, the parents set of each node undergoes mutation across the evolution by means of the *move* operator: it modifies the parents set of a node by replacing one of the parents with a nonparent.

The *Hybrid Structure Learner Genetic Algorithm* [15] does not set any limit to the number of parents per node: indeed with large networks computational time speeds up with exponential rate.

5.5.2 Parents Limitation in the Proposed Method

Every time that an individual is evaluated, the scoring function extracts a set of statistics from the whole dataset in conjunction with the individual structure: this evaluation becomes more and more expensive as the number of data instances or structure complexity grows.

By *structure complexity* we refer in particular to each node's parents set size: in the case of a tabular conditional probability distribution, time and space complexity for score evaluation of each node X_i grows with respect to each possible configuration of $X_i \cup Pa_g(X_i)$, and is indeed $\mathcal{O}\Big(\prod_{i=1}^{k} r_i\Big)$, where the considered set of nodes sized k is $X_i \cup Pa_g(X_i)$ and r_i is the number of values that the i^{th} node in the set can assume.

Primary target to achieve with parent reduction is to allow the learning on large networks in a reasonable time frame and with feasible space requirements for Conditional Probability Tables allocation: indeed by setting a maximum fan-in, the time and space complexity for the evaluation of a single node score does not depend on network size anymore: we can thus limit the parents set in any case, regardless of how many potential parents a node can have. As a consequence, the entire scoring procedure on a single DAG with any S&S method driven by standard scoring metrics grows at a bounded-exponential rate with the number of variables, and not exponentially as by leaving unconstrained the maximum indegree.

Furthermore, as stated by Berzan in [12] and explained in Section 1.4, if we try to evaluate the score of a node having too many parents, any scoring function behavior may be compromised by the phenomenon of data fragmentation, especially in situations of data scarcity: by putting a limit on the maximum fan-in, our method is able to address this issue.

Our task is given by the following: given a node with a number of parents NP higher than MP, we should delete the edges between the node and the set of its $NP - MP$ *worst parents*, i.e. the nodes that are less likely to be the actual parents of the node in the target DAG.

In all our previous methods, we dealt with the parent limitation problem by employing the naïve approach to the issue: given a node and its parents set of exceeding size with respect to

MP, one parent per time is randomly picked from the set and consequently detached from the node, until we bring the fan-in for that node below the MP threshold.

This approach can be advantageous sometimes because it is very simple and fast with respect to any other strategy, and moreover, it is unbiased. On the other hand, it is not *informed* from data, in the sense that it does not employ any information extractable from available data (the unique source of knowledge we can make use of) in order to choose the parents to detach.

What we propose in this section is a parent limitation strategy that possibly gets the information it needs directly from available data, the unique source of knowledge we have available: our claim is that our method constitutes a more refined approach to search for the highest-scoring DAG, because it takes advantage from available data in order to dynamically restrict the search space across the evolution.

We already saw in Section 5.4.2 how the Elite set $\mathcal{E}$, already used by the Site-specific Rate approach, is exploited via the population disorder entropy metric so to provide the algorithm with a exploration/exploitation ratio dynamic monitoring method. The Elite set $\mathcal{E}$, being the best we can get at any generation across the evolution, is indeed a valuable resource that can be employed in this context as well.

By following an incremental approach in the design of our hybrid methods, we present in this section, in Algorithm 4, an extended version of DiG-SiRG method denoted as *Parent Reduced Diversity Guided Site-specific Rate Genetic* (PaRe-DiG-SiRG) algorithm.

All the information we need is enclosed in the PWM, which is built during the CONSTRUCT-PWM procedure: this is why before that first PWM instance is built the used parent limitation routine is still the basic version, and afterwards all calls to the LIMIT-PARENTS routine are replaced by calls to the new ELITE-GUIDED-LIMIT-PARENTS procedure – as shown (at lines 13 and 21) in Algorithm 4.

As previously done with the other algorithms, the new function will be explained in detail below by referring to its line number in Algorithm 4. For the reader's convenience, the newly introduced procedure with respect to its basic method (i.e. the DiG-SiRG algorithm) is indicated in bold in the pseudocode.

Algorithm 4: Parent reduced diversity guided site-specific rate genetic algorithm.

```
Input:  $SS$ – complete super-structure
        $data$ – dataset
        $N$ – population size
        $M$ – maximum number of generations
        $MP$ – maximum fan-in
        $scoring\_fn$ – scoring function
        $\alpha$ – elite eligibility threshold
        $\epsilon$ – small positive number
        $\mathbf{d_1}$ – healthy diversity interval at generation 1
        $\mathbf{d_M}$ – healthy diversity interval at generation $M$
Output: $\mathbf{x}^*$ – elite individual
1  $P \longleftarrow$ Init-Population($SS$, $2N$)
2  $P_{dag} \longleftarrow$ Make-DAG($P$)
3  $P_{dag} \longleftarrow$ Limit-Parents($P_{dag}$, $MP$)
4  $score \longleftarrow$ Score-DAG($data$, $P_{dag}$, $scoring\_fn$)
5  for $i = 1 : M$ do
6      $\mathbf{x}^* \longleftarrow$ Get-Elite($score$, $P_{dag}$)
7      $[P_{dag,1},\ score_1] \longleftarrow$ Selection($P_{dag}$, $score$)
8      $P_2 \longleftarrow$ Crossover($P_{dag,1}$)
9      $P_{dag,2} \longleftarrow$ Make-DAG($P_2$)
10     if $i == 1$ then
11         $P_{dag,2} \longleftarrow$ Limit-Parents($P_{dag,2}$, $MP$)
12     else
13         $P_{dag,2} \longleftarrow$ **Elite-Guided-Limit-Parents**($P_{dag,2}$, $MP$, $PWM$)
14     end
15     $P'_{dag} \longleftarrow [P_{dag,1}\ P_{dag,2}]$
16     $\mathcal{E} \longleftarrow$ Form-Elite-Group($P'_{dag}$, $\alpha$)
17     $\alpha \longleftarrow$ Update-Alpha($\mathcal{E}$, $\alpha$, $\mathbf{d_1}$, $\mathbf{d_M}$)
18     $PWM \longleftarrow$ Construct-PWM($\mathcal{E}$)
19     $P'' \longleftarrow$ SiRG-Mutation($P'_{dag}$, $PWM$, $\epsilon$)
20     $P''_{dag} \longleftarrow$ Make-DAG($P''$)
21     $P''_{dag} \longleftarrow$ **Elite-Guided-Limit-Parents**($P''_{dag}$, $MP$, $PWM$)
22     $score'' \longleftarrow$ Score-DAG($data$, $P''_{dag}$, $scoring\_fn$)
23     $[P_{dag},\ \mathbf{x}^*] \longleftarrow$ Place-Elite($score''$, $P''_{dag}$, $\mathbf{x}^*$)
24 end
```

5.5.3 Lines 13, 21: Elite-Guided Parents Reduction

The function ELITE-GUIDED-LIMIT-PARENTS is a simple strategy adopted to deal with parent reduction, that employs what is already provided in the base algorithm, i.e. the Elite-related PWM.

Given a node, first all its parents indexes are stored as keys in a key-value vector, referred to as *Parent Weight Vector*, PWV; next the routine searches in the PWM for allele and locus positions related to the specific oriented edge of the type (parent $\rightarrow$ current node), retrieves its value, i.e. the *probability of occurrence* of that parent-node edge, and stores it at the respective parent location in the PWV. Then the vector is sorted with respect to the values in increasing order, and finally edges related to the first $NP - MP$ parents in the PWV (found by means of the indexes, i.e. the keys) are removed.

In this way we can say that the Elite set, constructed on the basis of data and evolutionary progress up to that point, constitutes the selector that decides the worst edges to remove among the available ones.

Moreover, it may happen that all the PWM values related to the set of edges between a node and each of its current parents are exactly the same (e.g. 0): in this case the basic random picking strategy (used in previous algorithms) is enabled for that particular node of that specific individual.

5.5.4 Elite-Guided Parents Reduction Approach: an Example

For the sake of clarity a practical, naïve example of application of the new parents reduction technique is provided in this section.

Let us suppose to apply the PaRe-DiG-SiRG algorithm to learn the ASIA BN with a population size N of 7 (we consider such a low value in order to keep things simple) and a maximum number of parents MP equal to 3, and let us consider a generic situation that could occur at any generation across the evolution process, in particular the situation depicted in Figure 3: as we can observe in the representation, we can locate the execution point of this hypothetical run at the beginning of line 21 in Algorithm 4, just before the application of the novel ELITE-GUIDED-LIMIT-PARENTS procedure.

As shown in the representation, let us suppose that the elite set $\mathcal{E}$ has been previously defined by the FORM-ELITE-GROUP function as the set constituted of individuals 1,2 and 3.

Moreover, since parents reduction procedure is applied independently on each node of each individual, let us consider for simplicity the application of the new technique to one node only in all individuals: in this example we focus on the variable $X = 6$.

As a first operation in the cycle referring to each node, the ELITE-GUIDED-LIMIT-PARENTS procedure builds the PWV of the node.

In Table III it is represented the Parent Weight Vector (PWV) for node 6 related to the current situation: we can observe that parents recurring more across elite individuals are char-

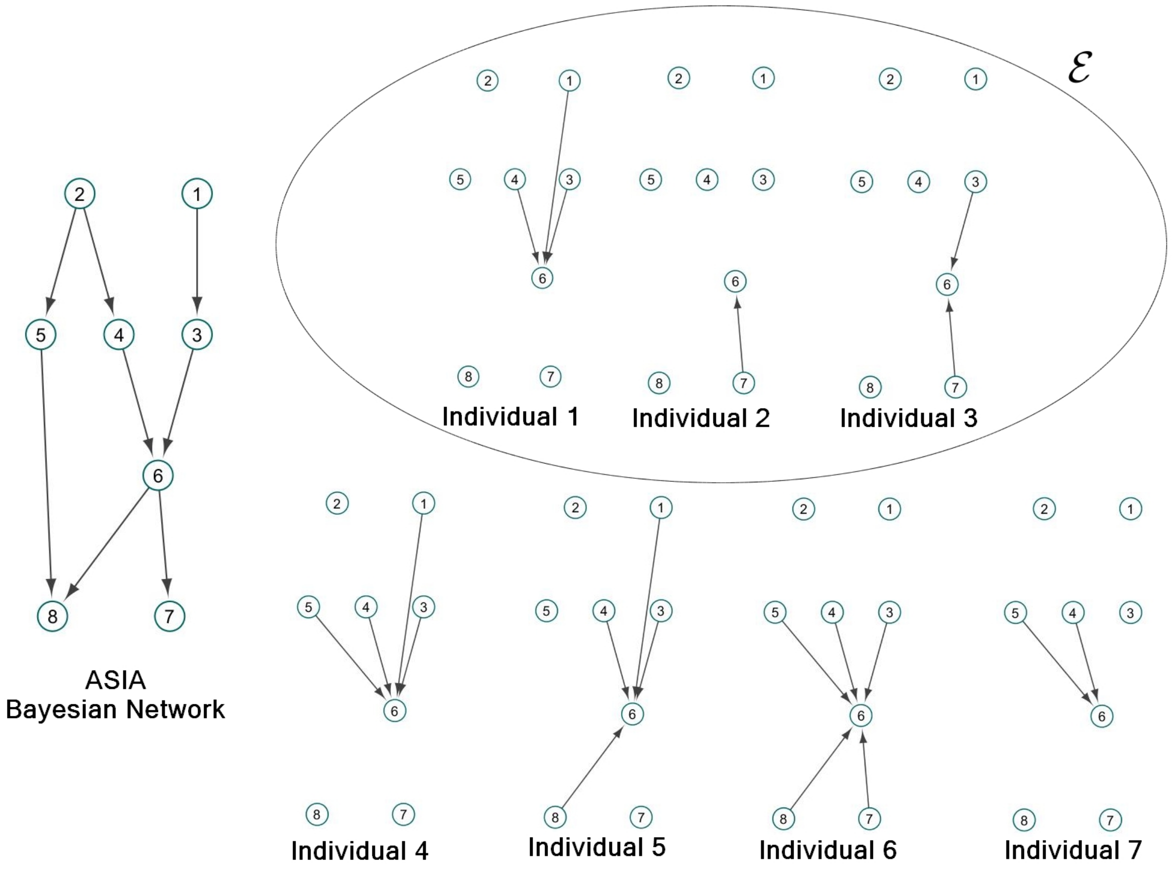


Figure 3: Elite-guided parents reduction example: situation before parents limitation.

acterized by an edge towards the node having higher weights.

When reduction time comes, the parents set size of the node is compared with the MP threshold, for each individual: if the threshold is overcome for any individual, then a parent per time is deleted from the parents set of that individual on the basis of the PWV.

TABLE III: PARENT WEIGHT VECTOR FOR NODE 6 DERIVED FROM THE SITUATION DEPICTED IN THE EXAMPLE, INCREASINGLY ORDERED WITH RESPECT TO THE WEIGHT.

Parent Node	**Edge Weight**
2	0.00
5	0.00
8	0.00
1	0.33
4	0.33
3	0.67
7	0.67

In the example, none of elite individuals is affected by reduction, contrarily to individuals 4, 5 and 6, containing respectively 4, 4 and 5 parents for node 6, a fan-in that is higher than MP value (3).

Parents reduction is finally applied on each individual by simply parsing the PWV and, whenever the parent related to current PWV slot is found in the parents set of the node, by detaching it from the node, until the parents set size becomes equal to the threshold MP. Parents limitation stage is portrayed in Figure 4, where deleted edges related to detached parents are drawn in red.

As can be seen in the example, the proposed parents reduction technique, if provided with sufficiently reliable data, is able to decide in an informed way which parents to retain for any node of any individual, at a generic iteration of the evolutionary process.

As we can ascertain in Section 4.5.2, the proposed method is not overcome by the baseline random parents reduction approach in any conducted experiment, and results to be as efficient and performing as the input dataset size grows, i.e. as it becomes more informative.

5.6 "Self Parent Reducing" Enhancement

As shown in Section 5.5 and as we will ascertain in Section 6, we formulated a new, reliable method to efficiently reduce the number of parents for each node in a DAG during its evolution. In this subsection we will propose an enhancement to the new method that attempts to take advantage from the safe parent limitation procedure presented in Section 5.5.3, so to optimally reduce the search space size and to produce a method insensitive to MP parameter setting.

In particular, if we know a sufficiently reliable value for the maximum fan-in per each node in a network to learn, then we can bound any evolving DAG to keep the number of parents per node below that threshold: as a result, the population will be eligible to explore only a portion of the overall search landscape related to the complete input super-structure, and this means that the efforts of each individual will be focused on a more restricted and direct search space.

If we can rely on a particular MP value, supposed to be a minimum bound near enough the actual maximum number of parents in the target network, then the PaRe-DiG-SiRG method would be our first choice; on the other hand, if we are not able to get such knowledge, we propose a further method able to utilize again the Elite set in order to optimally adjust the

MP value so to constrain the search space as much as possible.

The strategy we propose involves a set of n node-specific MP values (one per each node), denoted as $\mathbf{MP}$, utilized in a dynamic and adaptive approach: related method, named *Self Parent Reducing DiG-SiRG* (SPaRe-DiG-SiRG) algorithm, is described in Algorithm 5, whereas details of it are shown below; specifically, the only relevant variation with respect to the PaRe-DiG-SiRG method is given by the addition of the new UPDATE-MP function.

As previously done with the other algorithms, the new function will be explained in detail below by referring to its line number in Algorithm 5. For the reader's convenience, the newly introduced procedure with respect to its underlying method (i.e. the PaRe-DiG-SiRG algorithm) is indicated in bold in the pseudocode.

Since huge computational requirements are still a concern to us, in order to comply with the need to set a maximum number of parents overall, we consider the value MP given as input to the algorithm as a maximum bound for any i^{th} value in the $\mathbf{MP}$ vector.

Algorithm 5: Self parent reducing diversity guided site-specific rate genetic algorithm.

```
Input:  $SS$ – complete super-structure
        $data$ – dataset
        $N$ – population size
        $M$ – maximum number of generations
        $MP$ – maximum fan-in
        $scoring\_fn$ – scoring function
        $\alpha$ – elite eligibility threshold
        $\epsilon$ – small positive number
        $\mathbf{d_1}$ – healthy diversity interval at generation 1
        $\mathbf{d_M}$ – healthy diversity interval at generation $M$
Output: $\mathbf{x}^*$ – elite individual
1  $P \longleftarrow$ Init-Population($SS$, $2N$)
2  $P_{dag} \longleftarrow$ Make-DAG($P$)
3  $P_{dag} \longleftarrow$ Limit-Parents($P_{dag}$, $MP$)
4  $score \longleftarrow$ Score-DAG($data$, $P_{dag}$, $scoring\_fn$)
5  for $i = 1 : M$ do
6      $\mathbf{x}^* \longleftarrow$ Get-Elite($score$, $P_{dag}$)
7      $[P_{dag,1},\ score_1] \longleftarrow$ Selection($P_{dag}$, $score$)
8      $P_2 \longleftarrow$ Crossover($P_{dag,1}$)
9      $P_{dag,2} \longleftarrow$ Make-DAG($P_2$)
10     if $i == 1$ then
11         $P_{dag,2} \longleftarrow$ Limit-Parents($P_{dag,2}$, $MP$)
12         $\mathbf{MP} \longleftarrow [MP\ MP\ \ldots\ MP]_n$
13     else
14         $P_{dag,2} \longleftarrow$ Elite-Guided-Limit-Parents($P_{dag,2}$, $\mathbf{MP}$, $PWM$)
15     end
16     $P'_{dag} \longleftarrow [P_{dag,1}\ P_{dag,2}]$
17     $\mathcal{E} \longleftarrow$ Form-Elite-Group($P'_{dag}$, $\alpha$)
18     $\alpha \longleftarrow$ Update-Alpha($\mathcal{E}$, $\alpha$, $\mathbf{d_1}$, $\mathbf{d_M}$)
19     $PWM \longleftarrow$ Construct-PWM($\mathcal{E}$)
20     $P'' \longleftarrow$ SiRG-Mutation($P'_{dag}$, $PWM$, $\epsilon$)
21     $\mathbf{MP} \longleftarrow$ **Update-MP**($n$, $\mathbf{MP}$, $MP$, $\mathcal{E}$)
22     $P''_{dag} \longleftarrow$ Make-DAG($P''$)
23     $P''_{dag} \longleftarrow$ Elite-Guided-Limit-Parents($P''_{dag}$, $\mathbf{MP}$, $PWM$)
24     $score'' \longleftarrow$ Score-DAG($data$, $P''_{dag}$, $scoring\_fn$)
25     $[P_{dag},\ \mathbf{x}^*] \longleftarrow$ Place-Elite($score''$, $P''_{dag}$, $\mathbf{x}^*$)
26 end
```

5.6.1 Line 21: Node-specific Maximum Parents Threshold Adaptive Reduction

During the evolution, at each generation every individual's node is characterized by a certain number of parents; ideally, throughout the evolution each node's parents set eventually converges to the actual nodes' parents set in the target DAG.

Convergence speed could depend, among other factors, on the maximum parents threshold: indeed, given the actual number of parents of a node in the target DAG, NP, and a maximum parents threshold, MP, if $MP > NP$ it is trivial to conclude that, at any generation, even if the parents set for that node already includes optimal (hopefully actual) parents, this set will be continuously altered with the addition or deletion of other nodes as parents.

The Update-MP procedure can serve for helping the above-mentioned convergence process: at each generation this procedure decides whether to increase or decrease by an unitary step the $MP(i)$ threshold (related to the i^{th} node), or keep it unchanged, for each node.

In the case the actual number of parents for a node is consistently lower with respect to the MP threshold, the *Elite set* $\mathcal{E}$ at each iteration can dynamically adjust it, in this case by means of a unitary decrement, until the threshold stabilizes.

On the other hand, if evolution leads the Elite set to decrease the $MP(i)$ threshold too much, i.e. below the actual number of parents for the i^{th} node, the procedure is still reversible and leaves enough freedom to the Elite set to recover from its errors and possibly raise the threshold when it needs to.

In any case, the dynamic threshold $MP(i)$ is not allowed to exceed the starting overall threshold MP, provided as input to the SPaRe-DiG-SiRG algorithm.

The method through which the Elite set in its entirety can decide if and how to modify the $MP(i)$ threshold for the i^{th} node of each individual at any generation is enclosed in Equation 5.9: in brief, it consists in an adaptive adjustment strategy in which each node-specific threshold at each step independently follows the value of a time-varying reference threshold, referred to as the *Safety Threshold* S_{th}.

$$\begin{cases} S_{th}(i) = 1 + E_{nP}(i) \\ MP(i) = \begin{cases} MP(i) - 1, & \text{if } MP(i) > S_{th}(i) \\ MP(i), & \text{if } MP(i) = S_{th}(i) \\ MP(i) + 1, & \text{if } MP(i) < S_{th}(i) \end{cases} \end{cases} \tag{5.9}$$

$E_{nP}(i)$ denotes the *cardinality of the set given by all distinct parents of the* i^{th} *node over all individuals in the Elite set* $\mathcal{E}$ at the current generation: this member in the equation allows all elite individuals to take part to the MP adjustment choice.

If the problem could be solved by a node-specific threshold reduction only, we would need just $E_{nP}(i)$ as reference to compare with current $MP(i)$ value, but in fact we need to provide this threshold variation strategy with the possibility to raise the $MP(i)$ limit when it is needed, i.e. when the evolution process brings at least one of elite individuals to get one more parent (or several additional ones) for the i^{th} node. We included E_{nP} in the safety threshold S_{th} expression as its main contribution, but, in order to address the above-mentioned need, we should augment its definition.

If in the Elite set it happens that the same value for the number of parents of i^{th} node is kept and if we consider the i^{th} Safety Threshold $S_{th}(i)$ as equal to $E_{nP}(i)$ only, then the equation does not give the chance to increase the number of parents for the i^{th} node to any elite individual. Since we don't want to affect in any way the evolution process, we included in the safety threshold expression a constant additional unitary value, so to provide the evolutionary process with total freedom even when all elite individuals have exactly the same value as the number of parents for current node.

The presented strategy aims to enhance any Elite-driven genetic method with a dynamic node-specific MP threshold so to reduce the search space as much as possible at each generation and help convergence, but at the same time by providing the parents acquisition process of Elite DAGs with enough freedom and adaptiveness so to not affect the evolutionary process itself.

5.6.2 Node-specific Maximum Parents Threshold Adaptive Reduction: an Example

For the sake of clarity a practical, naïve example of application of this adaptive variation to the parents reduction process is provided in this section.

Let us suppose to be in the same situation portrayed in the example at Section 5.5.4, with a slight variation for the MP value: the PaRe-DiG-SiRG algorithm is applied to learn the ASIA BN with a population size N of 7 (we consider such a low value in order to keep things simple), until evolution gets to a generic situation that could occur at any generation across the evolutionary process, in particular the situation depicted in Figure 3; in this case, let us consider a slight variation with respect to the example in Section 5.5.4: let us assign to MP a lower value of 2 instead of 3.

Given that the proposed parents reduction technique is driven by an informative environment (defined by data), we should take into account that the same environment can also be deceptive, especially in data scarcity situations: if we use the PaRe-DiG-SiRG algorithm with a too low value for MP, it is more likely that the number of wrong choices decided by the elite set (or rather the PWV) for what concerns parents reduction relevantly grows at a point that the evolutionary process results to be affected.

Consequently, by following the Elite-Guided-Limit-Parents procedure in our example, on the basis of related PWV (reported in Table IV for the convenience of the reader), the parents

TABLE IV: PARENT WEIGHT VECTOR FOR NODE 6 DERIVED FROM THE SITUATION DEPICTED IN THE EXAMPLE, INCREASINGLY ORDERED WITH RESPECT TO THE WEIGHT.

Parent Node	**Edge Weight**
2	0.00
5	0.00
8	0.00
1	0.33
4	0.33
3	0.67
7	0.67

reduction technique results in the situation depicted in Figure 5: in this occasion, individuals reduced with respect to $X = 6$ are individuals 1, 4, 5 and 6.

As we can observe from Figure 5, in opposition to previous case depicted in Figure 4, a slight decrease on MP threshold may lead the parents reduction process to undertake wrong decisions about nodes parenthood: indeed node 6 of sixth individual results in this case deprived of one of its actual parents (node 4), whereas node 7 is erroneously kept as potential parent of the node.

The proposed variation to the definition of MP threshold, enclosed in the SPaRe-DiG-SiRG algorithm (specifically in the UPDATE-MP function), may help to overcome wrong decisions by dynamically adjusting the threshold itself, because it takes into account elite set uncertainty about the parents set of any node.

Let us suppose to apply the SPaRe-DiG-SiRG method to the presented example, so that at current generation the value in the **MP** vector referred to node 6 results equal to 2, i.e. $\mathbf{MP}(6) = 2$: we can thus locate the execution at line 21 of Algorithm 5, before the application of the parents reduction routine and just before MP update.

When the execution of the UPDATE-MP function starts, first we need to compute the cardinality of the set given by all distinct parents of the 6^{th} node over all individuals in the Elite set $\mathcal{E}$, denoted by $E_{nP}(6)$: as we can see in Figure 3 the above-mentioned set is given by $1, 3, 4, 7$, and thus $E_{nP}(6) = 4$, i.e. the size of the set.

We can now define the Safety Threshold $S_{th}(6)$ as $S_{th}(6) = 1 + E_{nP}(6)$, and thus we have $S_{th}(6) = 5$. By following Equation 5.9, it results that the MP threshold for node 6, being lower than $S_{th}(6)$, is incremented from 2 to 3.

Thanks to this adjustment, the ELITE-GUIDED-LIMIT-PARENTS procedure (consequently executed at line 23 in Algorithm 5) avoids to undertake too hazardous decisions that may lead to the situation depicted in Figure 5, and results to apply a safer parents reduction as in Figure 4.

As can be seen in the example, the proposed parents reduction enhancement allows to automatically and dynamically decide the maximum fan-in value for each node over all individuals in the population in a way that is supposed to not affect the overall evolutionary process.

The proposed technique has the peculiarity to be knowledge-driven (since it is based on Elite parents sets information) and is provided with a recovery feature that allows any elite individual to raise the MP threshold for a particular node in the case a new parent is found for that node.

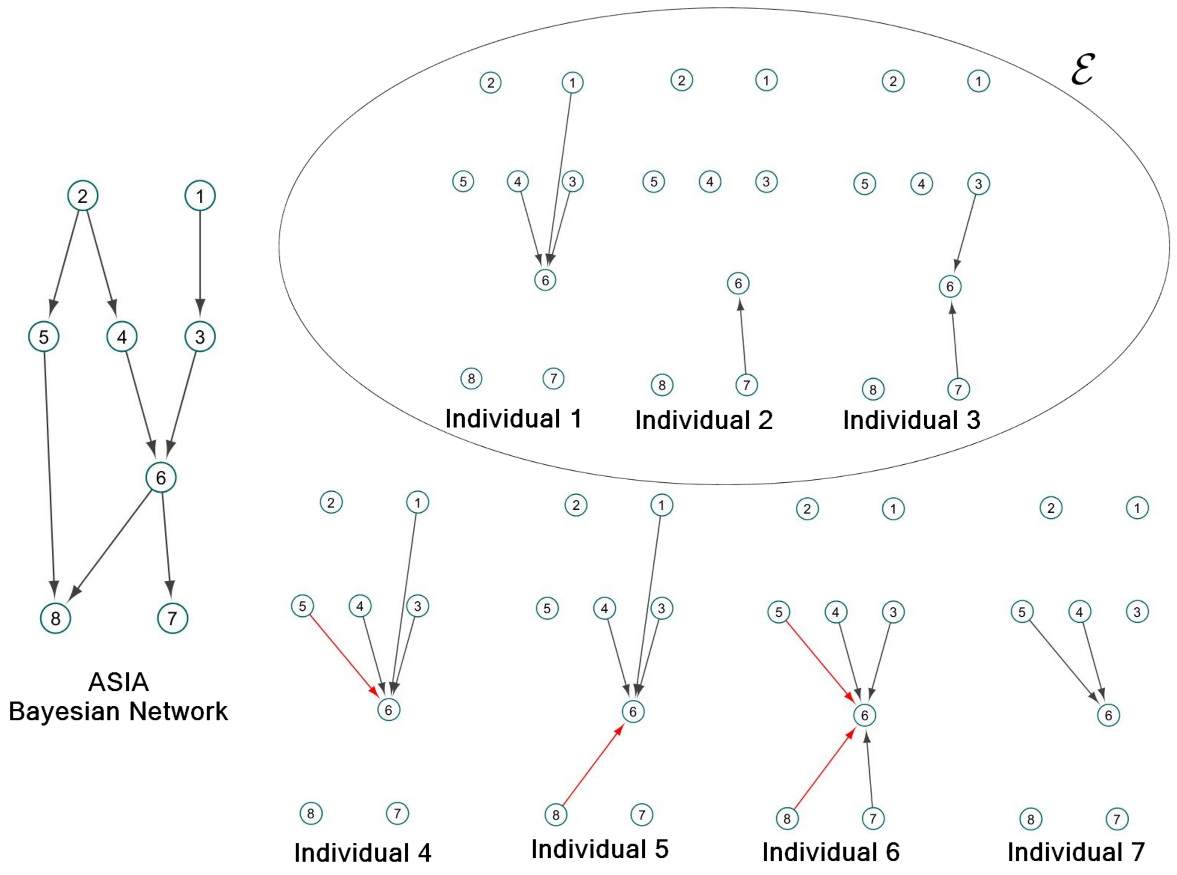


Figure 4: Elite-guided parents reduction example: during the limitation procedure, worst parents are identified and related edges (depicted in red) are deleted, for each individual.

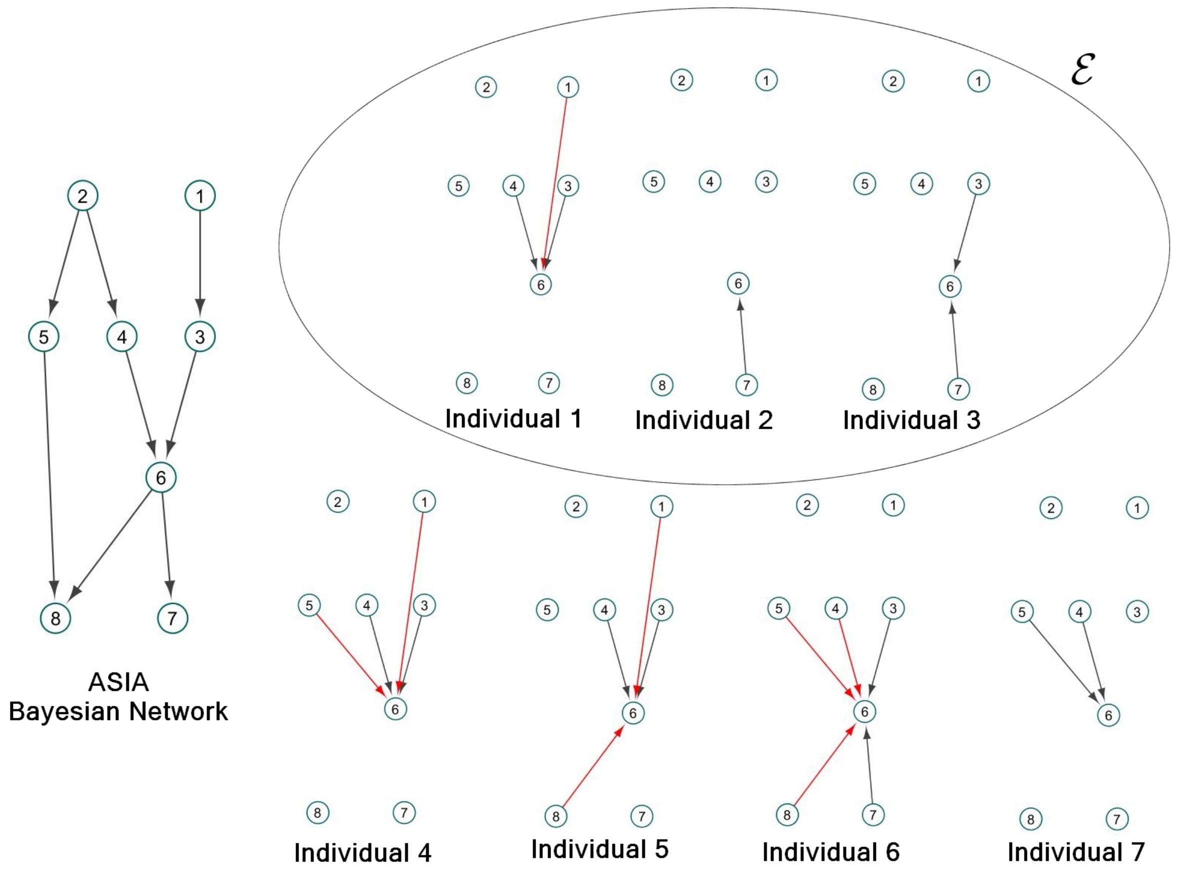


Figure 5: Elite-guided parents reduction, example with a lower threshold: result of the reduction process.

CHAPTER 6

EXPERIMENTS AND RESULTS

In this chapter it is described the experimental environment we used to evaluate presented methods and results of several experiments are reported in terms of a set of relevant performance metrics.

In particular, in the first place the set of algorithms included in the benchmark is presented, together with all the details about simulation settings and employed datasets. Then two experiments aimed at identifying the best performing method for the CB phase are reported. Finally, a series of statistics and experiments developed in an incremental manner, similarly to the inner structure of introduced genetic algorithms themselves, is described and provided with all relevant results in order to validate all claims stated in previous chapters of this thesis.

1 Compared Algorithms

In order to make our methods deal with a systematic and fair comparison with other competitor approaches, this thesis work involves a rich benchmark constituted by a series of well-known and widely-used structure learning algorithms. All of them are described in detail in Chapter 2, and have been implemented within the BNT-SLP package [108] for experimental use. We will briefly describe them in this section, for the convenience of the reader.

PC by Spirtes [7] is a traditional CB method that starts with the complete graph and is articulated in two phases: first it is executed a sequence of loops, each characterized by an increasing index n; within each loop all possible combinations of node pairs and related condition sets of cardinality n are evaluated by means of CI tests, and, in case the test is positive, related edge is deleted; the sequence of loops stops when all sets of neighbors of each pair of nodes is of cardinality lower than n; in the second part of the method edges are directed on the basis of v-structures identification and induction rules.

Maximum Weight Spanning Tree (MWST) by Chow and Liu [56] attempts to find on a S&S basis a tree that maximizes the data likelihood; in particular, first each possible edge of the graph is provided with a weight defined by the mutual information between the two nodes of the edge, then a maximum weight spanning tree algorithm (such as Kruskal or Prim methods) is applied so to finally get a sub-optimal structure in a greedy fashion.

Tree-Augmented Naïve Bayes method by Friedman et al. [81] is an extension to the MWST approach specialized in building a tree augmented network based on a given class node. First it is assigned a weight to each edge of the complete graph on the basis of a conditional mutual independence scoring method, conditioned on the class node; then the MWST algorithm is executed so to obtain the final DAG, or rather the final tree.

K2 algorithm by Cooper and Herskovits [63] is a popular S&S greedy approach employing the *K2 metric*, reported in Section 1. The K2 search begins by assuming that a node has no parents and then greedily selects as its parents the variables from a given ordering whose addition best improves the score of the resulting structure in an incremental fashion, until the score stops to increase. It needs a node ordering as input: since we assume to not have available any information about node ordering, we fed the algorithm with a random permutation of nodes sequence directly generated at each trial.

Hill Climbing by Buntine [82] is a local S&S method for BN structure learning: at each step the algorithm considers all available local operations and chooses the one that yields the best improvement. Specifically, at each step the algorithm computes the differences in the overall score with respect to all possible local arc operations (addition, deletion and reversal) and chooses the one with the highest positive difference; the process is repeated until score improvement occurs.

Hybrid Structure Learner using Genetic Algorithm (HSL-GA) by Vafaee [15] is provided in this thesis work (with some minor variations) as the hybrid conjunction of the *Opt0SS* algorithm, presented in Section 4, and the first of our presented S&S methods, defined in Section 5.3 as *Standard GA*. Variations are given by: the adoption of the G^2 *likelihood-ratio test* instead of *Pearson's* χ^2 *test* for CI evaluation in the CB part, a *proportional* selection instead of a *rank* selection strategy and a *uniform* crossover in place of a *single-point* crossover approach, for what concerns the S&S part.

2 Benchmark Specification

In this section we devise the experimental environment that was employed for the evaluation of our methods: first we focus on the set of problem instances to tackle, i.e. we describe the collection of chosen target BNs; then we define default, general parameters for use with all benchmark algorithms; finally details on generation and composition of sample datasets used in the experiments are provided.

2.1 Bayesian Networks Selection

Given that a generic BN structure learning method may yield varying performance results with respect to variations in network and/or dataset sizes, we decided to select a set of four networks of various sizes as benchmark problems, each with a set of associated datasets of assorted amounts.

Here are described in details the four tested networks, chosen from Bayesian Network Repository [116]; they are provided with a three-elements tuple reporting $[|\mathbf{V}|, |\mathbf{E}|, \mathbf{NP_{max}}]$, i.e. [**number of nodes, number of edges, maximum number of parents**].

Declared sizes are based on BN Repository categorization.

- **ASIA [8,8,2]** is a small BN also known as LUNG CANCER, portrayed by Lauritzen et al. in [100]: it describes a simple expert system designed for medical diagnosis, useful in particular to diagnose tuberculosis, lung cancer or bronchitis on the basis of the occurrence of three causal factors given by dyspnoea, a recent visit to Asia or smoking. It is represented in Figure 6.
- **INSURANCE [27,52,3]** is a medium BN for car insurance risk estimation reported by Binder et al. in [133]: it is useful to estimate the expected claim costs for a car insurance policyholder. It is represented in Figure 7.

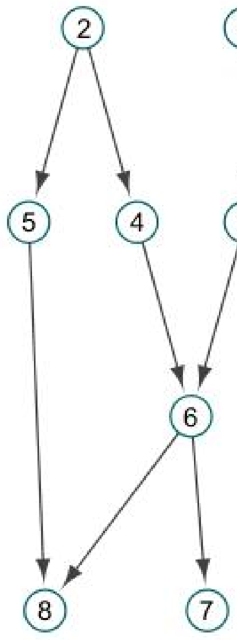


Figure 6: The ASIA Bayesian network.

- **ALARM [37,46,4]** is a popular, medium-sized BN outlining a monitoring system for medical diagnosis purposes, as reported by Beinlich et al. in [117]. Its name stands for *A Logical Alarm Reduction Mechanism*, and refers to a diagnostic application that implements an alarm message system for patient monitoring; it is able to accept a set of physiologic measurements and to generate specific text messages when they are outside of their normal range, so to advise the user of possible problems. It is represented in Figure 8.

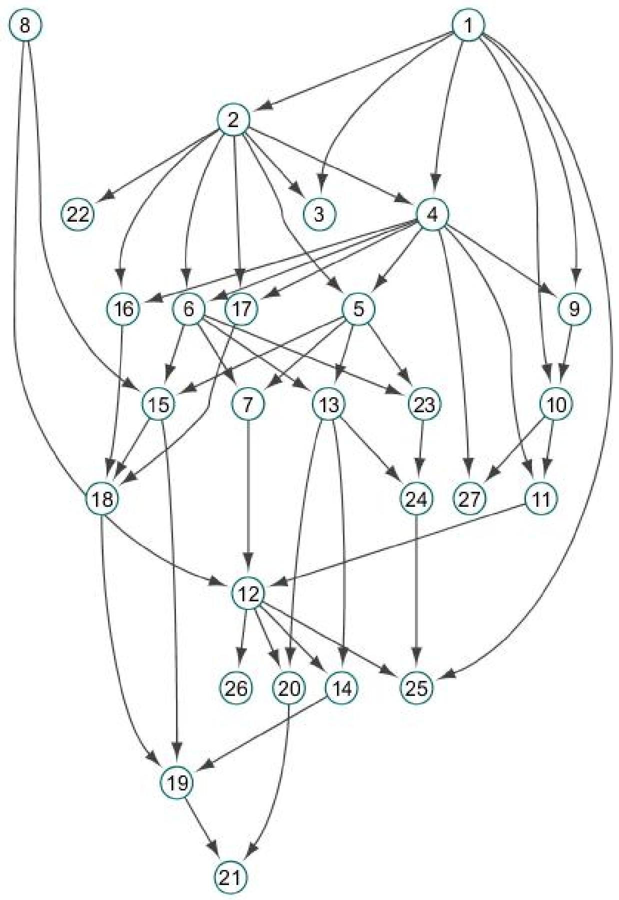


Figure 7: The INSURANCE Bayesian network.

- **HEPAR II [70,123,6]** is a large BN that constitutes a probabilistic causal model for diagnosis of liver disorders, as reported by Onisko in [134]. It constitutes an extension of the original HEPAR project [135]: it involves the HEPAR system, containing a database of patient records of the Gastroenterological Clinic of the Institute of Food and Feeding

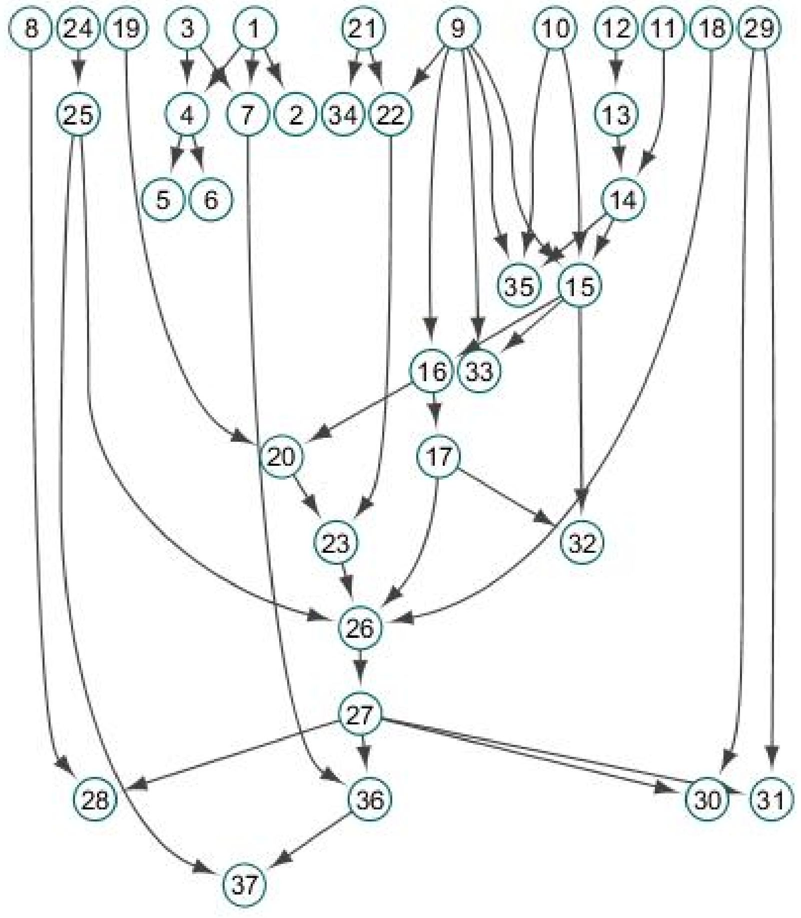


Figure 8: The ALARM Bayesian network.

in Warsaw, currently used in the clinic as a diagnostic and training aid; this BN aims at providing an expert system for medical diagnosis, as well as a means to train physicians. It is represented in Figure 9.

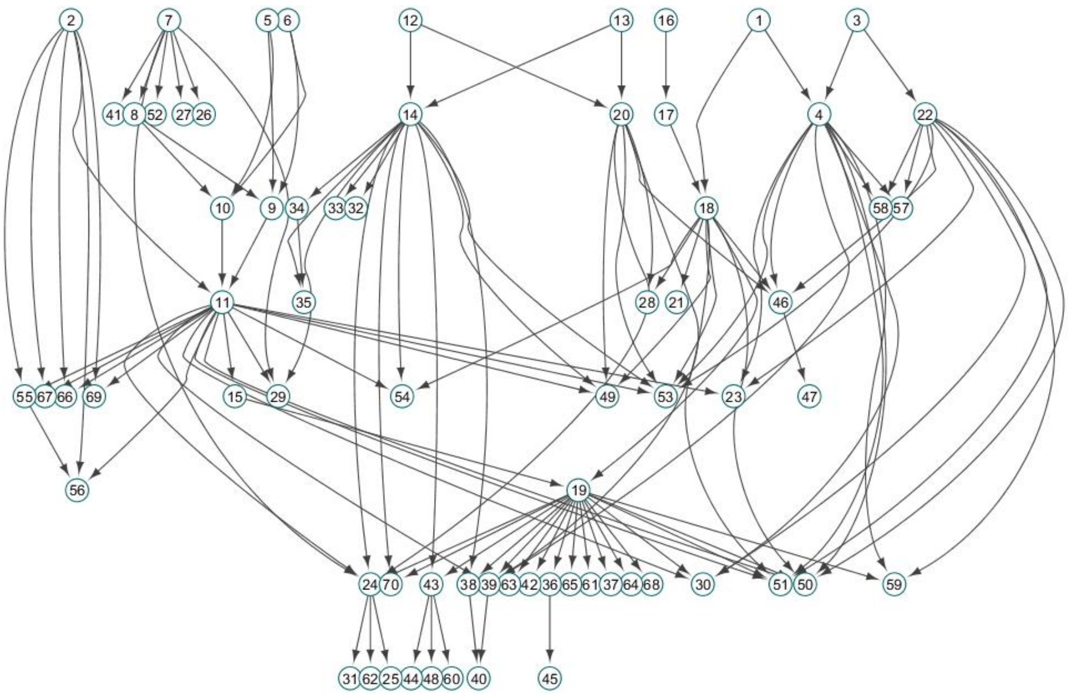


Figure 9: The HEPAR II Bayesian network.

Since the BNs in BN Repository are given in Bayes net Interchange Format (BIF), benchmark networks were first converted to the BNT format, readable by BNT-SLP package, using the *bif2bnt* program developed by Shan [136].

2.2 Simulation Settings

The machine used for the experiments is provided with a Intel i7 [137] CPU of 2.60 GHz, and a RAM of 32.0 GB.

To have a feasible running time and a fair comparison across different methods, all routines are adjusted to stop when either the maximum number of iterations is reached or when the maximum allotted CPU time T is reached, whichever occurs the first.

Based on our resources, T is set to the values reported in Table V, one specifically defined for each tested network.

We chose to use the nonparametric *Wilcoxon signed-rank test* [138] in order to validate statistical significance in the comparison between two methods results.

As Gibbons and Chakraborti explain in [139], "the Wilcoxon signed rank test is a nonparametric test for two populations when the observations are paired. In this case, the test statistic, W, is the sum of the ranks of positive differences between the observations in the two samples."

Basically, provided that the *p-value* is defined as the probability that the null hypothesis is true [140] and given two data vectors $\mathbf{x}$ and $\mathbf{y}$, this procedure returns the *p-value* of a paired, two-sided test for the null hypothesis that $\mathbf{x} - \mathbf{y}$ comes from a distribution with zero median; if resulting p-value is below some predefined threshold called *significance level*, it is possible to reject the null hypothesis and thus conclude that the results improvement yielded by the best performing method is statistically significant.

In our benchmark the same settings for free parameters that are common between two or more methods are adopted: used default settings are reported in Table V. The parameters in the rest of compared algorithms are all set to their default values (see [108] for details).

TABLE V: DEFAULT PARAMETER VALUES USED IN ALL BENCHMARK METHODS.

Parameter	**Variable**	**Value**
ASIA Max CPU Time	`T_as`	200
INSURANCE Max CPU Time	`T_in`	1000
ALARM Max CPU Time	`T_al`	4000
HEPAR II Max CPU Time	`T_he`	12000
CI Test	`CI_test`	G^2 likelihood-ratio
CI Significance Level	`CI_th`	0.01
Scoring Function	`scoring_fn`	Bayesian (BDeu)
Maximum Number of Parents	`MP`	12
Statistical Significance Level	`ss_th`	0.05

2.3 Dataset

The `sample_bnet` function implemented in BNT [16] has been used to generate synthetic datasets of various sizes: exact sizes employed in the experiments are reported in Table VI.

In particular, each dataset is not generated from scratch every time: in order to ensure a fair comparison among benchmark methods, the same instances of 30 randomly generated datasets were stored and repeatedly used (in their entirety or partially) in all experiments.

TABLE VI: TEST CASES AND DATASET SIZES TAKEN INTO ACCOUNT IN OUR BENCHMARK.

Network	**Network Size**	**Datasets Sizes**
ASIA	8	50, 1000
INSURANCE	27	50, 100, 1000
ALARM	37	30, 50, 70, 100, 1000
HEPAR-II	70	100, 1000

2.4 Dataset and Network Sizes in the Literature

In all publications reported in Chapter 2, the authors try to apply their methods to learn networks of different sizes with variously-sized datasets.

In order to realize where past work focused its efforts and where this thesis work aims to, in Table VII we reported the benchmarks taken into account by other authors and by us.

Mentioned benchmark networks belong to the Bayesian Network Repository [116] and to the UCI Repository of Machine Learning Databases [141].

Accordingly to Bayesian Network Repository [116], a small-sized network has less than 20 nodes, a medium-sized one has between 20 and 60 nodes whereas a large network has more than 60 nodes.

For what concerns dataset sizes, the literature qualitatively defines the same amounts of samples as being "small-," "medium-" or "large-"sized, depending on the case.

As we can deduce from Table VII, this thesis work attempts to tackle the task by considering relatively small sample sizes with respect to the other experiments reported in the literature: only GS, Opt01SS, K2, Carrillo's, μGP and HSL-GA experiments took into account datasets of comparable sizes (relatively to similarly sized networks) to those under analysis in this work.

3 Super-Structure Construction

In order to restrict the huge search landscape we deal with, we need to focus our attention on a subset of the totality of DAGs extractable from the complete graph on n nodes: the basis of a Hybrid approach such as ours is indeed to take into account the set of those edges in the complete graph that we suppose may contain an actual oriented edge in the target DAG.

We chose to estimate the performance of the *Opt01SS* algorithm: by means of a series of unconditional and conditional independence tests it is supposed to yield a sufficiently reliable super-structure, that should hopefully contain our target DAG.

TABLE VII: A COMPARISON AMONG NETWORK AND DATASET SIZES IN LITERATURE BENCHMARKS.

Algorithm	**Network (#Nodes)**	**Dataset**
PC	ALARM (37)	2000–10000
TPDA	ALARM (37), HAILFINDER (56)	1000 to 10000, 2500 to 20000
GS	ALARM (37), INSURANCE (27), HAILFINDER(56), DIABETES (413)	100 to 500, 500, 500, 500
RAI	ALARM (37)	10000
Opt01SS	ALARM (37), CHILD (20), INSURANCE (27), HAILFINDER (56)	250–500–1000–10000 each
K2	ALARM (37)	100–200–500–1000–2000–3000–10000
CB	ALARM (37), LED (8), LETTERS (17), SOYBEAN (36)	10000, 199, 10000, 683
TAN	CHESS (36), FLARE (10), GERMAN (20), SATIMAGE (36), . . .	2130, 1066, 1000, 4435
SC	ALARM (37)	10000
OR	ADULT (15), ALARM (37), LETTERS (17), . . .	49000, 20000, 20000
GES	MSWEB (50), HOUSEVOTES (17), MUSHROOM (22), . . .	5000, 435, 8124
OS	ALARM (37), DIABETES (413)	100–1000–10000–20000, 10000
Carrillo's	ALARM (37)	300 to 10000
Larrañaga	ASIA (8), ALARM (37)	500–1000–2000–3000 each
K2GA	ALARM (37)	3000
Chain-Model GA	ASIA (8), CAR (18), ALARM (37)	5000, 10000, 3000
Carvalho's CCGA	ALARM (37), INSURANCE (27)	1000–3000–5000 each
μGP	ALARM (37)	100
Wong's CCGA	ALARM (37), PRINTD (26)	1000–2000–5000–10000, 5000
HSL–GA	ASIA (8), ALARM (37), HEPAR-II (70)	50, 30–50–70–100, 100
This Work	ASIA (8), ALARM (37), INSURANCE (27), HEPAR-II (70)	50–1000, 30–50–70–100–1000, 50–1000, 100–1000

Even if the Opt01SS algorithm claims to return a *sound SS* [60] (i.e. any PDAG that contains the *skeleton* of the target DAG), we decided to compare its performance with respect to a simple application of 0^{th}-order CI tests to all pairs of nodes in the graph and consequent edges removal on node pairs characterized by independence: in order to deal with an analogy, we denote this simple method as *Opt0SS*. Opt01SS and Opt0SS algorithms are deterministic: by feeding them with the same dataset as input, they will always yield the same super-structure.

The *CI significance level* was set to 0.01 in all experiments.

All experiments involve a benchmark constituted by all available test cases for ALARM and INSURANCE networks; results are reported as average values over 30 trials for each test case, each provided with related standard deviation over the set of trials. As a matter of fact, we will see in this section how much data availability strongly affects the quality of the final super-structure.

3.1 Performance estimation

We will estimate the quality of obtained super-structures by means of three normalized scoring metrics explained in-depth in [100]: *F_1 score, sensitivity and specificity.*

In order to adapt the above-mentioned metrics to this context, we make the following assumptions for what concerns a super-structure with respect to the target DAG, given any pair of vertices (A, B):

- a *true positive* (TP) occurrence is identified as the presence of an undirected edge $(A — B)$ in the super-structure and the presence of the same edge directed as $(A \rightarrow B)$ or $(A \leftarrow B)$ in the target DAG;
- a *true negative* (TN) occurrence is identified as the absence of an edge $(A \not- B)$ in the super-structure and the absence of the same edge $(A \not- B)$ in the target DAG;

- *retrieved instances*, i.e. the sum of true positives and false positives ($TP+FP$), correspond to the set of edges in the obtained super-structure;
- the *set of positives* (P), i.e. the sum of true positives and false negatives ($TP + FN$), is identified by the set of edges that are oriented in the target DAG;
- the *set of negatives* (N), i.e. the sum of true negatives and false positives ($TN + FP$), is identified by the set of edges that are absent in the target DAG.

Provided with the definitions above, we can compute F_1 score, sensitivity and specificity for any obtained super-structure. In particular:

Sensitivity is the proportion of directed edges in the target BN that are associated to undirected edges in the learnt SS;

Specificity is the proportion of absent edges in the target BN that are associated to absent edges in the learnt SS;

Precision is the proportion of undirected edges in the learnt SS that are associated to directed edges in the target BN;

$\mathbf{F_1}$ score is defined as the harmonic mean of sensitivity and precision measures [100].

3.2 Conditional Independence Tests Evaluation

In this section performances related to two different CI tests are compared, so to choose the method that is the most suitable to our task.

We compared the SS outcomes of the basic Opt0SS algorithm when it is enabled with Pearson's χ^2 test with respect to when it is provided with the G^2 likelihood-ratio test: in Figure 10 are reported F_1, sensitivity and specificity scores averaged on the 30 available datasets per each size of both networks; error bars on the bar graphs express standard deviations of results over each set of 30 trials.

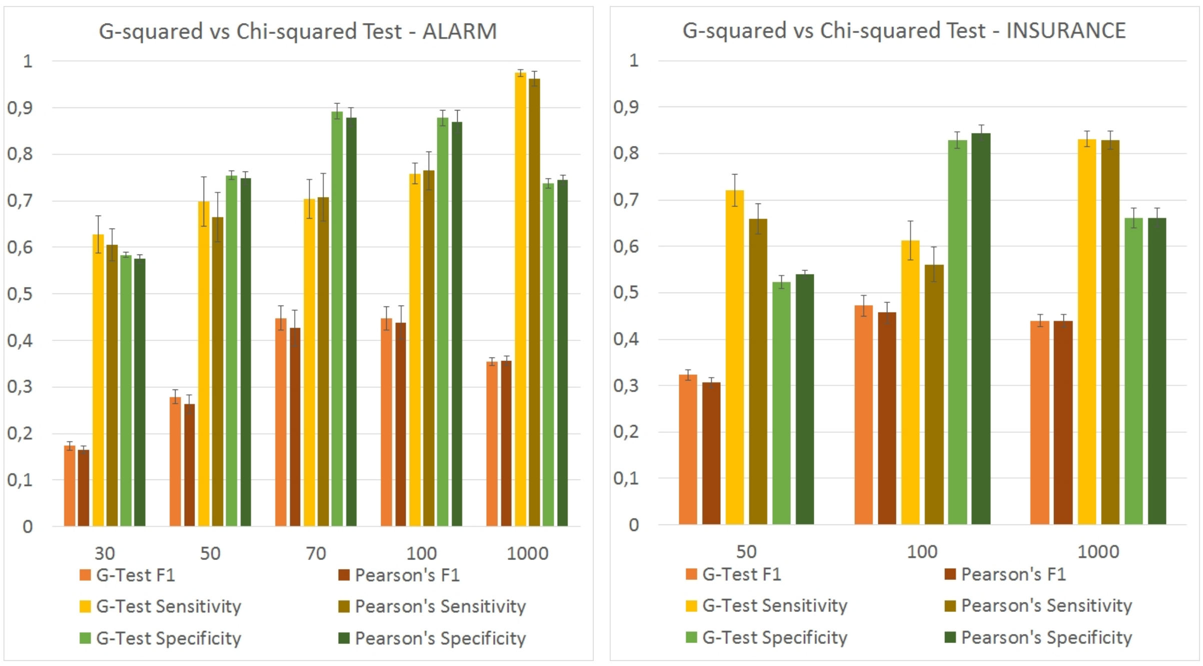


Figure 10: Comparison between Pearson's χ^2 test and G^2 likelihood-ratio test for conditional independence.

In the resulting graphs we can observe what we expected: as stated in [106,107], the G^2 test results to be more reliable with respect to Pearson's χ^2, especially in a data scarcity condition; moreover the G^2 test seems to be characterized by a slightly more deterministic behavior: we can indeed ascertain lower values for G^2 test standard deviations overall. Hence, we decided to make use of the G^2 test in the entirety of this thesis work.

3.3 CB Methods Comparison

Here the target is to achieve a SS that contains as much as possible of the target DAG: more formally it consists in maintaining the false negatives (edges present in the target DAG but not detected neither included in the SS construction) rate as low as possible.

Results are reported in Figure 11.

What we really care of in this context is sensitivity: the higher it is, the lower the false positives rate is; this means that a SS characterized by high sensitivity contains a relevant portion of the final DAG.

As we can observe in both ALARM and INSURANCE resulting trends, sensitivity is always higher with the base method involving only 0^{th}-order CI tests, and F_1 scores are always comparable in experiments conducted with few data (below 1000 test cases). We indeed chose the basic method Opt0SS for use in our Hybrid approach.

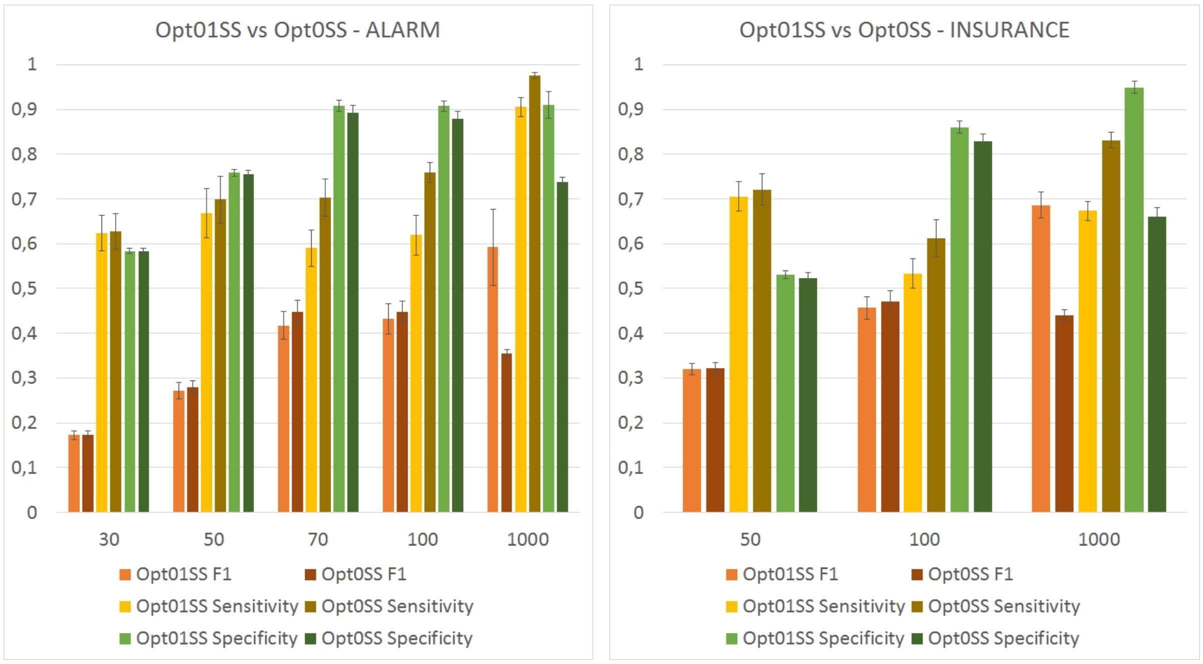


Figure 11: Comparison between the two tested constraint-based methods: Opt01SS and Opt0SS.

3.4 Search Space Reduction: Quantitative Evaluation

In this subsection we will experimentally quantify how much the application of Opt0SS CB method actually restricts the search space, regardless of its soundness (already discussed in previous subsection).

In Table VIII the percentage reported for each test case expresses the ratio between the number of (undirected) edges in the reduced SS and the number of (undirected) edges in the associated complete graph.

Simulation settings are the same of previous experiments: employed benchmark is constituted by all available test cases for ALARM and INSURANCE networks; results are reported in Table VIII as average values over 30 trials for each test case, each provided with related standard deviation over the set of trials, reported between brackets.

TABLE VIII: SPACE REDUCTION ANALYSIS ON INSURANCE AND ALARM NETWORKS.

Test Case	INSURANCE			ALARM				
	50	100	1000	30	50	70	100	1000
% Space Reduction	51.320 (0.951)	23.666 (1.832)	41.206 (1.892)	43.118 (0.726)	27.638 (1.160)	14.855 (1.658)	16.582 (1.580)	31.156 (0.914)

As we can see in Table VIII, it results that adopted CB method succeeds in reducing the search landscape (i.e. the number of edges to take into account for the search) to a portion approximately in the range $[15, 50]\%$ of the original complete structure.

Furthermore, if we look at the same results we can also ascertain that, by moving from small-sized datasets to larger ones, it can be observed first a relative reduction in the search space and then an increase of it.

We can justify this behavior by stating the following:

- relatively high percentages bounded to small datasets can be explained by data scarcity and (as a consequence) data fragmentation, because this phenomenon causes inconsistency on a relevant portion of executed CI tests: whenever a CI test fails, the related edge is kept in the super-structure;
- relatively high percentages bounded to large datasets can find an explanation in ADRs (Approximate Deterministic Relationships); as we reported in Section 1.6, an ADR is a "fortuitous" strong association between two variables, related to the fact that a consistently large portion of data exhibits by accident a *deterministic relation* for those variables: it is indeed reasonable to conclude that the amount of ADRs grows with data availability.

4 Optimal DAG Evolution

In this section it is estimated the performance related to employed genetic methods, and thus to the final learnt DAGs.

The sets of super-structures obtained with the first CB part of the algorithm constitute now the input to each of the proposed strategies: they are described together with their F_1 measures, sensitivities and specificities in Section 3.

Each of the proposed GAs requires a set of parameters: in order to guarantee an adequate comparison among them, a default set of parameters, reported in Table IX, is adopted; in particular, a POI value of 1 indicates that all edges in the input super-structure are randomly oriented in every individual. Any possible variation to this set will be expressed in the description of each experiment.

TABLE IX: DEFAULT SET OF PARAMETERS FOR USE IN THE EXPERIMENTS.

Parameter	**Variable**	**Value**
Population Size	`N`	100
Maximum #Iterations	`M`	100
Probability of Orientation at Initialization	`POI`	1
Elite Eligibility Threshold	`alpha`	0.9
Small Constant in μ equation	`epsilon`	0.01

In order to adjust for the stochasticity effect of datasets, especially relevant in small-sized ones, and to obtain more reliable performance measures, all experiments were run on the first 20 datasets extracted from the pre-built sets, described in Section 2.3; error bars on the bar graphs express standard deviations of resulting statistics over each set of 20 trials.

4.1 Performance estimation

We will estimate the quality of obtained DAGs by means of three normalized scoring metrics: *F_1 score, sensitivity and specificity*, explained in detail in [100].

In order to adapt the above-mentioned metrics to this context, we make the following assumptions for what concerns an output DAG with respect to the target DAG, given any pair of vertices (A, B):

- a *true positive* (TP) occurrence is identified as the presence of the directed edge $(A \rightarrow B)$ in both the output DAG and the target DAG or of the directed edge $(A \leftarrow B)$ in both the output DAG and the target DAG;
- a *true negative* (TN) occurrence is identified as the absence of an edge $(A \not\!\!- B)$ in both the output DAG and the target DAG;
- *retrieved instances*, i.e. the sum of true positives and false positives $(TP{+}FP)$, correspond to the set of oriented edges in the obtained DAG;
- the *set of positives* (P), i.e. the sum of true positives and false negatives $(TP + FN)$, is identified by the set of edges that are oriented in the target DAG;
- the *set of negatives* (N), i.e. the sum of true negatives and false positives $(TN + FP)$, is identified by the set of edges that are absent in the target DAG.

Provided with the definitions above, we can compute F_1 measure, sensitivity and specificity for any obtained DAG. In particular:

Sensitivity is the proportion of directed edges of the type $(A \rightarrow B)$ in the target BN that are associated to edges directed in the same orientation, i.e. $(A \rightarrow B)$, in the learnt BN;

Specificity is the proportion of absent edges in the target BN that are associated to absent edges in the learnt BN;

Precision is the proportion of directed edges of the type $(A \rightarrow B)$ in the learnt BN that are associated to edges directed in the same orientation, i.e. $(A \rightarrow B)$, in the target BN;

$\mathbf{F_1}$ score is defined as the harmonic mean of sensitivity and precision measures [100].

4.2 Standard GA Results

In this section differently-sized datasets from ASIA, ALARM and HEPAR-II networks are tested on the first presented method (Standard GA) and on competitor methods included in our benchmark, i.e. HC [82], MWST [56], K2 [63], TAN [81] and PC [7] algorithms.

Resulting graphs are reported in Figure 12, Figure 13, Figure 14 and Figure 15.

As we can observe in results graphs, our Standard GA performs generally better than its competitors; this is what we expected, since a similar trend was observed relatively to HSL-GA (a slightly different method with respect to our Standard GA) in Vafaee's experiments in [15].

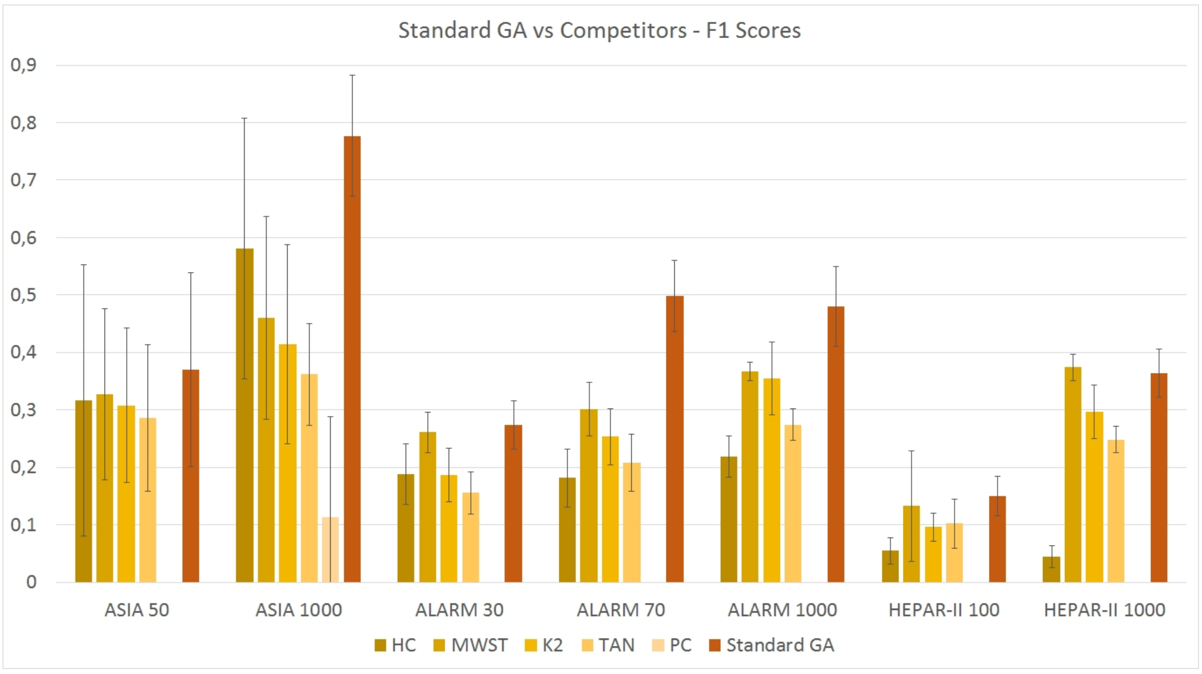


Figure 12: Comparison between standard GA and external competitors over differently-sized datasets sampled from ASIA, ALARM and HEPAR-II networks: F1 scores.

In particular HC seems to perform well with small-sized datasets, but timing issues hinder its exploration process as we increase network size: its execution ends due to time stop condition in all experiments with ALARM and HEPAR-II networks.

PC suffers even more than HC from time complexity: in medium and large problems it is forced to stop after 24 hours spent on a single run (and this explains the absence of related bars on graphs).

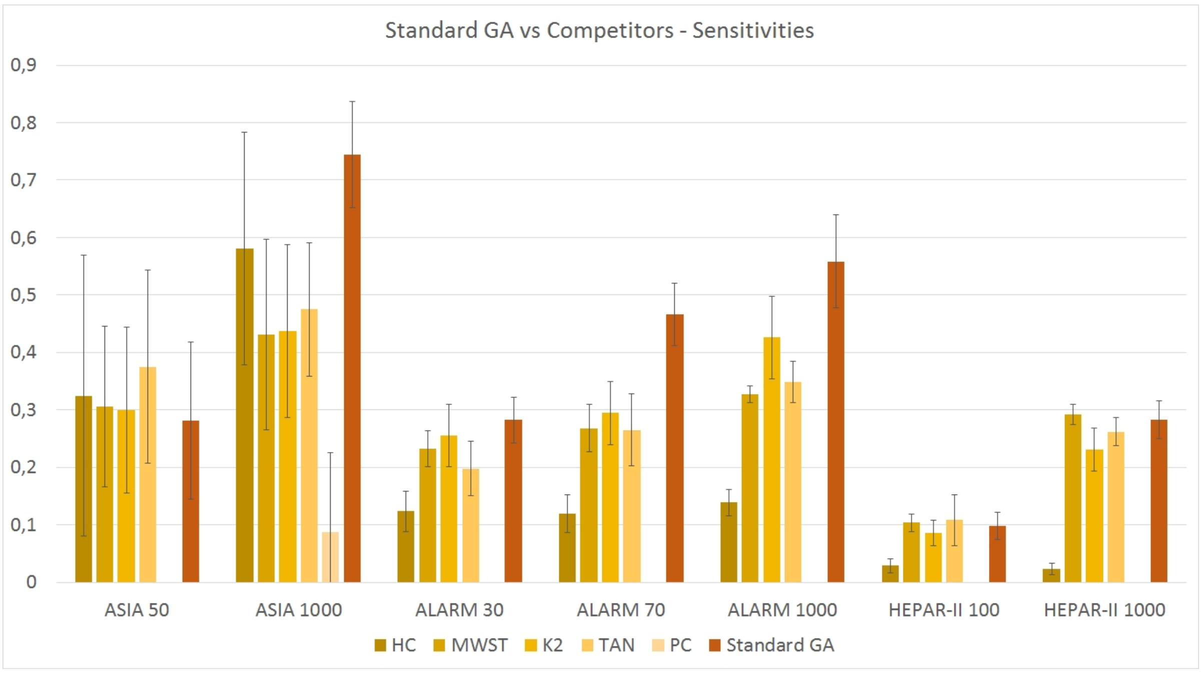


Figure 13: Comparison between standard GA and external competitors over differently-sized datasets sampled from ASIA, ALARM and HEPAR-II networks: sensitivities.

K2 and MWST are characterized by comparable specificities and Bayesian scores on the ALARM network with respect to Standard GA, but related sensitivities and F_1 scores are comparatively lower.

MWST performs similarly to Standard GA for what concerns F_1 scores and sensitivities on HEPAR-II network test cases, but performs worse in all other cases.

TAN performs always slightly worse than its founding strategy, MWST.

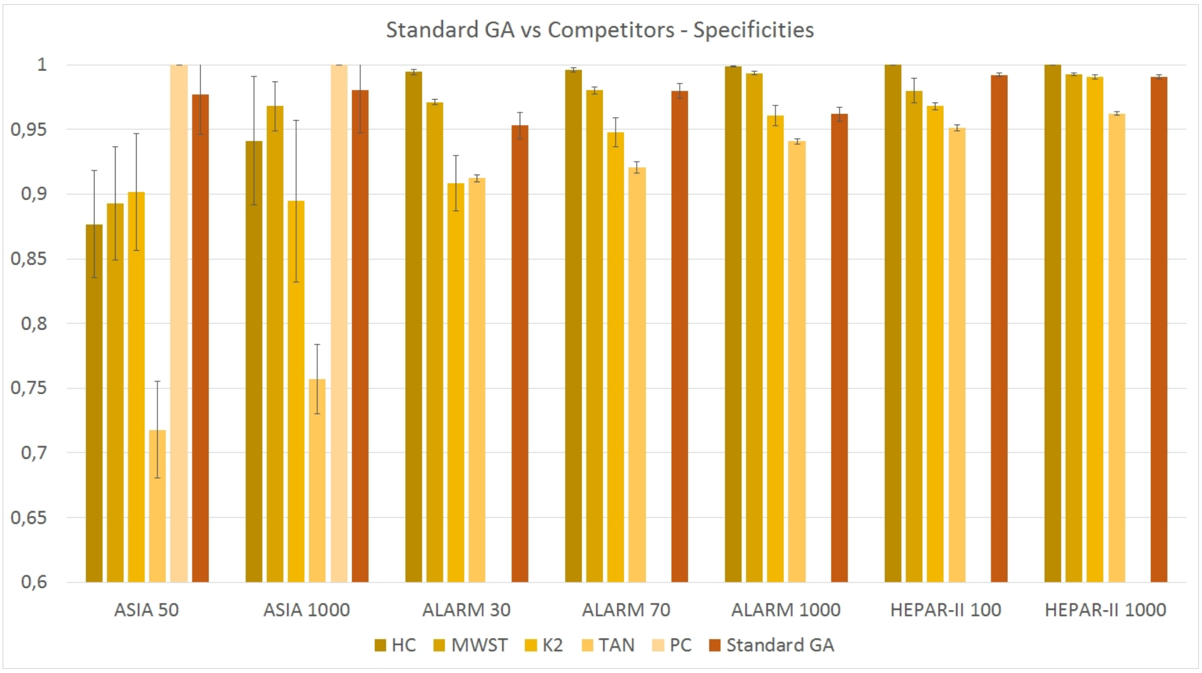


Figure 14: Comparison between standard GA and external competitors over differently-sized datasets sampled from ASIA, ALARM and HEPAR-II networks: specificities.

4.3 SiRG Results

In Figure 16, Figure 17, Figure 18 and Figure 19 are reported respectively F_1 scores, sensitivities, specificities and Bayesian scores related to the application of Simple GA and three versions of the Site-specific Rate genetic strategy on differently-sized datasets extracted from ASIA, ALARM and HEPAR II networks. In particular, first and second SiRG versions involve a standard SiRG algorithm with α respectively equal to 0.9 and 0.5, whereas the third version is the DiG-SiRG (adaptive SiRG) method.

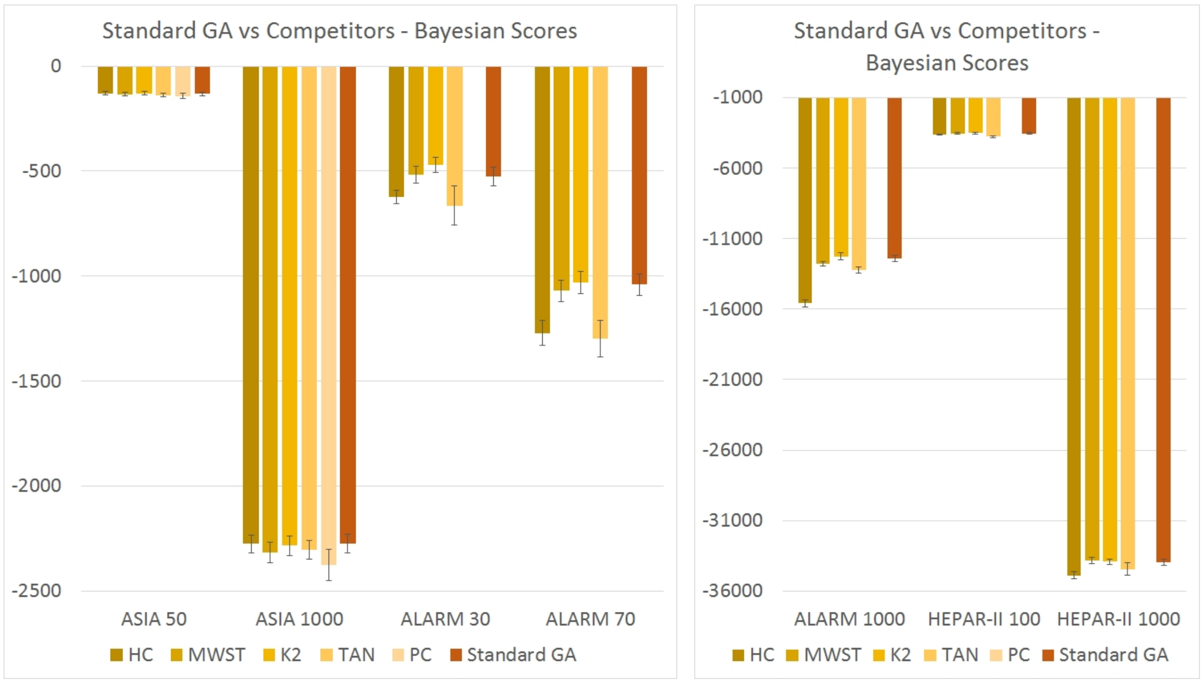


Figure 15: Comparison between standard GA and external competitors over differently-sized datasets sampled from ASIA, ALARM and HEPAR-II networks: Bayesian scores.

In particular, in all experiments: initial value for α in the DiG-SiRG procedure is set to 0.5 and the healthy diversity interval extremes are set accordingly to the time-sensitive method. Specifically, at the beginning the healthy diversity interval is set to $[\frac{1}{5}H_{max}, \frac{3}{5}H_{max}]$ then this interval is gradually reduced, generation after generation, so to get an end-of-evolution interval equal to $[\frac{1}{10}H_{max}, \frac{1}{2}H_{max}]$.

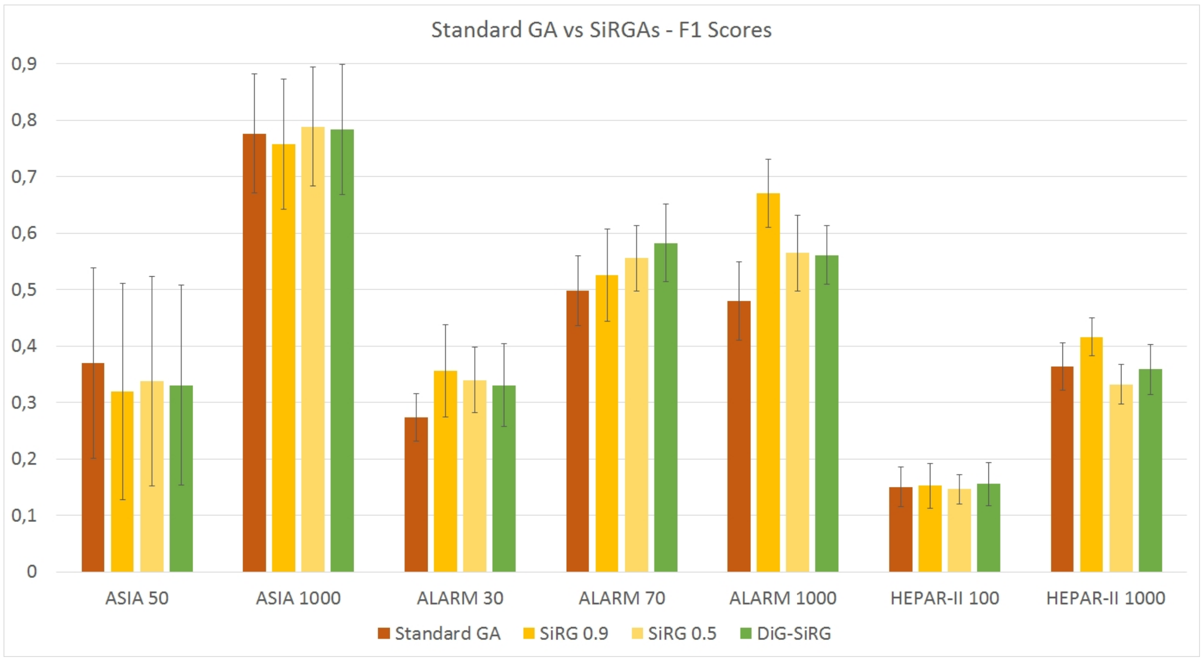


Figure 16: Standard genetic and SiRG algorithms results over differently-sized datasets sampled from ASIA, ALARM and HEPAR II networks: F1 scores.

As we can observe in Figure 16 and Figure 17 the standard SiRG method with $\alpha = 0.9$ performs better than the other basic SiRG algorithm with $\alpha = 0.5$ with ALARM-30, ALARM-1000 and HEPAR-II-1000 sample datasets, whereas the situation is reversed with the ASIA datasets and ALARM-70 dataset: as a conclusion we can say that an α value that is optimal for a particular size for the dataset, or rather for the consequent search landscape, can generally cease to be optimal with other sizes.

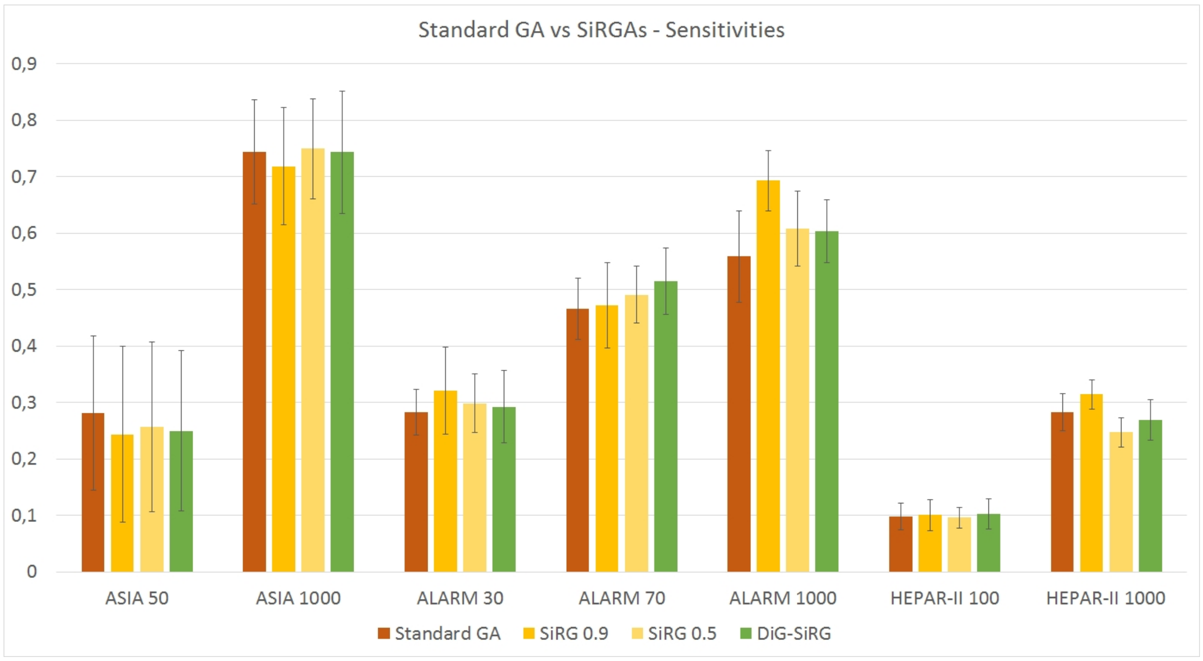


Figure 17: Standard genetic and SiRG algorithms results over differently-sized datasets sampled from ASIA, ALARM and HEPAR II networks: sensitivities.

On the other hand, the DiG-SiRG method has been implemented with a time-sensitive diversity guided strategy, which claim is that the α value should be adaptively adjusted depending on the situation.

As we can observe in Figure 16, the DiG-SiRG approach in all cases (excepting for the ALARM-1000 dataset, where the Wilcoxon test returns F_1 scores and sensitivities that are better in a statistically significant manner for the SiRGA with $\alpha = 0.9$), yields comparable or slightly better results with respect to other static-α strategies: hence we can say that theoretical expectations are confirmed by this experiment, especially when very scarce amounts of data are

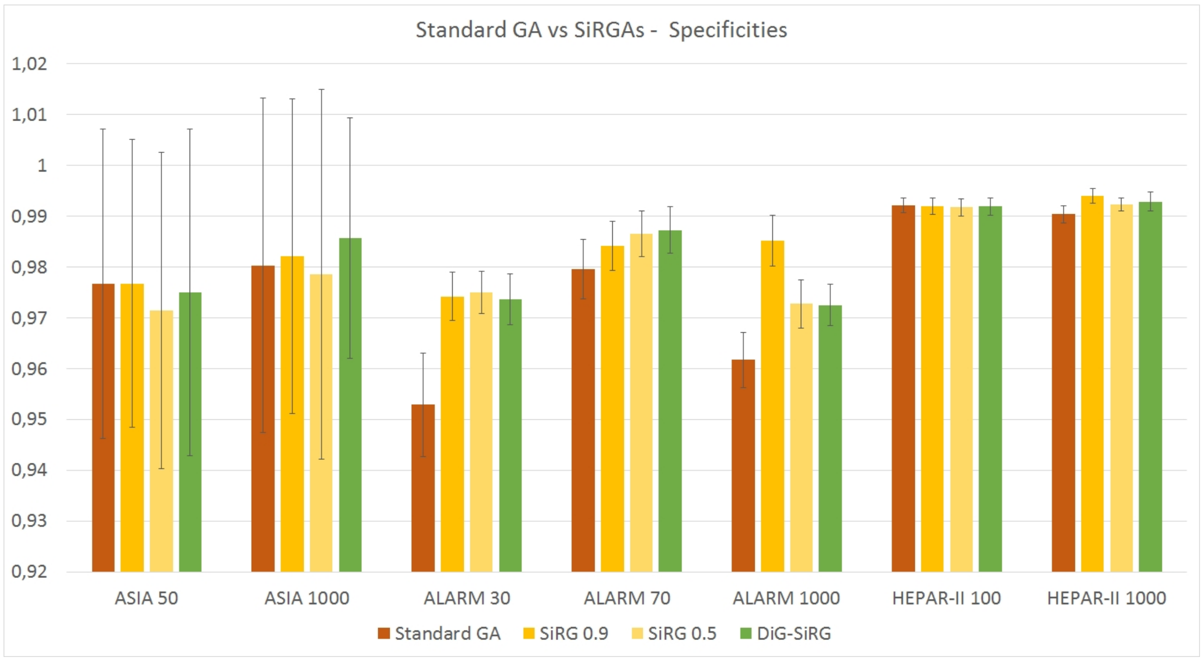


Figure 18: Standard genetic and SiRG algorithms results over differently-sized datasets sampled from ASIA, ALARM and HEPAR II networks: specificities.

available, i.e. when the diversity guidance given by Elite set information source is as deceptive as possible.

As a consequence, provided also with an improved insensitivity with respect to α parameter setting, we chose to use the DiG-SiRG method as a starting point for the development of reduced parents strategies.

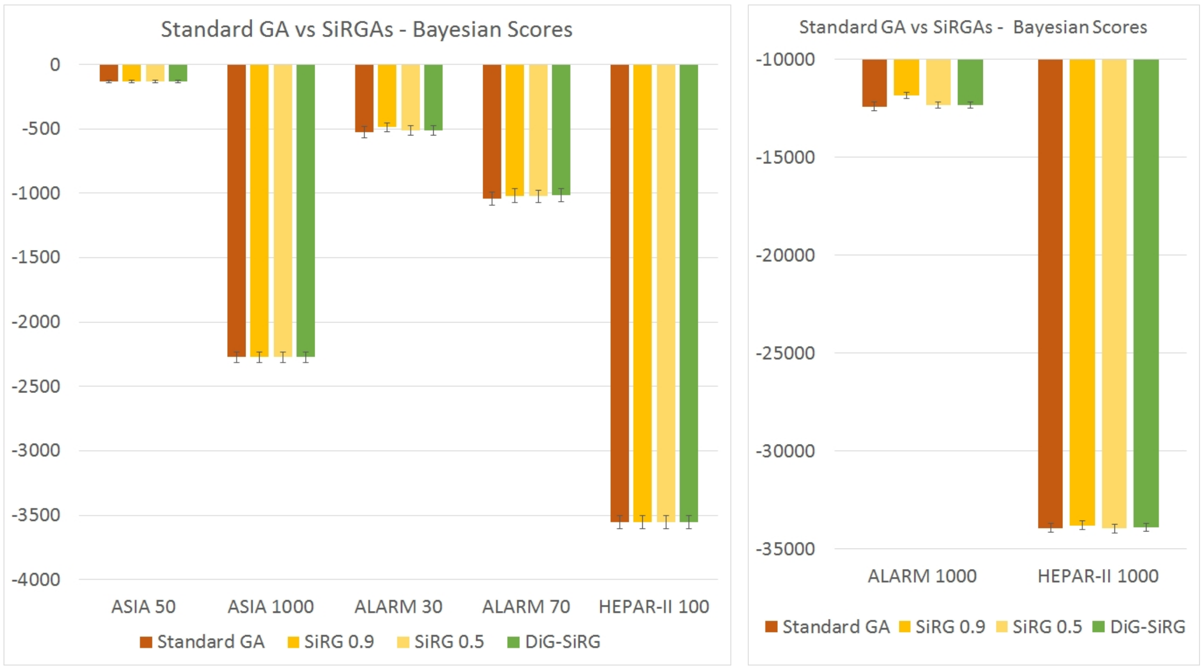


Figure 19: Standard genetic and SiRG algorithms results over differently-sized datasets sampled from ASIA, ALARM and HEPAR II networks: Bayesian scores.

4.4 Sensitivity Analysis: Amount of Edges Directed at Initialization

Another achievement of the Site-specific Rate approach, especially of its adaptive version, resides in its improved *insensitivity to initialization* with respect to a traditional GA.

As shown in Figure 20 and Figure 21, we can observe how performance of the standard GA is more sensitive to the amount of oriented edges at initialization with respect to its site-specific rate counterparts; even if the GA performs better than the SiRG methods with certain specific POI parameter values (specifically with $POI = 0$), it is more important to deal with a more

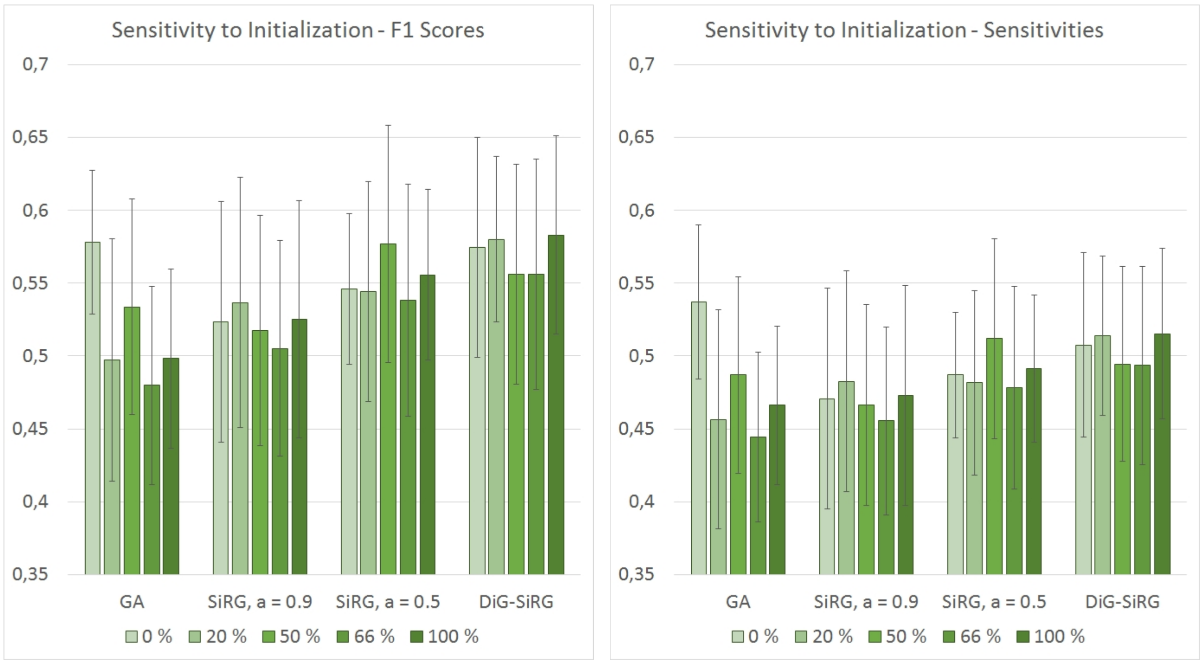


Figure 20: Standard genetic and SiRG algorithms results tested with different POI values (expressed as percentages) on the ALARM-70 sample dataset: F1 scores and sensitivities.

insensitive approach, able to return a more conservative and stable result.

By comparing SiRGAs, we can notice that performances with the DiG-SiRG method are slightly better and more robust. Indeed we focused our attention on how to improve the more insensitive method, i.e. the DiG-SiRG algorithm.

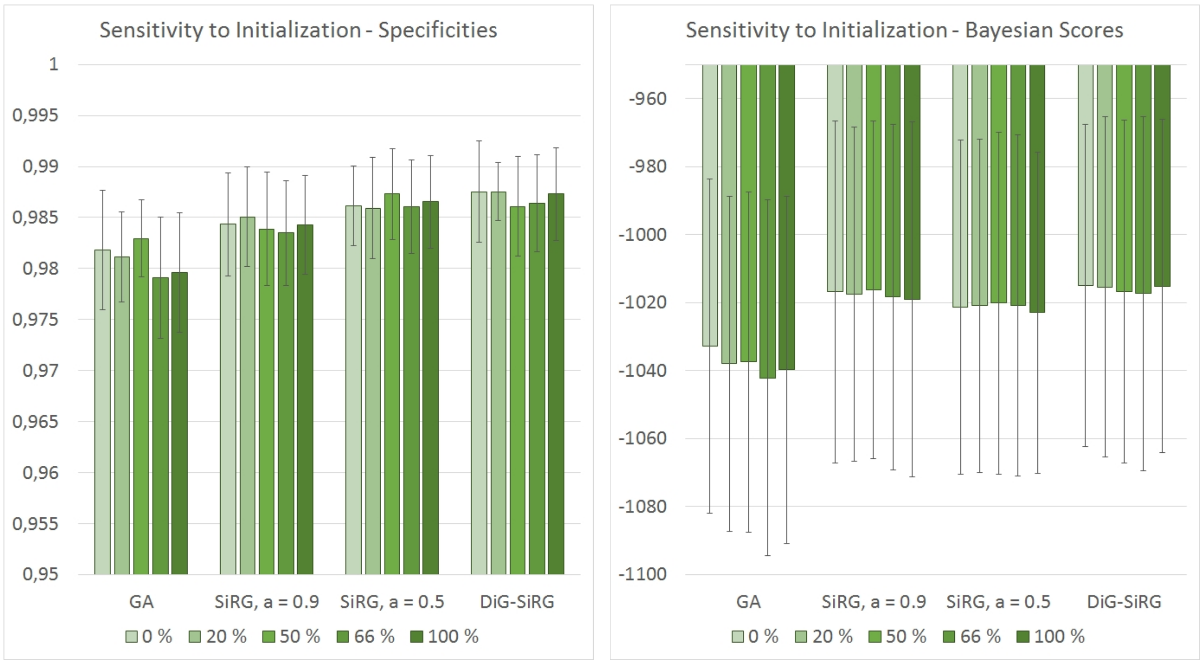


Figure 21: Standard genetic and SiRG algorithms results tested with different *POI* values (expressed as percentages) on the ALARM-70 sample dataset: specificities and Bayesian scores.

4.5 New Methods Testing

In this section a comparison among the method that experimentally resulted to be the best performing one and the novel Parents-Reduction-related methods, i.e. among DiG-SiRG, PaRe-DiG-SiRG and SPaRe-DiG-SiRG algorithms, is presented.

The objectives here are:

1. to prove the reliability and added sustain to the evolutionary process given by the data-driven parent reduction technique provided in PaRe-DiG-SiRG method, able to allow to learn large networks in a reasonable time frame and with feasible space requirements and to further enhance the learning process by dynamically restricting the search space across the evolution, in a way that addresses the phenomenon of data fragmentation.
2. to show that with the SPaRe-DiG-SiRG algorithm we can get comparable performances with respect to its basic version, i.e. the DiG-SiRG algorithm, but with an improved insensitivity to one needed parameter for use in most of BN structure learning algorithms: the maximum number of parents per node MP.

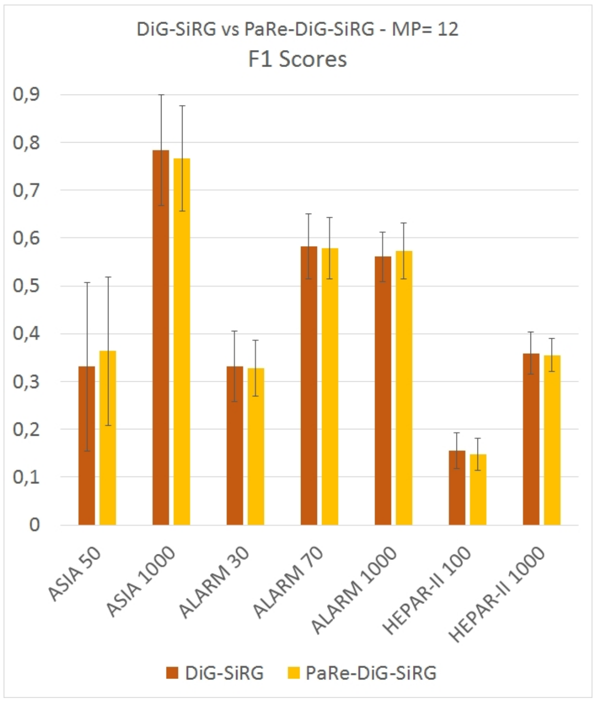
4.5.1 Evaluation with Large MP Value

This experiment was conducted to evaluate whether the MP value chosen in previous experiments ($MP = 12$) is large enough to make parents reduction action negligible with respect to final outcome performance, in a way that the evolutionary process is not altered.

As we can observe in Figure 22 and Figure 23, with such a large MP setting it results no statistically significant change in final performance related to the application of the baseline DiG-SiRG and its enhancement PaRe-DiG-SiRG, so initial hypothesis is confirmed.

4.5.2 Parents Reduction Methods Reliability Analysis

Before executing any comparison between the two presented parent reduction methods, we should first check how performance varies by applying each method at different intensities. We



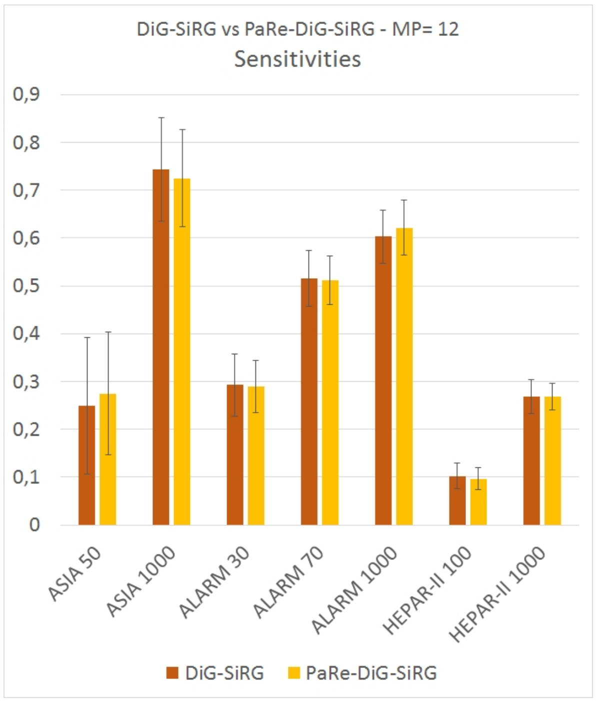
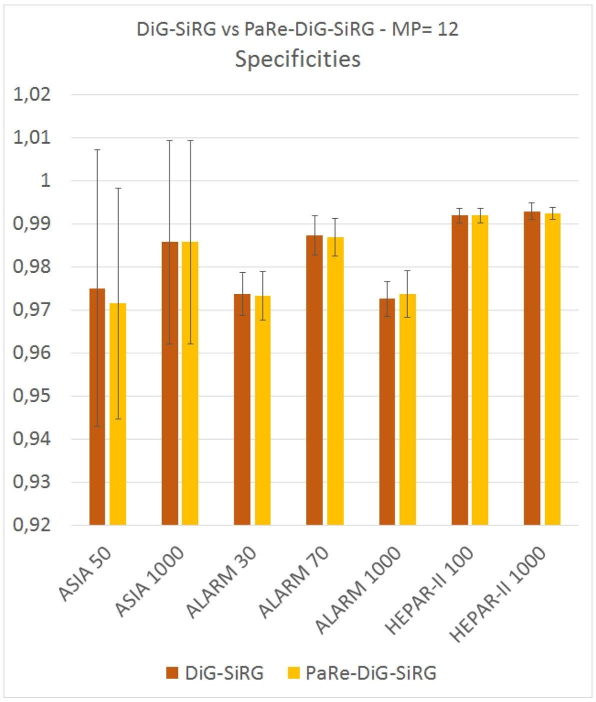


Figure 22: Comparison between standard DiG-SiRG strategy and the proposed PaRe-DiG-SiRG method with a large MP value (F1 scores and sensitivities): it results a performance variation that is not significant.

can indeed increase the effect of parent reduction by just reducing the MP threshold: if one method yields a better performance with a lower threshold value, it means that it succeeds in reducing the search space by retaining actual node's parents and by detaching the others.

In order to better compare the sensitivites to a variation in the MP parameter related to DiG-SiRG and PaRe-DiG-SiRG methods, we propose in Figure 24 and Figure 25 a view in



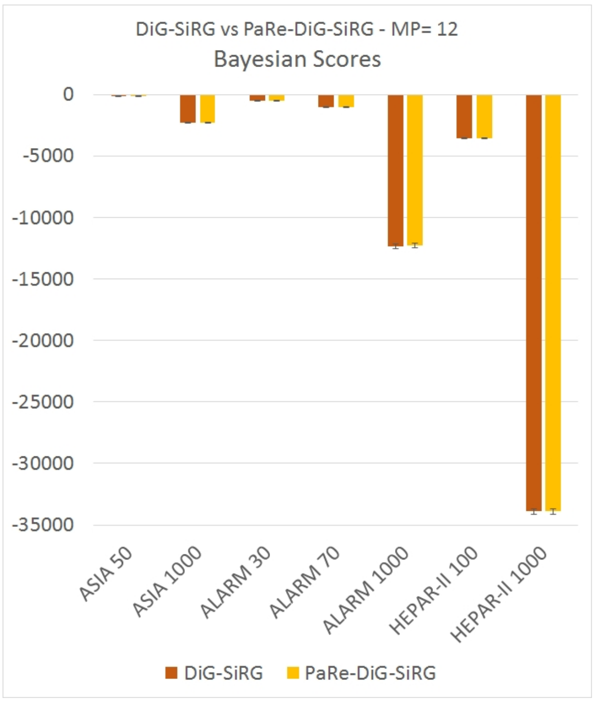


Figure 23: Comparison between standard DiG-SiRG strategy and the proposed PaRe-DiG-SiRG method with a large MP value (specificities and Bayesian scores): it results a performance variation that is not significant.

parallel of both methods performances for two different values of MP (minimum possible for the network and 12) on test cases related to two benchmark networks (ALARM and HEPAR II) with various dataset sizes. The minimum possible MP setting for each network is basically its maximum indegree: we indeed set the more effective value for MP as equal to 2 for ASIA, 4 for ALARM, 3 for INSURANCE and 6 for HEPAR II.

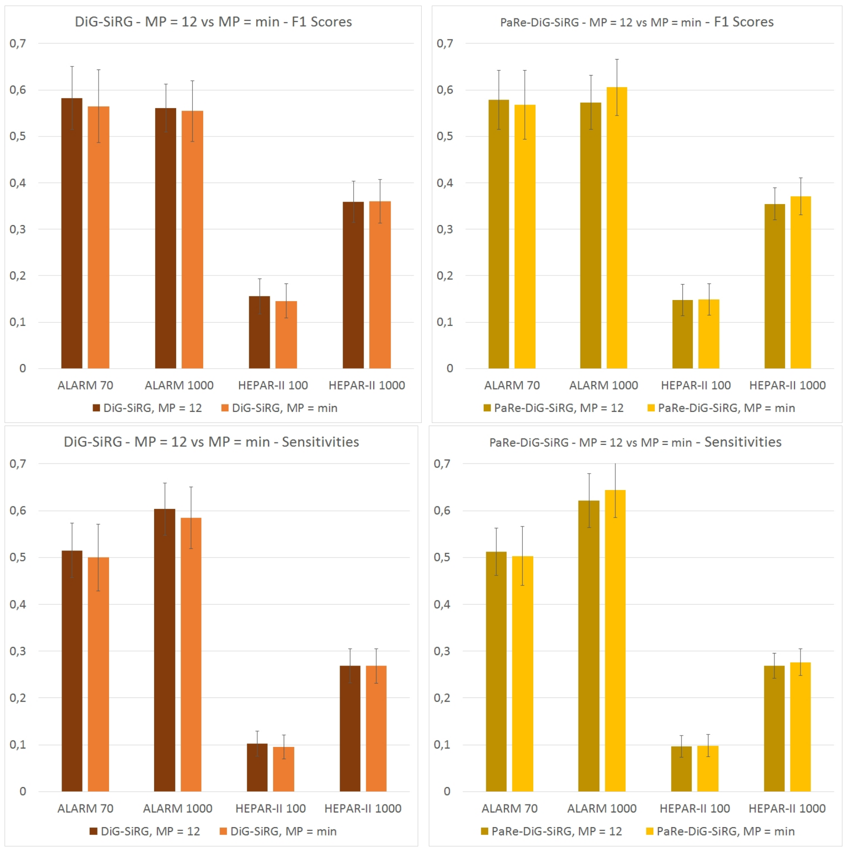


Figure 24: MP sensitivity analysis between standard DiG-SiRG strategy and the proposed PaRe-DiG-SiRG method (F1 scores and sensitivities): two different values for MP are applied to each method on four test cases with ALARM and HEPAR-II networks.

As we can observe in Figure 24 and Figure 25, for what concerns the standard DiG-SiRG method, the MP variation experiments on large ALARM-1000 and HEPAR-II-1000 test cases seem to not influence performance in any case, but if we take into account the experiments driven on the remaining small-sized datasets, we can see a slight drop in the performance

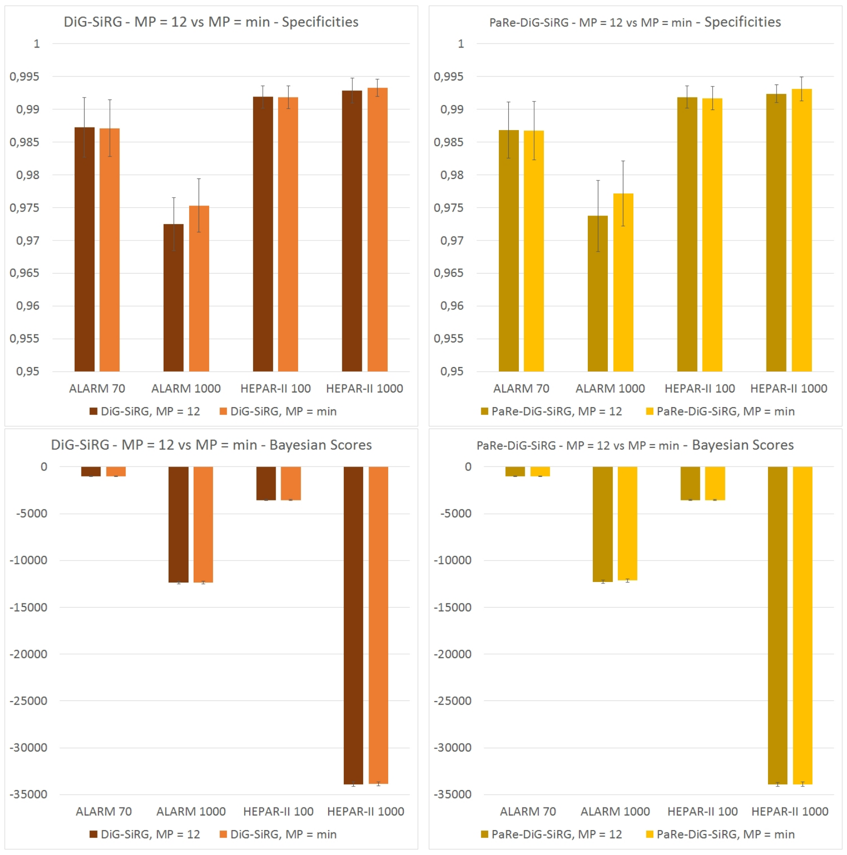


Figure 25: MP sensitivity analysis between standard DiG-SiRG strategy and the proposed PaRe-DiG-SiRG method (specificities and Bayesian scores): two different values for MP are applied to each method on four test cases with ALARM and HEPAR-II networks.

whenever we try to intensify the standard random picking MP reduction technique. This could be explained by the fact that the basic MP reduction technique takes a consistent amount of wrong choices for what concerns parents selection, driving the search towards cases in which nodes have the wrong parents.

By observing again Figure 24 and Figure 25, we can ascertain a different trend on PaRe-DiG-SiRG method results. Indeed ALARM-70 and HEPAR-II-100 performances still remain comparable for both tested MP values, but on the other hand we can observe a slight improvement in large dataset test cases, especially on the 1000-samples ALARM test case.

As suggested by above-mentioned results, we can say that the Elite-guided parent reduction technique does not perform worse than standard random picking method in correctly retaining actual nodes parents, but unfortunately, in the environment defined by used parameters, it does not result any statistically significant improvement on any metric, excepting for the Bayesian score on the ALARM-1000 test case: here it results that a stronger parents reduction action operated by the new method significantly improves the Bayesian score.

We can also observe that PaRe-DiG-SiRG performance always improves by increasing data sample size, because more information is available for data-driven parents reduction.

4.5.3 Parents Reduction Methods Performance Comparison

In this section it is provided a performance comparison between the two presented parents reduction techniques, i.e. the random picking baseline and the elite-guided parents reduction method.

In order to make this comparison operational, we set MP to the minimum allowable value relatively to each test case, i.e. a value equal to the maximum fan-in in the network: $MP = 3$ for INSURANCE, $MP = 4$ for ALARM and $MP = 6$ for HEPAR-II.

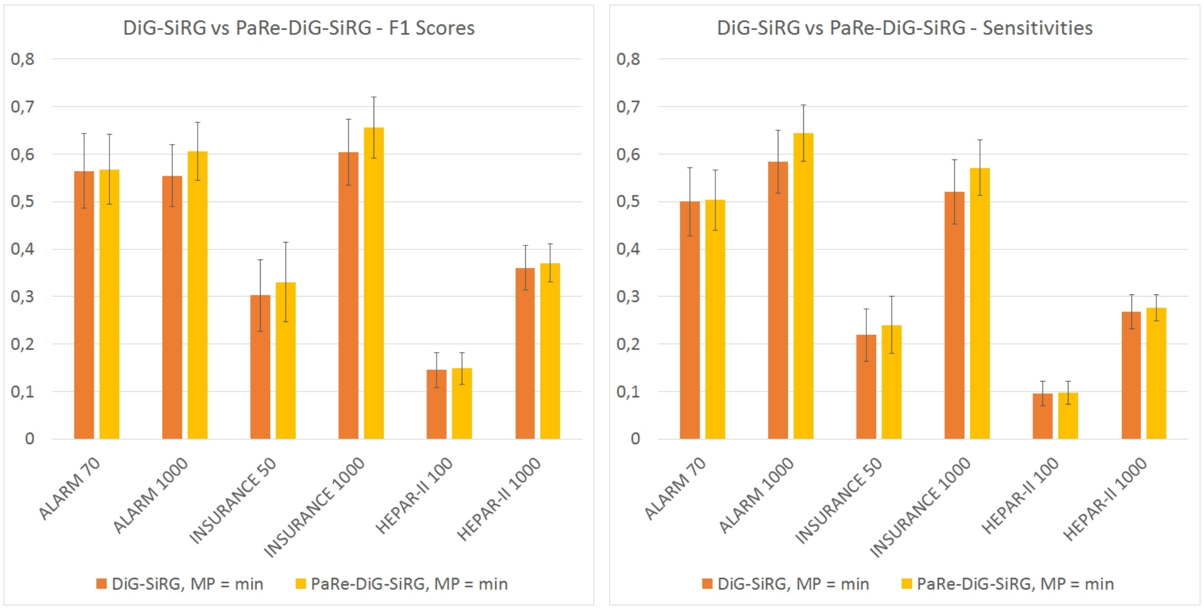


Figure 26: Performance comparison between DiG-SiRG strategy and the proposed PaRe-DiG-SiRG method (F1 scores and sensitivities): the minimum value for MP is applied to each method on six test cases with ALARM, INSURANCE and HEPAR-II networks.

As can be seen in Figure 26 and Figure 27, when we intensify the parents reduction action it results that the novel PaRe-DiG-SiRG method yields comparable or better results than the baseline. In particular, it results a statistically significant improvement relatively to F_1, sensitivity and Bayesian metrics on test cases with large datasets, i.e. ALARM-1000 and INSURANCE-1000, but not on smaller-sized ones: the more significant improvement on large datasets can be explained by the higher availability of information, useful to steer the elite-guided parents reduction mechanism.

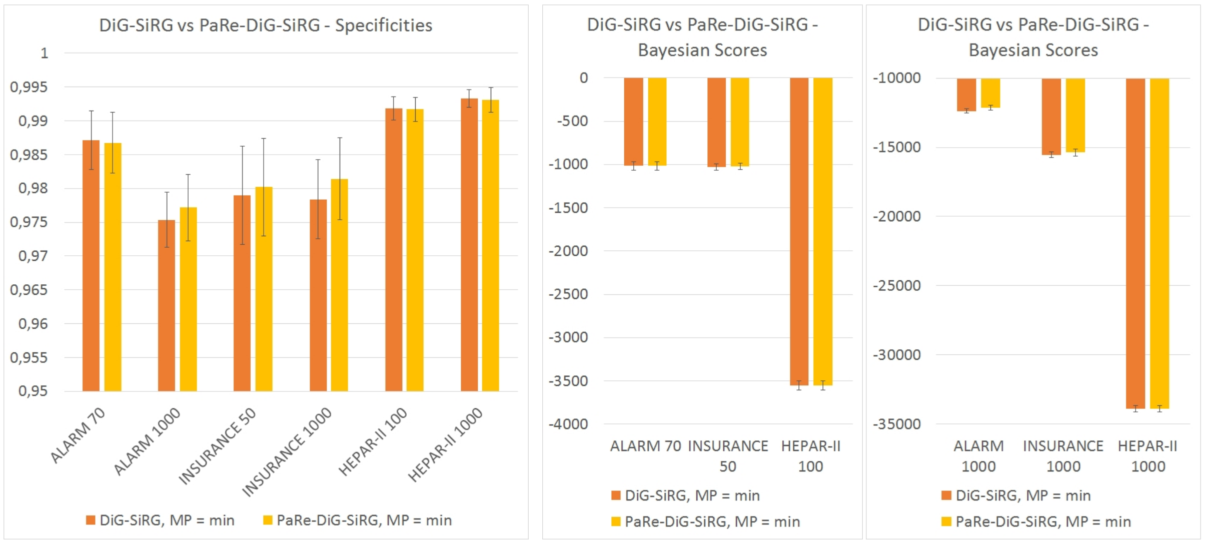


Figure 27: Performance comparison between DiG-SiRG strategy and the proposed PaRe-DiG-SiRG method (specificities and Bayesian scores): the minimum value for MP is applied to each method on six test cases with ALARM, INSURANCE and HEPAR-II networks.

4.5.4 Adaptive Parents Reduction Statistics Evaluation

In Figure 28 are reported some statistics concerning the per-node MP threshold reduction process related to tests with SPaRe-DiG-SiRG on 70 and 1000 sample sizes extracted from the ALARM network.

On the x axis it is reported the generation number, whereas on the y axis it is reported the number of nodes related to each of the following statistics:

Max indicates the maximum number of parents over all nodes at the current generation;

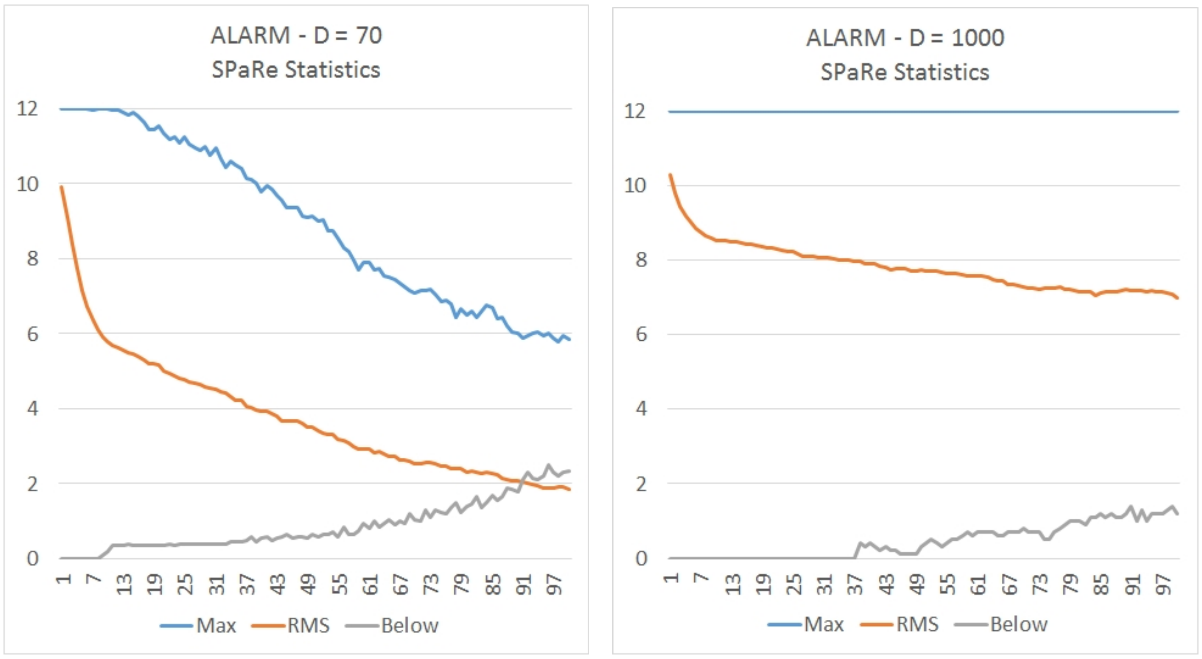


Figure 28: Statistics describing the dynamic and adaptive MP threshold reduction process throughout a 100 generations evolution, extracted from tests on the ALARM network driven with differently-sized datasets; in particular Max indicates the maximum number of parents over all nodes at the current generation, RMS is the Root Mean Square between the vector containing the actual number of parents of the target DAG and the MP vector at the given generation, Below reports the sum over the difference vector NP-MP, but by considering only those MP values that are lower than the actual number of parents in the target DAG, i.e. NP, at the given generation.

RMS is the Root Mean Square between two vectors, one being the actual number of parents of the target DAG and the other one being the **MP** vector at the given generation;

Below reports the sum over all differences $NP(i) - MP(i)$ related to those $MP(i)$ values that are lower than the actual number of parents in the target DAG $NP(i)$ at the given generation.

The optimal trend for these statistics would be a RMS value rapidly converging to 0, a Max value rapidly converging to the overall maximum number of parents in the target DAG and a Below value at a constant 0.

We can observe in Figure 28 that the Max statistic, in the ALARM-1000 test case, is kept at the constant maximum allowable value (12): this means that at least one node in the network at every generation has a numerous set of parents across the whole elite set, i.e. many nodes in the network are considered to be good parent candidates for that specific node.

The candidate parents set results to be more various with a larger dataset: we presume that the greater information availability encourages and intensifies possible (even fortuitous) dependences between a node and its potential parents.

By looking again at the graphs, specifically by focusing on the Below statistic, it is possible to ascertain another significant property of the elite-driven MP threshold update process: with more data availability the amount of "wrong choices" relatively to MP threshold setting is decreased: indeed with a dataset size of 70 the Below statistic surpasses the value 2 on average, whereas with a greater dataset of 1000 samples the Below trend seems to stabilize at around 1.5 on average. Consequently, we can conclude that the information extracted from the elite set and thus from data is correctly employed to guide the MP threshold update process.

4.5.5 Final Results

In this section SPaRe-DiG-SiRG performance is evaluated and compared to DiG-SiRG and PaRe-DiG-SiRG performances with two different parameter sets: in particular we set the first experiment with previously used values for population size N and number of generations M (respectively $N = 100$ and $M = 100$), whereas in the second experiment we set N and M respectively to 50 and 200.

Moreover, DiG-SiRG and PaRe-DiG-SiRG are evaluated with the minimum allowable value for MP, as in previous experiment; on the other hand, since it only constitutes the maximum allowable value in the dynamic adaptation process, MP is set to 12 (sufficiently large value) for the SPaRe-DiG-SiRG algorithm.

As we can observe in Figure 29 and Figure 30, the adaptive method for parents reduction, SPaRe-DiG-SiRG, in no case performs worse than the baseline DiG-SiRG, thus it yields comparable results with the advantage that it is not needed to choose a suitable value for MP since it is automatically found by the algorithm, per each node.

As it already happened with PaRe-DiG-SiRG, also the self parent reducing GA yielded a statistically significant improvement on the INSURANCE-1000 test case with respect to F_1 score, sensitivity and specificity metrics.

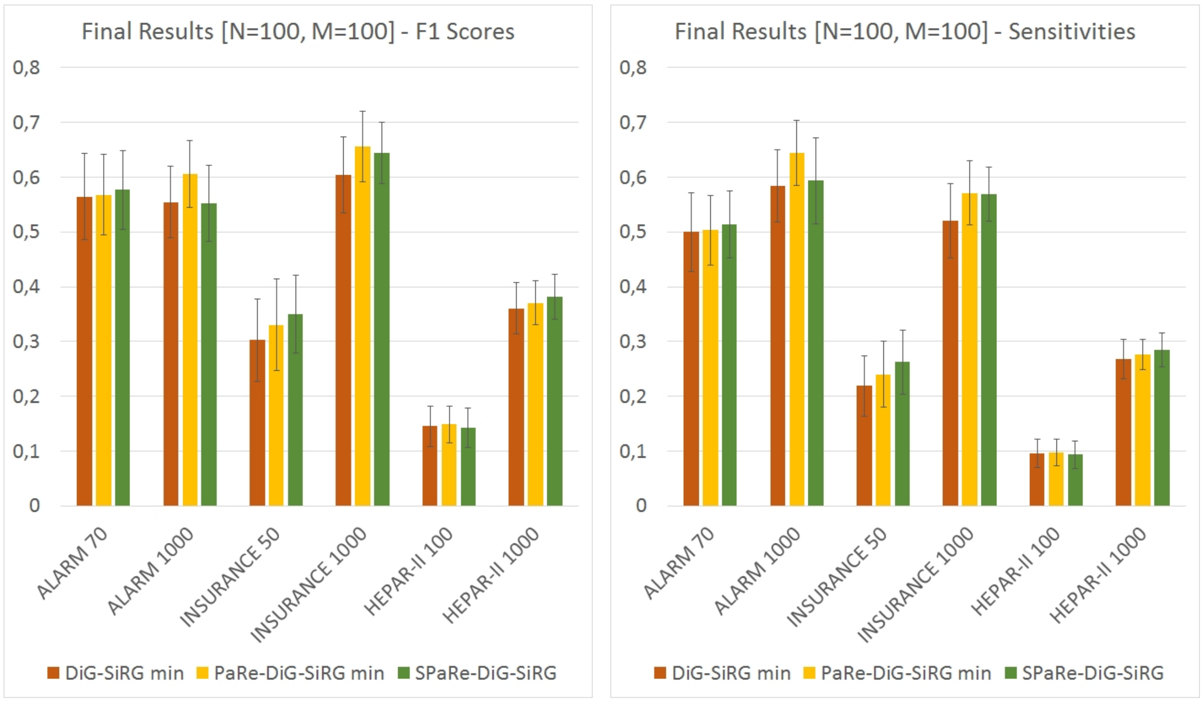


Figure 29: Performance comparison between DiG-SiRG strategy and the proposed PaRe-DiG-SiRG and SPaRe-DiG-SiRG methods (F1 scores and sensitivities): experiments involved six test cases related to ALARM, INSURANCE and HEPAR-II networks.

Moreover, SPaRe-DiG-SiRG yields a statistically significant improvement on the INSURANCE-50 test case with respect to the baseline (DiG-SiRG) for what concerns sensitivity and specificity metrics: the relevance of this result is empathized by the fact that, on the other hand, in this case PaRe-DiG-SiRG does not yield a statistically significant improvement with respect to the baseline.

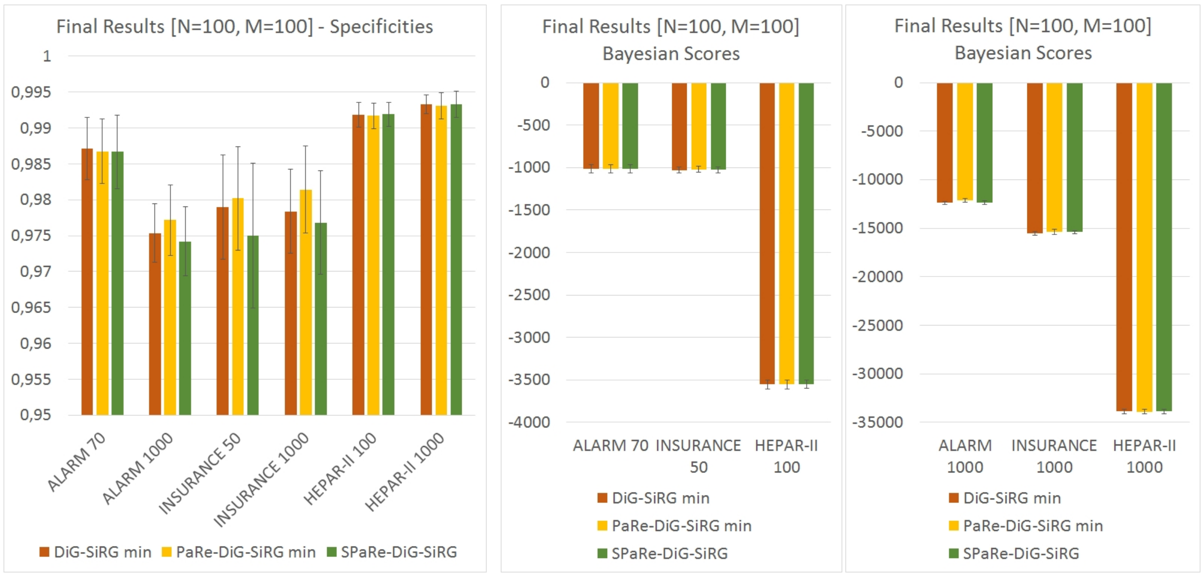
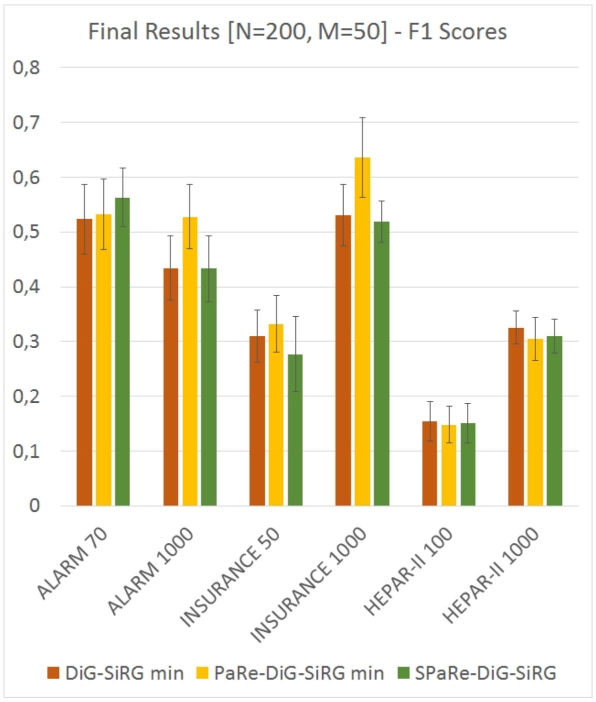


Figure 30: Performance comparison between DiG-SiRG strategy and the proposed PaRe-DiG-SiRG and SPaRe-DiG-SiRG methods (specificities and Bayesian scores): experiments involved six test cases related to ALARM, INSURANCE and HEPAR-II networks.

By varying the parameters setting so to have $N = 200$ and $M = 50$, we obtain the graph bars reported in Figure 31 and Figure 32.

As can be observed in Figure 31 and Figure 32, PaRe-DiG-SiRG keeps being the best performing method with ALARM-1000 and INSURANCE-1000 test cases: this can be considered as a confirmation that the knowledge-driven parents reduction outperforms the baseline when the algorithm is provided with enough data in more than one among the possible parameter settings.



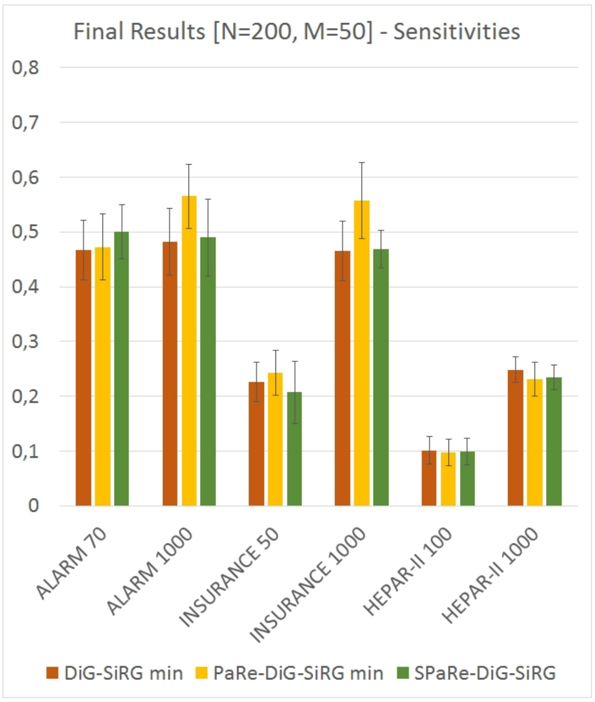


Figure 31: Performance comparison between DiG-SiRG strategy and the proposed PaRe-DiG-SiRG and SPaRe-DiG-SiRG methods (F1 scores and sensitivities): experiments involved six test cases related to ALARM, INSURANCE and HEPAR-II networks with a different setting for population size and number of generations parameters.

With this particular parameter set we can also observe that SPaRe-DiG-SiRG outperforms the baseline (DiG-SiRG) on the ALARM-70 test case with respect to F_1 score and sensitivity metrics. In all other cases, the three presented algorithms yield significantly similar results.

All results presented in this section are reported in terms of F_1 scores, sensitivities, specificities and Bayesian scores in Table X and Table XI. In particular, the result metrics indicated

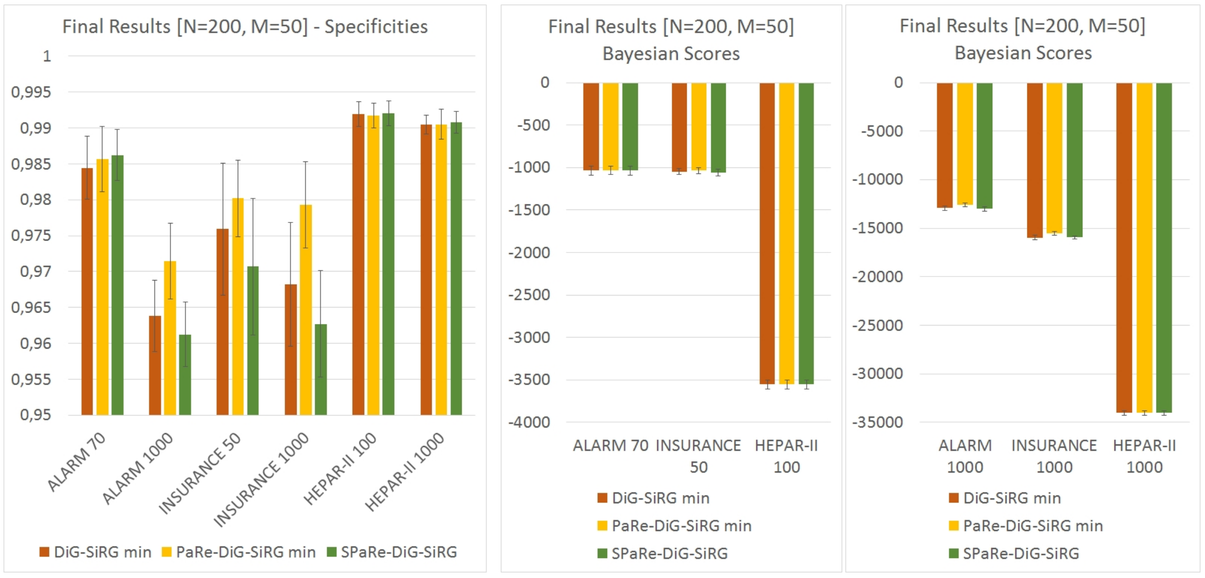


Figure 32: Performance comparison between DiG-SiRG strategy and the proposed PaRe-DiG-SiRG and SPaRe-DiG-SiRG methods (specificities and Bayesian scores): experiments involved six test cases related to ALARM, INSURANCE and HEPAR-II networks with a different setting for population size and number of generations parameters.

in bold are those with the highest values, as well as values whose difference with the highest one are statistically insignificant.

In conclusion, we can say that overall it is preferable to use the PaRe-DiG-SiRG method, especially in a data scarcity situation: the unique issue in this case is the need to choose an acceptable and efficient value for MP.

On the other hand, if we don't have available the knowledge about an acceptable MP value, then we can still make use of the SPaRe-DiG-SiRG method.

TABLE X: FINAL RESULTS ON ALARM, INSURANCE AND HEPAR-II NETWORKS ASSUMING N = 100 AND M = 100.

Method	**ALARM**							
	70				1000			
	F_1	Sens	Spec	Score	F_1	Sens	Spec	Score
DiG-SiRG	**0.565** (0.078)	**0.5** (0.071)	**0.987** (0.004)	**-1016.4** (48.7)	0.555 (0.065)	0.585 (0.066)	**0.975** (0.004)	-12372.0 (160.15)
PaRe-DiG-SiRG	**0.568** (0.074)	**0.503** (0.063)	**0.987** (0.004)	**-1015.1** (50.40)	**0.606** (0.061)	**0.645** (0.06)	**0.977** (0.005)	**-12143.0** (178.48)
SPaRe-DiG-SiRG	**0.577** (0.072)	**0.514** (0.061)	**0.987** (0.005)	**-1016.0** (48.25)	0.553 (0.07)	0.593 (0.078)	**0.974** (0.004)	-12354.0 (206.59)

Method	**INSURANCE**							
	50				1000			
	F_1	Sens	Spec	Score	F_1	Sens	Spec	Score
DiG-SiRG	**0.303** (0.075)	0.219 (0.055)	**0.979** (0.007)	**-1028.4** (37.04)	0.604 (0.069)	0.520 (0.068)	**0.978** (0.006)	-15545.0 (210.34)
PaRe-DiG-SiRG	**0.33** (0.084)	**0.24** (0.06)	**0.98** (0.007)	**-1023.0** (35.73)	**0.656** (0.065)	**0.571** (0.058)	**0.981** (0.006)	**-15377.0** (236.27)
SPaRe-DiG-SiRG	**0.35** (0.071)	**0.263** (0.058)	**0.975** (0.01)	**-1027.1** (34.04)	**0.644** (0.056)	**0.569** (0.049)	**0.977** (0.007)	**-15409.0** (181.57)

Method	**HEPAR II**							
	100				1000			
	F_1	Sens	Spec	Score	F_1	Sens	Spec	Score
DiG-SiRG	**0.145** (0.037)	**0.096** (0.026)	**0.992** (0.002)	**-3552.6** (51.63)	**0.36** (0.047)	**0.268** (0.037)	**0.993** (0.001)	**-33883.0** (209.69)
PaRe-DiG-SiRG	**0.149** (0.034)	**0.098** (0.024)	**0.992** (0.002)	**-3552.8** (51.88)	**0.371** (0.04)	**0.276** (0.028)	**0.993** (0.002)	**-33899.0** (230.45)
SPaRe-DiG-SiRG	**0.143** (0.035)	**0.093** (0.025)	**0.99** (0.002)	**-3552.2** (51.26)	**0.381** (0.041)	**0.284** (0.031)	**0.993** (0.002)	**-33886.0** (225.13)

TABLE XI: FINAL RESULTS ON ALARM, INSURANCE AND HEPAR-II NETWORKS ASSUMING N = 200 AND M = 50.

Method	**ALARM**							
	70				1000			
	F_1	Sens	Spec	Score	F_1	Sens	Spec	Score
DiG-SiRG	0.523 (0.063)	0.467 (0.054)	**0.984** (0.004)	**-1036.2** (52.14)	0.434 (0.058)	0.482 (0.061)	**0.964** (0.005)	-12936.0 (247.08)
PaRe-DiG-SiRG	**0.532** (0.064)	**0.473** (0.06)	**0.986** (0.005)	**-1030.4** (50.76)	**0.528** (0.059)	**0.565** (0.059)	**0.971** (0.005)	**-12603.0** (172.27)
SPaRe-DiG-SiRG	**0.563** (0.072)	**0.5** (0.061)	**0.986** (0.005)	**-1036.5** (48.25)	0.433 (0.07)	0.49 (0.078)	**0.961** (0.004)	-13002.0 (206.59)

Method	**INSURANCE**							
	50				1000			
	F_1	Sens	Spec	Score	F_1	Sens	Spec	Score
DiG-SiRG	**0.31** (0.048)	0.226 (0.036)	0.976 (0.009)	-1046.9 (34.8)	0.531 (0.056)	0.465 (0.054)	0.968 (0.009)	-15969.0 (235.55)
PaRe-DiG-SiRG	**0.332** (0.052)	**0.243** (0.041)	**0.98** (0.005)	**-1036.0** (34.0)	**0.636** (0.073)	**0.558** (0.069)	**0.979** (0.006)	**-15537.0** (199.66)
SPaRe-DiG-SiRG	0.277 (0.068)	0.208 (0.057)	0.971 (0.01)	-1059.8 (36.38)	0.52 (0.038)	0.468 (0.035)	0.963 (0.007)	-15931.0 (164.64)

Method	**HEPAR II**							
	100				1000			
	F_1	Sens	Spec	Score	F_1	Sens	Spec	Score
DiG-SiRG	**0.154** (0.036)	**0.101** (0.025)	**0.992** (0.002)	**-3552.7** (51.63)	**0.326** (0.031)	**0.248** (0.023)	**0.991** (0.001)	**-34034.0** (225.4)
PaRe-DiG-SiRG	**0.148** (0.034)	**0.098** (0.024)	**0.992** (0.002)	**-3552.9** (51.25)	**0.305** (0.04)	**0.231** (0.031)	**0.991** (0.002)	**-34032.0** (225.05)
SPaRe-DiG-SiRG	**0.152** (0.036)	**0.099** (0.025)	**0.992** (0.002)	**-3552.6** (51.4)	**0.31** (0.031)	**0.234** (0.023)	**0.991** (0.002)	**-34034.0** (227.94)

CHAPTER 7

SUMMARY AND CONCLUSION

Motivated by the fact that Bayesian networks constitute a powerful framework for probabilistic reasoning and expert elicitation and have been extensively used in a variety of research domains, this thesis work offers and analyzes new methods for unsupervised Bayesian network structure learning.

In this work it is provided a method able to learn the structure of the Bayesian network underlying a set of data samples, by focusing on problems with a limited amount of available data. In particular, the main contribution of this thesis is a Hybrid learning algorithm able first to reliably reduce the search space and then to exhaustively explore it, by taking advantage of data-informed expedients as well. The proposed approach involves a parameterized Genetic Algorithm in order to pursue the task: this metaheuristic has been chosen because of its efficient global search capabilities even across a very large search space and because of its flexibility and adaptability; on the other hand it consistently suffers from time and space complexity relatively to other search methods in the literature, and moreover its performance is heavily influenced by the choice of a large set of parameters.

The research covered in this work is concerned with designing a series of hybrid methods on a build-up basis: they were provided with enhancements already existing in the literature but not yet applied to the Bayesian network structure learning topic and also with novel improvements able to further restrict the search space during the evolutionary process, in a data-driven manner. In the experimental chapter of this thesis it is possible to ascertain how presented algorithms allow to better address time and space complexity, sensitivity to parameters setting issues as well as the problem of data fragmentation, with the advantage of higher performances in some cases.

CHAPTER 8

FUTURE WORK

Future work of this research is twofold: it is focused on enhancing the novel methods presented in Sections 5.5 and 5.6, as well as identifying and formalizing problem instances and conditions in which their peculiarities are fully operative and efficacious.

In this thesis work we have successfully applied genetic rate adaptation enhancements already existing in the literature to our task and then we have concentrated our efforts on a smart way to perform parent reduction: the number of opportunities for further improvement still waiting to be undertaken is large as the quantity of genetic subroutines and parameters that have still to be considered for optimization. Hence, our future efforts may be directed towards refinement of elite selection or initialization procedures, as well as an adaptive approach to the population size parameter, among the others.

Furthermore, it would be interesting to analyze which Bayesian network structure learning problem instances are most suitable to be tackled by means of the proposed methods, so to identify related real-life contexts to address.

For instance, in the first place we will investigate about the performance of our method when applied to learn very large or massive networks (respectively with a network size comprised between 100 and 1000 nodes and larger than 1000 nodes, accordingly to Bayesian Network

Repository [116] categorization). After that we ascertain the most suitable applications, we will apply our algorithms to real data and conduct all pertinent analysis, preferably by providing a concrete contribution to some specific on-going research topic.

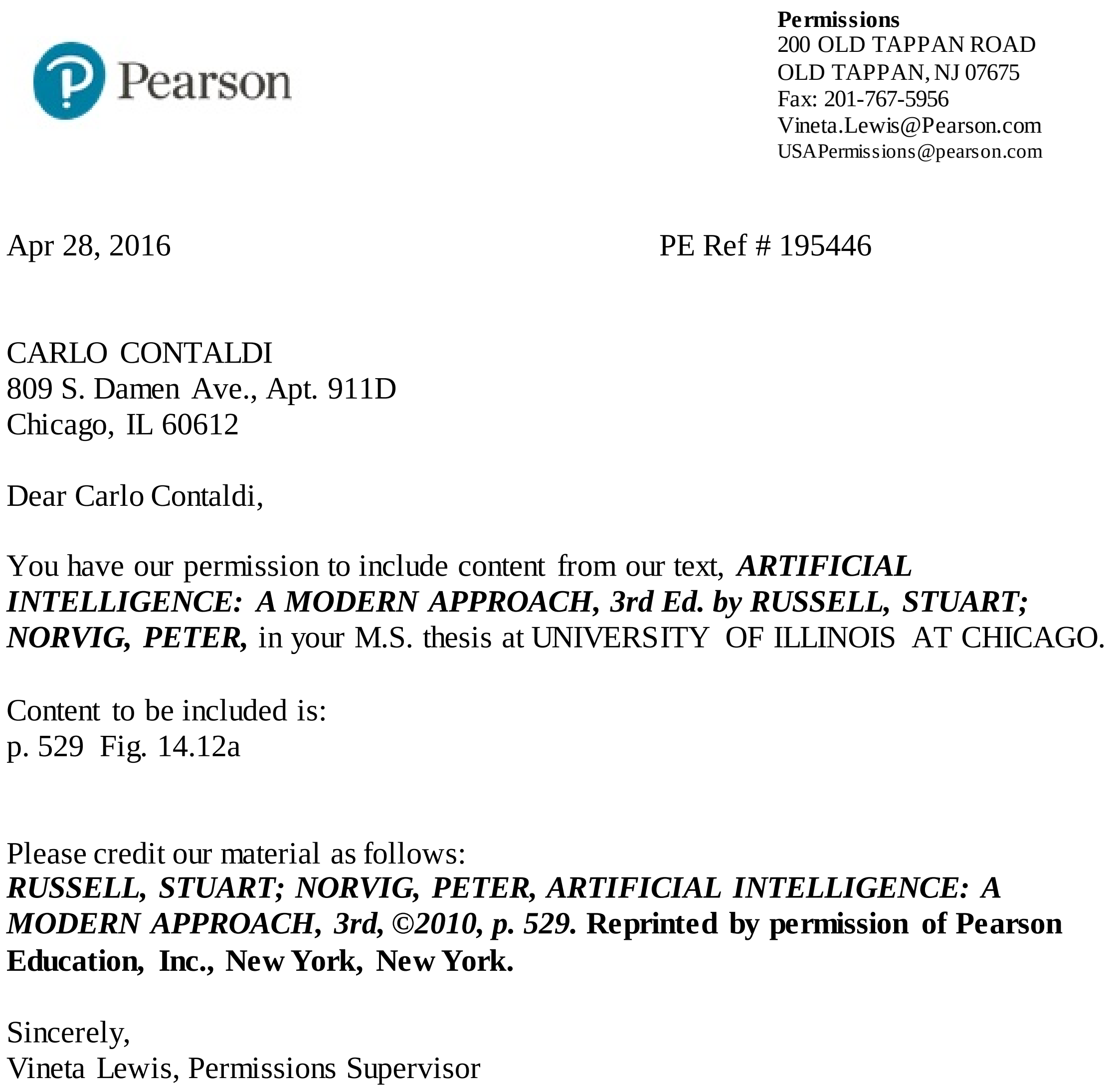
APPENDIX

PERMISSION LETTERS TO REPRINT COPYRIGHT MATERIAL

1 Pearson Education

In this section it is reported the permission letter I received from Pearson Education (USAPermissions@pearson.com) related to Figure 1.

APPENDIX (continued)



Permissions
200 OLD TAPPAN ROAD
OLD TAPPAN, NJ 07675
Fax: 201-767-5956
Vineta.Lewis@Pearson.com
USAPermissions@pearson.com

Apr 28, 2016 PE Ref # 195446

CARLO CONTALDI
809 S. Damen Ave., Apt. 911D
Chicago, IL 60612

Dear Carlo Contaldi,

You have our permission to include content from our text, ***ARTIFICIAL INTELLIGENCE: A MODERN APPROACH, 3rd Ed. by RUSSELL, STUART; NORVIG, PETER,*** in your M.S. thesis at UNIVERSITY OF ILLINOIS AT CHICAGO.

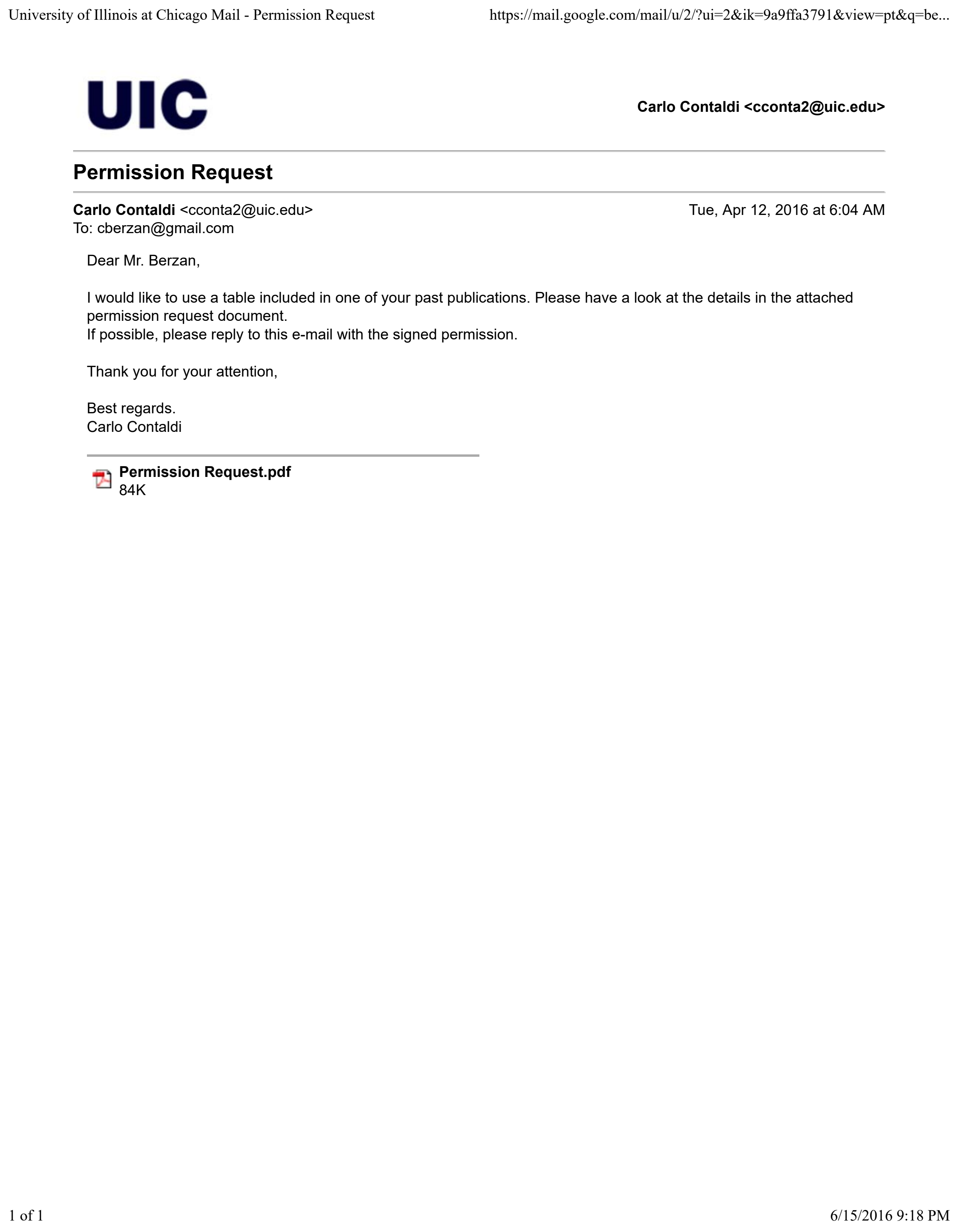
Content to be included is:
p. 529 Fig. 14.12a

Please credit our material as follows:
RUSSELL, STUART; NORVIG, PETER, ARTIFICIAL INTELLIGENCE: A MODERN APPROACH, 3rd, ©2010, p. 529. **Reprinted by permission of Pearson Education, Inc., New York, New York.**

Sincerely,
Vineta Lewis, Permissions Supervisor

2 Constantin Berzan

In this section it is reported the permission letter related to Table I. It is constituted by an e-mail exchange I had with Mr. Constantin Berzan.



University of Illinois at Chicago Mail - Permission Request https://mail.google.com/mail/u/2/?ui=2&ik=9a9ffa3791&view=pt&q=be...

Carlo Contaldi <cconta2@uic.edu>

Permission Request

Carlo Contaldi <cconta2@uic.edu> Tue, Apr 12, 2016 at 6:04 AM
To: cberzan@gmail.com

Dear Mr. Berzan,

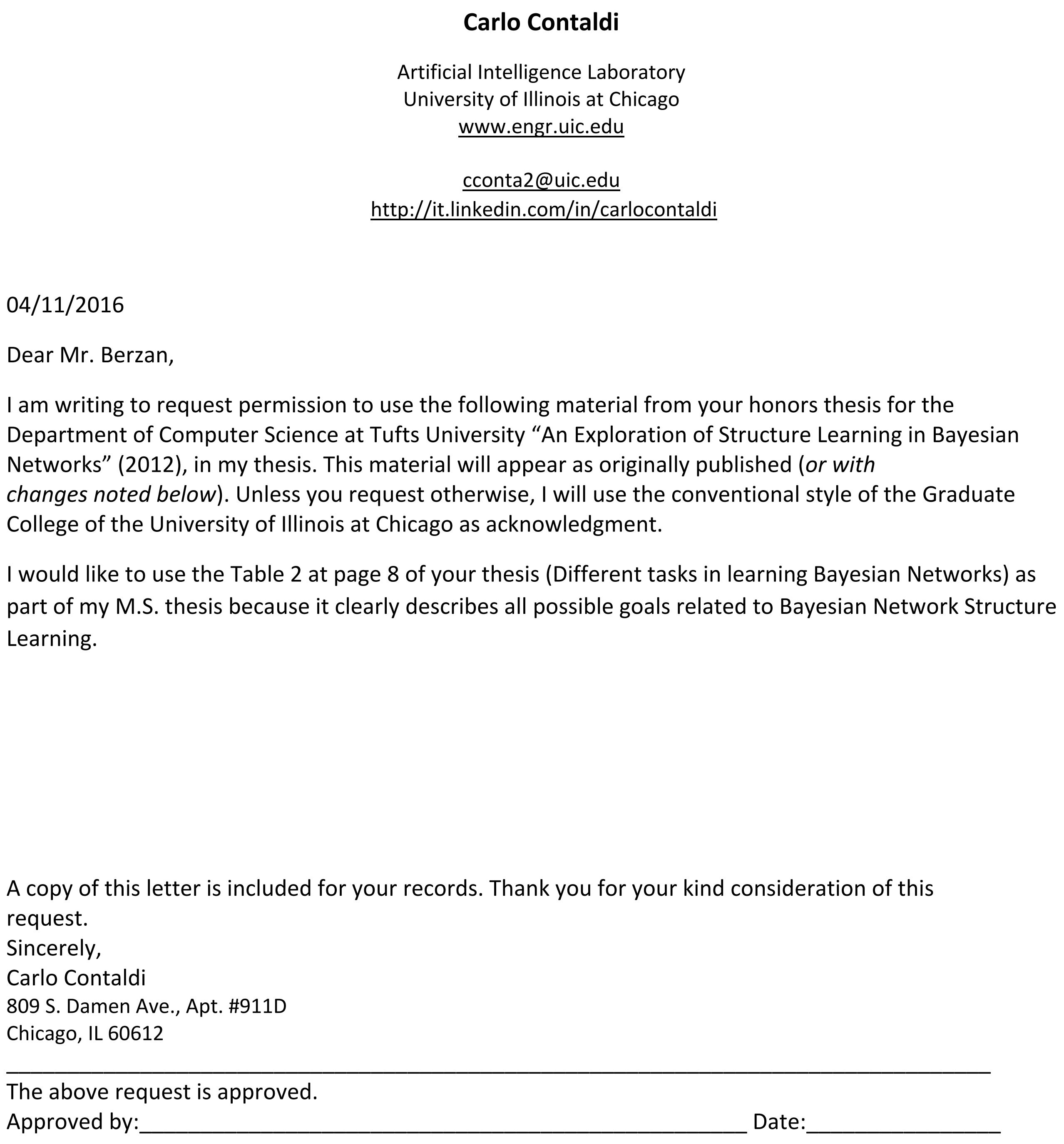
I would like to use a table included in one of your past publications. Please have a look at the details in the attached permission request document.
If possible, please reply to this e-mail with the signed permission.

Thank you for your attention,

Best regards.
Carlo Contaldi

Permission Request.pdf
84K

1 of 1 6/15/2016 9:18 PM



Carlo Contaldi

Artificial Intelligence Laboratory
University of Illinois at Chicago
www.engr.uic.edu

cconta2@uic.edu
http://it.linkedin.com/in/carlocontaldi

04/11/2016

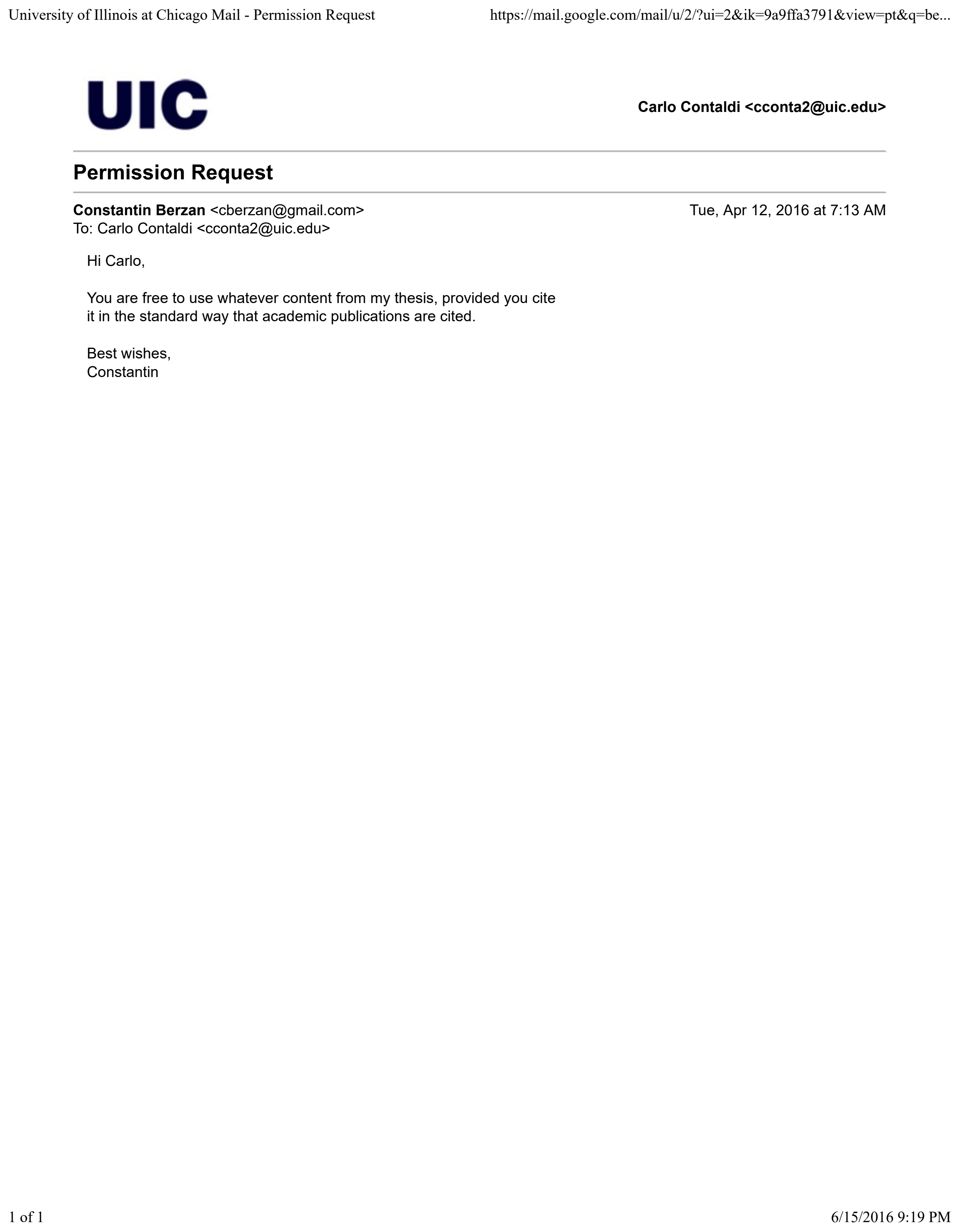
Dear Mr. Berzan,

I am writing to request permission to use the following material from your honors thesis for the Department of Computer Science at Tufts University "An Exploration of Structure Learning in Bayesian Networks" (2012), in my thesis. This material will appear as originally published (*or with changes noted below*). Unless you request otherwise, I will use the conventional style of the Graduate College of the University of Illinois at Chicago as acknowledgment.

I would like to use the Table 2 at page 8 of your thesis (Different tasks in learning Bayesian Networks) as part of my M.S. thesis because it clearly describes all possible goals related to Bayesian Network Structure Learning.

A copy of this letter is included for your records. Thank you for your kind consideration of this request.
Sincerely,
Carlo Contaldi
809 S. Damen Ave., Apt. #911D
Chicago, IL 60612

The above request is approved.
Approved by:____ Date:________________



University of Illinois at Chicago Mail - Permission Request https://mail.google.com/mail/u/2/?ui=2&ik=9a9ffa3791&view=pt&q=be...

UIC **Carlo Contaldi <cconta2@uic.edu>**

Permission Request

Constantin Berzan <cberzan@gmail.com> Tue, Apr 12, 2016 at 7:13 AM
To: Carlo Contaldi <cconta2@uic.edu>

Hi Carlo,

You are free to use whatever content from my thesis, provided you cite
it in the standard way that academic publications are cited.

Best wishes,
Constantin

1 of 1 6/15/2016 9:19 PM

CITED LITERATURE

1. Russell, S. J. and Norvig, P.: Artificial Intelligence, A Modern Approach - Third Edition. Upper Saddle River, New Jersey, Prentice Hall, 2010.

2. Rietz, H. L.: Review: Grundbegriffe der wahrscheinlichkeitsrechnung by a. kolmogoroff. Bull. Amer. Math. Soc., 40:522–523, 1934.

3. Stuart, A. and Ord, K.: Kendall's Advanced Theory of Statistics: Volume I – Distribution Theory. Edward Arnold, 1994.

4. Pearl, J.: Causality: Models, Reasoning, and Inference. Cambridge University Press, 2000.

5. Diestel, R.: Graph Theory. Springer-Verlag, 1997.

6. Kalisch, M. and Bühlmann, P.: Estimating high-dimensional directed acyclic graphs with the pc-algorithm. Journal of Machine Learning Research, 8:613–636, 2007.

7. Spirtes, P., Glymour, C., and Scheines, R.: Causation, Predition and Search - 2nd edition. Mit Press, 2000.

8. Verma, T. S. and Pearl, J.: Equivalence and synthesis of causal models. Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence, 6:255–270, June 1990.

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

10. Andersson, S. A., Madigan, D., and Perlman, M. D.: A characterization of markov equivalence classes for acyclic digraphs. Ann. Statist., 25:505–541, 1997.

11. Satyanarayana, B. and Prasad, K. S.: Discrete Mathematics and Graph Theory. PHI Learning Pvt. Ltd., 2014.

12. Berzan, C.: An Exploration of Structure Learning in Bayesian Networks. Doctoral dissertation, Tufts University, 2012.

13. Whitehead, A.: Process and reality. an essay in cosmology. In Gifford Lectures Delivered in the University of Edinburgh During the Session 1927–1928, 1929.

14. Chickering, D. M., Heckerman, D., and Meek, C.: Large-sample learning of bayesian networks is np-hard. Journal of Machine Learning Research, 5:1287–1330, October 2004.

15. Vafaee, F.: Learning the structure of large-scale bayesian networks using genetic algorithm. In Genetic and Evolutionary Computation Conference, July 2014.

16. Murphy, K.: The bayes net toolbox for matlab. Computing science and statistics, 33:1024–1034, 2001.

17. Margaritis, D.: Learning Bayesian Network Model Structure from Data. Doctoral dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2003.

18. Kruse, R. and Borgelt, C.: Data mining with graphical models. In Proceedings of the 5th International Conference on Discovery Science, pages 2–11, 2002.

19. Pearl, J.: Causality. Cambridge University Press, 2009.

20. Berkson, J.: Limitations of the application of fourfold table analysis to hospital data. Biometrics Bulletin, 2:47–53, June 1946.

21. Koller, D. and Friedman, N.: Probabilistic Graphical Models – Principles and Techniques. MIT Press, Adaptive Computation and Machine Learning series, 2009.

22. Vafaee, F.: Controlling Genetic Operator Rates in Evolutionary Algorithms. Doctoral dissertation, University of Illinois at Chicago, July 2010.

23. Uusitalo, L.: Advantages and challenges of bayesian networks in environmental modelling. Ecological Modelling, 203:312–318, 2007.

24. Heckerman, D., Geiger, D., and Chickering, D. M.: Learning bayesian networks: The combination of knowledge and statistical data. Machine Learning, 20:197–243, 1995.

CITED LITERATURE (continued)

25. Kontkanen, P., Myllymki, P., Silande, T., and Tirri, H.: Comparing predictive inference methods for discrete domains. In Proceedings of the sixth International Workshop on Artificial Intelligence and Statistics, pages 311–318, 1997.

26. Kuikka, S., Hildén, M., Gislason, H., Hansson, S., Sparholt, H., and Varis, O.: Modeling environmentally driven uncertainties in baltic cod (gadus morhua) management by bayesian influence diagrams. Can. J. Fish. Aquat. Sci., 56:629–641, 1999.

27. Marcot, B., Holthausen, R., Raphael, M., Rowland, M., and Wisdom, M.: Using bayesian belief networks to evaluate fish and wildlife population viability under land management alternatives from an environmental impact statement. Forest Ecol. Manage., 153:29–42, 2001.

28. Jensen, F. V.: Bayesian Networks and Decision Graphs. Springer-Verlag, 2001.

29. Walters, C. and Martell, S.: Fisheries Ecology and Management. Princeton University Press, 2004.

30. Jr., C. E. K., Roberts, L. M., k. A. Shaffer, and Haddawy, P.: Construction of a bayesian network for mammographic diagnosis of breast cancer. Computers Biol. Med., 27:19–29, 1997.

31. Xia, J., Neapolitan, R., Barmada, M. M., and Visweswaran, S.: Learning genetic epistatis using bayesian network scoring criteria. BMC bioinformatics, 12, 2011.

32. Sachs, K., Perez, O., and Pe'er, D.: Causal protein-signaling networks derived from multiparameter single-cell data. Science, 308:523–529, 2005.

33. Hill, S. M., Lu, Y., J.Molina, Heiser, L. M., Spellman, P. T., Speed, T. P., Gray, J. W., Mills, G. B., and Mukherjee, S.: Bayesian inference of signaling network topology in a cancer cell line. Bioinformatics, 28:2804–2810, 2012.

34. Zhang, C., Frias, M. A., Mele, A., Ruggiu, M., Eom, T., Marney, C. B., Wang, H., Licatalosi, D. D., Fak, J. J., and Darnell, R. B.: Integrative modeling defines the nova splicing-regulatory network and its combinatorial controls. Science, 329:439–443, 2010.

35. Friedman, N.: Inferring cellular networks using probabilistic graphical models. Science, 303:799–805, 2004.

36. Beaumont, M. and Rannala, B.: The bayesian revolution in genetics. Nat. Rev. Genet., 5:251–261, 2004.

37. de Campos, L. M., Fernández-Luna, J. M., and Huete, J. F.: Bayesian networks and information retrieval: an introduction to the special issue. Information Processing and Management (Elsevier), 40:727–733, 2004.

38. Varis, O., Kettunen, J., and Sirviö, H.: Bayesian influence diagram approach to complex environmental management including observational design. Computat. Stat. Data Anal., 9:77–91, 1990.

39. Lee, D. C. and Rieman, B. E.: Population viability assessment of salmonids by using probabilistic networks. N. Am. J. Fish. Manage., 17:1144–1157, 1997.

40. Varis, O.: Bayesian decision analysis for environmental and resource management. Environ. Modell. Software, 12:177–185, 1997.

41. Reckhow, K. H.: Water quality prediction and probability network models. Can. J. Fish. Aquat. Sci., 56:1150–1158, 1999.

42. Borsuk, M. E., Stow, C. A., and Reckhow, K. H.: A bayesian network of eutrophication models for synthesis, prediction, and uncertainty analysis. Ecol. Model., 173:219–239, 2004.

43. Little, L., Kuikka, S., Punt, A., Pantus, F., Davies, C., and Mapstone, B.: Information flow among fishing vessels modelled using a bayesian network. Environ. Modell. Software, 19:27–34, 2004.

44. Wooldridge, S. and Done, T.: Learning to predict large-scale coral bleaching from past events: a bayesian approach using remotely sensed data, in-situ data, and environmental proxies. Coral Reefs, 23:96–108, 2004.

45. Bromley, J., Jackson, N., Clymer, O., Giacomello, A., and Jensen, F.: The use of hugin to develop bayesian networks as an aid to integrated water resource planning. Environ. Model. Software, 20:231–242, 2005.

46. Uusitalo, L., Kuikka, S., and Romakkaniemi, A.: Estimation of atlantic salmon smolt carrying capacity of rivers using expert knowledge. ICES J. Marine Sci., 62:708–722, 2005.

CITED LITERATURE (continued)

47. Dong, A. and Agogino, A. M.: Text analysis for constructing design representations. Artif. Intell. Eng., 11:65–75, 1997.

48. Garbolino, P. and Taroni, F.: Evaluation of scientific evidence using bayesian networks. Forensic Sci. Int., 125:149–155, 2002.

49. Huang, L., Nan, J., Guo, L., and Lin, Q.: A bayesian network approach in the relevance feedback of personalized image semantic model. Advances in Multimedia, Software Engineering and Computing, 1, 2012.

50. Nikolopoulos, S., Papadopoulos, G. T., Kompatsiaris, I., and Patras, I.: Evidence-driven image interpretation by combining implicit and explicit knowledge in a bayesian network. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 41, 2011.

51. Yan, G., Lee, R., Kent, A., and Wolpert, D.: Towards a bayesian network game framework for evaluating ddos attacks and defense. In ACM Conference on Computer and Communications Security, 2012.

52. Vafaee, F., Turán, G., Nelson, P. C., and Berger-Wolf, T. Y.: Among-site rate variation: Adaptation of genetic algorithm mutation rates at each single site. In GECCO'14, 2014.

53. Vafaee, F., Turán, G., Nelson, P. C., and Berger-Wolf, T. Y.: Balancing the exploration and exploitation in an adaptive diversity guided genetic algorithm. In CEC'14, 2014.

54. Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M.: Occam's razor. Information Processing Letters, 24:377–380, 1987.

55. Cheng, J., Bell, D., and Liu, W.: Learning bayesian networks from data: An efficient approach based on information theory. In Proceedings of the 6th ACM International Conference on Information and Knowledge Management, 1997.

56. Chow, C. K. and Liu, C. N.: Approximating discrete probability distributions with dependence trees. IEEE Trans. Inform. Theory, 14:462–467, 1968.

57. Pellet, J. and Elisseeff, A.: Using markov blankets for causal structure learning. The Journal of Machine Learning Research, 9:1295–1342, 2008.

58. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, 1988.

59. Yehezkel, R. and Lerner, B.: Bayesian network structure learning by recursive autonomy identification. Journal of Machine Learning Research, 10:1527–1570, 2009.

60. Villanueva, E. and Maciel, C. D.: Efficient methods for learning bayesian network superstructures. Neurocomputing, 123:3–12, 2014.

61. Steck, H. and Tresp, V.: Bayesian belief network for data mining. In Proceedings of the second DMDW, 1999.

62. Liu, Z., Malone, B., and Yuan, C.: Empirical evaluation of scoring functions for bayesian network model selection. In Proceedings of MCBIOS Conference, 2012.

63. Cooper, G. F. and Herskovits, E.: A bayesian method for the induction of probabilistic networks from data. Machine Learning, 9:309–347, 1992.

64. Heckerman, D.: A tutorial on learning with bayesian networks. Studies in Computational Intelligence, 156:33–82, 1998.

65. Glover, F.: Tabu search: A tutorial. Interfaces, 20:74–94, 1990.

66. Friedman, N., Nachman, I., and Peér, D.: Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm. Uncertainty in Artificial Intelligence, pages 206–215, 1999.

67. Chickering, D. M.: Learning equivalence classes of bayesian network structures. J Mach Learn Res, 2:445–498, 2002.

68. Moore, A. and Wong, W. K.: Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning. In Proceedings of the 20th International Conference on Machine Learning (ICML '03), pages 552–559, 2003.

69. Koivisto, M. and Sood, K.: Exact bayesian structure discovery in bayesian networks. Journal of Machine Learning Research, pages 549–573, 2004.

70. Silander, T. and Myllymaki, P.: A simple approach for finding the globally optimal bayesian network structure. In Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence, 2006.

71. Malone, B., Yuan, C., and Hansen, E. A.: Memory-efficient dynamic programming for learning optimal bayesian networks. In Proceedings of the 25th AAAI Conference on Artificial Intelligence, pages 1057–1062, 2011.

72. Ott, S., Imoto, S., and Miyano, S.: Finding optimal models for small gene networks. Pac Symp Biocomput 2004, pages 557–567, 2004.

73. Singh, A. and Moore, A.: Finding optimal bayesian networks by dynamic programming. Technical report, Carnegie Mellon University, 2005.

74. de Campos, C. P., Zeng, Z., and Ji, Q.: Structure learning of bayesian networks using constraints. In Proceedings of the 26th Annual International Conference on Machine Learning, pages 113–120, 2009.

75. Jaakkola, T., Sontag, D., Globerson, A., and Meila, M.: Learning bayesian network structure using lp relaxations. In Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, 2010.

76. Cussens, J.: Bayesian network learning with cutting planes. In Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence, 2011.

77. Yuan, C., Malone, B., and Wu, X.: Learning optimal bayesian networks using a $\times$ search. In Proceedings of the 22nd International Joint Conference on Artificial Intelligence, pages 2186–2191, 2011.

78. Malone, B., Yuan, C., Hansen, E., and Bridges, S.: Improving the scalability of optimal bayesian network learning with external-memory frontier breadth-first branch and bound search. In Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence, pages 479–488, 2011.

79. Yuan, C. and Malone, B.: An improved admissible heuristic for finding optimal bayesian networks. In Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence, 2012.

80. Borgelt, C. and Kruse, R.: An empirical investigation of the k2 metric. In Proc. of the 6th European Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, pages 240–251, 2001.

CITED LITERATURE (continued)

81. Friedman, N., Geiger, D., and Goldszmidt, M.: Bayesian networks classifiers. Machine Learning, 29:131–163, 1997.

82. Buntine, W.: Operations for learning with graphical models. Journal of Artificial Intelligence Research, 2:159–225, 1994.

83. Teyssier, M. and Koller, D.: Ordering-based search: A simple and effective algorithm for learning bayesian networks. In Proceedings of the Twenty-first Conference on Uncertainty in AI (UAI), pages 584–590, July 2005.

84. Mkenyeleye, M.: MCMC Analysis for Optimization of Stochastic Models. Doctoral dissertation, Lappeenranta University of Technology, November 2011.

85. Carrillo, M. A., Ortiz, F. J. C., Morales-Menéndez, R., and non, L. E. G. C.: Learning bayesian network structures from small datasets using simulated annealing and bayesian score. In IASTED International Conference on Artificial Intelligence and Applications, part of the 23rd Multi-Conference on Applied Informatics, 2005.

86. Friedman, N. and Koller, D.: Being bayesian about network structure: A bayesian approach to structure discovery in bayesian networks. In Proc. Conference on Uncertainty in Artificial Intelligence, pages 201–210, 2000.

87. Antal, P., Millinghoffer, A., and Hullam, G.: Structure learning of bayesian networks with mcmc: extension to incomplete data and decision trees as local models. IEEE Signal Processing Magazine, 2009.

88. Ye, S., Cai, H., and Sun, R.: An algorithm for bayesian networks structure learning based on simulated annealing with mdl restriction. In International Conference on Computing, Networking and Communications, 2013.

89. Wang, T., Touchman, J. W., and Xue, G.: Applying two-level simulated annealing on bayesian structure. In Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference, 2004.

90. Hesar, A. S.: Structure learning of bayesian belief networks using simulated annealing algorithm. Middle-East Journal of Scientific Research, 18:1343–1348, 2013.

91. Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P.: Optimization by simulated annealing. Science, 220:671–680, 1983.

CITED LITERATURE (continued)

92. Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R. H., and Kuijpers, C. M. H.: Structure learning of bayesian networks by genetic algorithms: A performance analysis of control parameters. IEEE Transactions on Pattern Analysis and Machine Intelligence, 18:912–926, 1996.

93. Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., and Yurramendi, Y.: Learning bayesian network structures by searching for the best ordering with genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics, 26:487–493, 1996.

94. Wong, M. L., Lee, S. Y., and Leung, K. S.: Data mining of bayesian networks using cooperative coevolution. Decis. Support Syst., 38:451–472, December 2004.

95. Kabli, R., Herrmann, F., and McCall, J.: A chain-model genetic algorithm for bayesian network structure learning. In GECCO '07, 2007.

96. Carvalho, A.: A cooperative coevolutionary genetic algorithm for learning bayesian network structures. In Proceedings of the 13th Genetic and Evolutionary Computation Conference, pages 1131–1138, 2011.

97. Tonda, A. P., Lutton, E., Reuillon, R., Squillero, G., and Wuillemin, P.-H.: Bayesian network structure learning from limited datasets through graph evolution. EuroGP, pages 254–265, 2012.

98. Sourceforge: Host of μgp3, http://sourceforge.net/projects/ugp3.

99. Singh, M. and Valtorta, M.: Construction of bayesian network structures from data: a brief survey and an efficient algorithm. International Journal of Approximate Reasoning, pages 259–265, 1995.

100. Rijsbergen, C. J. V.: Information Retrieval (2nd ed.). Butterworth, 1979.

101. Hobson, A.: Concepts in statistical mechanics. Gordon and Breach, 1971.

102. Cover, T. M. and Thomas, J. A.: Elements of Information Theory - 2nd edition. Wiley Series in Telecommunications and Signal Processing, 1991.

103. Wyner, A. D.: A definition of conditional mutual information for arbitrary ensembles. Information and Control, 38:51–59, 1978.

CITED LITERATURE (continued)

104. Pearson, K.: On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. Philosophical Magazine Series, 5:157–175, 2013.

105. McDonald, J. H.: Handbook of Biological Statistics (Third ed.). Baltimore, Maryland: Sparky House Publishing, 2014.

106. MacKay, D. J.: Bayes or chi-squared? or does it not matter?, July 2005.

107. Sokal, R. R. and Rohlf, F. J.: Biometry: The Principles and Practices of Statistics in Biological Research - 3rd edition. W.H. Freeman, 1994.

108. Leray, P. and Francois, O.: Bnt structure learning package: Documentation and experiments. Technical report, Laboratoire PSI, Universitè et INSA de Rouen, 2004.

109. Carvalho, A. M.: Scoring functions for learning bayesian networks. Inesc-id Tec. Rep, 2009.

110. Buntine, W. L.: Theory refinement on bayesian networks. In Proc. UAI'91, pages 52–60, 1991.

111. Shannon, C. E.: A mathematical theory of communication. Bell System Technical Journal, pages 379–423, 1948.

112. Rissanen, J.: Stochastic complexity and modeling. Annals of Statistics, 14:1080–1100, 1986.

113. Akaike, H.: A new look at the statistical model identification. IEEE Transactions on Automatic Control, 19:716–723, 1974.

114. Kontkanen, P. and Myllymäki, P.: A linear-time algorithm for computing the multinomial stochastic complexity. Inf. Process. Lett., 103:227–233, 2007.

115. Roos, T., Silander, T., Kontkanen, P., and Myllymäki, P.: Bayesian network structure learning using factorized nml universal models. In Proceedings ITA'08, 2008.

116. Scutari, M.: Bayesian network repository, March 2016.

117. Beinlich, I. A., Suermondt, H. J., Chavez, R. M., and Cooper, G. F.: The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In Proceedings of the 2nd European Conference on Artificial Intelligence in Medicine, pages 247–256, 1989.

118. Turing, A. M.: Computing machinery and intelligence. Mind, 238:433–460, 1950.

119. Fraser, A.: Simulation of genetic systems by automatic digital computers. Aust. J. Biol. Sci., 10:484–491, 1957.

120. Rechenberg, I.: Evolutionsstrategie. Holzmann-Froboog, 1973.

121. Fogel, L. J., Owens, A., and Walsh, M.: An evolutionary prediction technique. In IEEE International Symposium on Microwaves, Circuit Theory, and Information Theory, pages 173–174, 1964.

122. Holland, J. H.: Adaptation in natural and artificial systems. In MI: The Univ. of Michigan Press, 1975.

123. Eshelman, L. J., Caruna, R. A., and Schaffer, J. D.: Biases in the crossover landscape. In Proc. 3rd Int. Conf. on Genetic Algorithms, pages 10–19, 1989.

124. Syswerda, G.: Uniform crossover in genetic algorithms. In Proc. 3rd Int. Conf. on Genetic Algorithms, 1989.

125. Goldberg, D. E.: Genetic algorithms in search, optimization and machine learning. Addison Wesley, 1989.

126. Baker, J. E.: Adaptive selection methods for genetic algorithms. In Proc. 1st Int. Conf. on Genetic Algorithms and Their Applications, pages 101–111, 1985.

127. Goldberg, D. E., Korb, B., and Deb, K.: Messy genetic algorithms: Motivation, analysis and first results. Complex Syst. 3, 5:493–530, 1989.

128. Eades, P., Lin, X., and Smyth, W. F.: A fast and effective heuristic for the feedback arc set problem. Information Processing Letters, 47:319–323, 1993.

129. Tsutsui, S., Ghosh, A., Corne, D., and Fujimoto, Y.: A real coded genetic algorithm with an explorer and an exploiter populations. In Proc. of the 73th Int. Con. on GAs, pages 238–245, 1997.

130. Beyer, H. G.: On the explorative power of es/ep-like algorithms. In Proceedings of the 7th Annual Conference on Evolutionary Programming, 1998.

131. Eiben, A. E. and Schipper, C. A.: On evolutionary exploration and exploitation. Fundamenta Informaticae, 35:35–50, 1998.

132. Das, M. K. and Dai, H. K.: A survey of dna motif finding algorithms. BMC Bioinformatics, page 8, 2007.

133. Binder, J., Koller, D., Russell, S., and Kanazawa, K.: Adaptive probabilistic networks with hidden variables. Machine Learning, 29:213–244, 1997.

134. Onisko, A.: Probabilistic Causal Models in Medicine: Application to Diagnosis of Liver Disorders. Doctoral dissertation, Institute of Biocybernetics and Biomedical Engineering, Polish Academy of Science, Warsaw, March 2003.

135. Bobrowski, L.: Hepar: Computer system for diagnosis support and data analysis. In Prace IBIB 31, 1992.

136. Shan, C.: Converting bayes nets from bif format to bnt format: bif2bnt program, January 2014.

137. Intel: Core i7 processors, http://www.intel.com/content/www/us/en/processors/core/core-i7-processor.html, April 2016.

138. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin, 1:80–83, 1945.

139. Gibbons, J. D. and Chakraborti, S.: Nonparametric Statistical Inference, 5th Ed.. Chapman & Hall/CRC Press, Taylor & Francis Group, 2011.

140. Hubbard, R.: Blurring the distinctions between p's and a's in psychological research. Theory Psychology, 14:295–327, 2004.

141. Murphy, P. M. and Aha, D. W.: Uci repository of machine learning databases, machine-readable data repository.

142. Geiger, D., Verma, T. S., and Pearl, J.: Identifying independence in bayesian networks. Networks, 20:507–534, 1990.

CITED LITERATURE (continued)

143. Duan, R., Yang, Y., and Li, G.: Controlling the inconsistent of the bayesian network structure learning with the recursive autonomy identification. In Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics, 2014.

144. Grossman, D. and Domingos, P.: Learning bayesian network classifiers by maximizing conditional likelihood. In Proceedings of the 21th International Conference on Machine Learning, 2004.

145. Kontkanen, P., Myllymaki, P., Sliander, T., and Tirri, H.: On supervised selection of bayesian networks. In Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, 1999.

146. Dobrushin, R. L.: General formulation of shannon's main theorem in information theory. Ushepi Mat. Nauk, 14:3–104, 1959.

147. Messaoud, M. B., Leray, P., and Amor, N. B.: Integrating ontological knowledge for iterative causal discovery and visualization. In ECSQUARU, pages 168–179, 2009.

148. Lauritzen, S. and Spiegelhalter, D.: Local computation with probabilities on graphical structures and their application to expert systems (with discussion). Journal of the Royal Statistical Society: Series B (Statistical Methodology), 50:157–224, 1988.

VITA

NAME	Carlo Contaldi
INTERESTS	
	Artificial Intelligence
	Machine Learning - Pattern Recognition, Data Mining
EDUCATION	
	Master of Science in Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, Illinois, USA, 2016
	Master of Science in Embedded Systems Computer Engineering, Politecnico di Torino, Turin, Italy, 2016
	Bachelor of Science in Computer Engineering, Politecnico di Torino, Turin, Italy, 2014
LANGUAGE SKILLS	
Italian	Native speaker
English	Full working proficiency
	A.Y. 2015/16 One Year of study abroad in Chicago, Illinois
	A.Y. 2014/15. Lessons and exams attended exclusively in English
	2014 - IELTS examination (6.5/9)
SCHOLARSHIPS	
Spring 2016	Research Assistantship (RA) position (10 hours/week) with monthly stipend at UIC
Fall 2015	Scholarship for joint double-degree program TOP-UIC students at Polytechnic of Turin
Fall 2015	Tuition Fees Reduction for merit at Polytechnic of Turin
Fall 2014	Tuition Fees Reduction for merit at Polytechnic of Turin
AWARDS	
Spring 2012	Ranked among excellent students at SATs, and consequently awarded with a netbook via the “Vinci un PC” (Win a PC) competition

VITA (continued)

	Polytechnic of Turin, Italy
TECHNICAL SKILLS	
Technical Knowledge	Problem Solving, Kernel Programming, Machine Learning, Data Mining, Data Analysis, Concurrent Programming, Automatic Controls, IC Design
Programming Languages	C, Java, Assembly x86, VHDL
IDE	Eclipse, MATLAB, Microsoft Visual Studio, MASM, LaTeX
OS	UNIX, Embedded Linux, Linux Kernel, WIN32 API
PUBLICATIONS	
2015	Graupe, D., Contaldi, C., Sattiraju, A. *Comparison of Lamstar NN & Convolutional NN Character Recognition.* Chicago, IL: University of Illinois.
WORK EXPERIENCE AND PROJECTS	
Oct 2015 - Dec 2015	A LAMSTAR Neural Network for Sentiment Classification

A semi-supervised learning approach involving a Word Embedding strategy in conjunction with an implementation of the LAMSTAR Neural Network has been developed and used to classify a corpus of tweets with respect to their sentiment polarity.

- Selected the "Twitter Sentiment Analysis Dataset" as target corpus and splitted in pre-training, training and test datasets
- Pre-processed the target corpus through a carefully tailored polishing pipeline in order to make tweets suitable for learning
- Set up and utilized a Word Embedding tool to build a new language model for tweets representation in order to enable them for use as input to a neural network
- Implemented and set up an ad-hoc version of LAMSTAR Neural Network for actual classification
- Analyzed results and compared performance with respect to other supervised methods

VITA (continued)

Feb 2015 - Jul 2015	Top-Down Design of a DLX RISC Processor

The project consisted in developing a DLX processor through all phases of the design flow, from its RTL description until its physical implementation.

- Developed working prototypes for datapath components and control unit through coordinated teamwork
- Collaborated with external consultant to assemble processor on basis of multiple metrics
- Optimized and enhanced each component w.r.t. expected overall implementation in experimental fashion
- Simulated processor through a worst-case benchmarking approach
- Produced processor physical design outcome, together with documentation and schematics

Feb 2014	Implementation of a Range Sensor on a Microcontroller

- Added system calls set to kernel of an Embedded Linux operating system template (loaded on the programmable board) to allow interfacing with ultrasonic range module
- Developed custom loadable kernel module for Embedded Linux kernel, to enhance flexibility and portability

Feb 2013 - Jul 2013	Markovian Classifier for Gestures Extraction from EMG Data

A machine learning algorithm based on Hidden Markov Models has been implemented for extracting hand gestures from a large database of electromyographic signals.

- Characterized differences for concerns of gestures recording and training between disabled and healthy subjects

VITA (continued)

- Analyzed given database to accurately devise learning strategy and signal features to be extracted on basis of computational power availability, data redundancy, noise and deviations from expected pattern
- Set environment by choosing IDE (MATLAB) and suitable tools
- Implemented a classifier algorithm based on Hidden Markov Models and Viterbi Path, resulting in relevant matching performance
- Optimized algorithm by fitting it for real-time purposes

Jan 2013 - Jun 2013	Politecnico di Torino – Polincontri Classica (classical music concerts) Usher and Attendant

- Supervised entrance and accommodated customers with a pleasant demeanor
- Promptly assisted customers in front-desk operations and during concert performances
- Guaranteed silence during performances by monitoring door opening and by handling noisy occurrences