Scalable Learning of Bayesian Networks Using Feedback Arc Set-Based Heuristics

Pierre Gillot

Thesis for the degree of Philosophiae Doctor (PhD)
University of Bergen, Norway
2023

UNIVERSITY OF BERGEN

Scalable Learning of Bayesian Networks Using Feedback Arc Set-Based Heuristics

Pierre Gillot

Thesis for the degree of Philosophiae Doctor (PhD)
at the University of Bergen

Date of defense: 10.11.2023

Year: 2023

Title: Scalable Learning of Bayesian Networks Using Feedback Arc Set-Based Heuristics

Name: Pierre Gillot

Print: Skipnes Kommunikasjon / University of Bergen

Scientific environment

The work for this PhD was funded by the University of Bergen and was conducted at the Machine Learning Group, part of the Department of Informatics at the University of Bergen.

Parts of this work have been done in the context of CEDAS (Center for Data Science, University of Bergen, UiB). Some of the computations were performed on resources provided by UNINETT Sigma2 - the National Infrastructure for High Performance Computing and Data Storage in Norway.

Acknowledgements

I would like to address my deepest thanks and appreciation to my PhD supervisor, Pekka Parviainen, for his great advising and generosity throughout the duration of my PhD. This long journey was certainly not without challenge for the both of us, yet I believe it contributed to shaping us into better versions of ourselves. Patience, communication, empathy, trust, openness were some of the values I'd like to believe we've come to develop during this very special time.

I also extend my gratitude to the members of the Machine Learning Group who were like a real family for me: Ketil Malde for his drive and enthusiasm, Pekka Parviainen and Nello Blaser for their selfless commitment to the wellness of the group, Troels Arnfred Bojesen and Ricardo Guimarães for their kindness and altruism, Odin Hoff Gardå for his calm and sharp sense of humour and great taste in beverage, Philip Andreas Turk for his collected self and communicative positive energy, Erlend Raa Vågset and Victor Lacerda Botelho for their good heart, humanity and hilarious never-ending debates, Emmanuel Sam for his humility and for being such a great office-mate, Madhumita Kundu for her wit and endearing blend of sensibility and bluntness, Cosimo Damiano Persia for his inspiring brightness and strong sense of morality, and Natacha Galmiche for her deep empathy and the mutual trust we have built over the past years.

Most importantly, my deepest gratitude goes to Margaux Susman for her unwavering faith, support, respect, loyalty and appreciation. I will always keep very fond memories of our adventures here in Bergen. Thank you so much !

Abstract in English

Bayesian networks form an important class of probabilistic graphical models. They consist of a structure (a directed acyclic graph) expressing conditional independencies among random variables, as well as parameters (local probability distributions). As such, Bayesian networks are generative models encoding joint probability distributions in a compact form.

The main difficulty in learning a Bayesian network comes from the structure itself, owing to the combinatorial nature of the acyclicity property; it is well known and does not come as a surprise that the structure learning problem is NP-hard in general. Exact algorithms solving this problem exist: dynamic programming and integer linear programming are prime contenders when one seeks to recover the structure of small-to-medium sized Bayesian networks from data. On the other hand, heuristics such as hill climbing variants are commonly used when attempting to approximately learn the structure of larger networks with thousands of variables, although these heuristics typically lack theoretical guarantees and their performance in practice may become unreliable when dealing with large scale learning.

This thesis is concerned with the development of scalable methods tackling the Bayesian network structure learning problem, while attempting to maintain a level of theoretical control. This was achieved via the use of related combinatorial problems, namely the maximum acyclic subgraph problem and its dual problem the minimum feedback arc set problem. Although these problems are NP-hard themselves, they exhibit significantly better tractability in practice. This thesis explores ways to map Bayesian network structure learning into maximum acyclic subgraph instances and extract approximate solutions for the first problem, based on the solutions obtained for the second.

Our research suggests that although increased scalability can be achieved this way, maintaining theoretical understanding based on this approach is much more challenging. Furthermore, we found that learning the structure of Bayesian networks based on maximum acyclic subgraph/minimum feedback arc set may not be the go-to method in general, but we identified a setting - linear structural equation models - in which we could exper-

imentally validate the benefits of this approach, leading to fast and scalable structure recovery with the ability to learn complex structures in a competitive way compared to state-of-the-art baselines.

Abstract in Norwegian

Bayesianske nettverk er en viktig klasse av probabilistiske grafiske modeller. De består av en struktur (en rettet asyklisk graf) som beskriver betingede uavhengighet mellom stokastiske variabler og deres parametere (lokale sannsynlighetsfordelinger). Med andre ord er Bayesianske nettverk generative modeller som beskriver simultanfordelingene på en kompakt form.

Den største utfordringen med å lære et Bayesiansk nettverk skyldes selve strukturen, og på grunn av den kombinatoriske karakteren til asyklisitetsegenskapen er det ingen overraskelse at strukturlæringsproblemet generelt er NP-hardt. Det eksisterer algoritmer som løser dette problemet eksakt: dynamisk programmering og heltalls lineær programmering er de viktigste kandidatene når man ønsker å finne strukturen til små til mellomstore Bayesianske nettverk fra data. På den annen side er heuristikk som bakkeklatringsvarianter ofte brukt når man forsøker å lære strukturen til større nettverk med tusenvis av variabler, selv om disse heuristikkene vanligvis ikke har teoretiske garantier og ytelsen i praksis kan bli uforutsigbar når man arbeider med storskala læring.

Denne oppgaven tar for seg utvikling av skalerbare metoder som takler det strukturlæringsproblemet av Bayesianske nettverk, samtidig som det forsøkes å opprettholde et nivå av teoretisk kontroll. Dette ble oppnådd ved bruk av relaterte kombinatoriske problemer, nemlig det maksimale asykliske subgrafproblemet (maximum acyclic subgraph) og det duale problemet (feedback arc set). Selv om disse problemene er NP-harde i seg selv, er de betydelig mer håndterbare i praksis. Denne oppgaven utforsker måter å kartlegge Bayesiansk nettverksstrukturlæring til maksimale asykliske subgrafforekomster og trekke ut omtrentlige løsninger for det første problemet, basert på løsninger oppnådd for det andre.

Vår forskning tyder på at selv om økt skalerbarhet kan oppnås på denne måten, er det adskillig mer utfordrende å opprettholde den teoretisk forståelsen med denne tilnærmingen. Videre fant vi ut at å lære strukturen til Bayesianske nettverk basert på maksimal asyklisk subgraf kanskje ikke er den beste metoden generelt, men vi identifiserte en kontekst - lineære strukturelle ligningsmodeller - der vi eksperimentelt kunne validere fordelene

med denne tilnærmingen, som fører til rask og skalerbar identifisering av strukturen og med mulighet til å lære komplekse strukturer på en måte som er konkurransedyktig med moderne metoder.

Articles

I. Pierre Gillot & Pekka Parviainen, *Scalable Bayesian Network Structure Learning via Maximum Acyclic Subgraph*, Proceedings of the 10th International Conference on Probabilistic Graphical Models, PMLR 138:209-220, 2020.

II. Pierre Gillot & Pekka Parviainen, *Learning Large DAGs by Combining Continuous Optimization and Feedback Arc Set Heuristics*, Proceedings of the AAAI Conference on Artificial Intelligence, 36(6):6713-6720, 2022.

III. Pierre Gillot & Pekka Parviainen, *Convergence of Feedback Arc Set-Based Heuristics for Linear Structural Equation Models*, Proceedings of the 11th International Conference on Probabilistic Graphical Models, PMLR 186:157-168, 2022.

Algorithms

Figures

Tables

Symbols

- Trace on $\mathbb{R}^{n\times n}$:
$$\mathcal{T}r : X \in \mathbb{R}^{n\times n} \mapsto \sum_{k=0}^{n-1} X[k,k]$$

- Scalar product on $\mathbb{R}^n$:
$$\langle\cdot,\cdot\rangle : (x,y) \in (\mathbb{R}^n)^2 \mapsto x^t y = \sum_{k=0}^{n-1} x[k]y[k]$$

- Scalar product on $\mathbb{R}^{m\times n}$:
$$\langle\cdot,\cdot\rangle : (X,Y) \in (\mathbb{R}^{m\times n})^2 \mapsto \mathcal{T}r(X^t Y) = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} X[i,j]Y[i,j]$$

- L_2 norm on $\mathbb{R}^n$ (*Euclidean* norm):
$$\|\cdot\| : x \in \mathbb{R}^n \mapsto \sqrt{\langle x,x\rangle} = \sqrt{\sum_{k=0}^{n-1}\big(x[k]\big)^2}$$

- L_2 norm on $\mathbb{R}^{m\times n}$ (*Frobenius* norm):
$$\|\cdot\| : X \in \mathbb{R}^{m\times n} \mapsto \sqrt{\langle X,X\rangle} = \sqrt{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\big(X[i,j]\big)^2}$$

- L_1 norm on $\mathbb{R}^n$ (*Manhattan* norm):
$$\|\cdot\|_1 : x \in \mathbb{R}^n \mapsto \sum_{k=0}^{n-1}\big|x[k]\big|$$

- L_1 norm on $\mathbb{R}^{m\times n}$:

$$\|\cdot\|_1 : X \in \mathbb{R}^{m\times n} \mapsto \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} \big|X[i,j]\big|$$

- Integer interval from m to $m+n$:

$$[\![m, m+n]\!] \coloneqq \{m, m+1, \dots, m+n-1, m+n\}$$

- Open ball in the metric space $\big(\mathbb{R}^n, \|\cdot\|\big)$ centered at $x \in \mathbb{R}^n$ with radius $r > 0$:

$$B(x,r) \coloneqq \big\{y \in \mathbb{R}^n, \|y-x\| < r\big\}$$

- Interior of a set S in the metric space $\big(\mathbb{R}^n, \|\cdot\|\big)$:

$$\mathring{S} \coloneqq \bigcup_{\substack{\mathcal{O} \text{ open:} \\ \mathcal{O}\subset S}} \mathcal{O} \ = \big\{x \in \mathbb{R}^n : \ \exists r > 0 \text{ such that } B(x,r) \subset S\big\}$$

- Closure of a set S in the metric space $\big(\mathbb{R}^n, \|\cdot\|\big)$:

$$\overline{S} \coloneqq \bigcap_{\substack{\mathcal{C} \text{ closed:} \\ S\subset\mathcal{C}}} \mathcal{C} \ = \Big\{x \in \mathbb{R}^n : \ \exists (x_k)_k \in S^{\mathbb{N}} \text{ such that } \|x - x_k\| \underset{k\to+\infty}{\longrightarrow} 0\Big\}$$

- Gradient at point $x \in \mathbb{R}^n$ of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$:

$$\nabla f(x) \coloneqq v \in \mathbb{R}^n \ \text{ such that } \lim_{\|h\|\to 0} \frac{f(x+h) - f(x) - \langle v, h\rangle}{\|h\|} = 0$$

- $(u_k)_k$ is a small "o" of (is negligible compared to) $(v_k)_k$ asymptotically:

$$u_k = o(v_k) \iff \lim_{k\to+\infty} \frac{u_k}{v_k} = 0$$

- $(u_k)_k$ is a big "o" of (is dominated by) $(v_k)_k$ asymptotically:

$$u_k = \mathcal{O}(v_k) \iff \exists c > 0, \exists k_0 \in \mathbb{N} : \ \forall k \geq k_0 \ |u_k| \leq c|v_k|$$

Contents

Chapter 1

Introduction

Machine learning is the field concerned with the design of computer programs solving specific *tasks* (usually, real-world problems that one would like to handle automatically when human expertise is limited or expensive or simply better used elsewhere), that are able to *learn* (improve in performance as measured by some metrics) via a learning process (an algorithmic and/or mathematical mechanism) when exposed to some form of *knowledge/experience* (typically, data) [Mitchell, 1997]. It has become an ubiquitous field in modern science, to the point that even the general public has become acquainted to it via the daily usage of (and interaction with) modern technologies and services that integrate machine learning through and through.

A prominent paradigm is machine learning is to adopt a probabilistic point of view: in such setting, phenomena are assumed to exhibit a level of uncertainty (that may be caused by noisy perturbations, lack of knowledge, etc...) and are conveniently represented as random variables. The goal of the machine learner is then to observe how these phenomena interact with one another in an attempt to understand and replicate holistically these interactions. Formally, one considers a set of phenomena $\{X_0, \ldots, X_{d-1}\}$ each represented as a random variable; the machine learner then aims at learning the joint probability distribution $P(X_0, \ldots, X_{d-1})$: corresponding models are called *generative* and are strictly more general than the so-called *discriminative* models that merely learn a conditional probability distribution $P(X_{\text{queried}}|X_{\text{observed}})$.

Of course, we know from the chain rule in probability theory that one can always decompose the joint probability distribution into a product of conditional probability dis-

tributions. Assuming random variables are all binary, the chain rule yields:

$$\begin{aligned} P(X_0, \ldots, X_{d-1}) &= P(X_0 | X_1, \ldots, X_{d-1}) \times P(X_1, \ldots, X_{d-1}) \\ &\vdots \\ &= \underbrace{P(X_0 | X_1, \ldots, X_{d-1})}_{2^{d-1} \text{ parameters}} \times \cdots \times \underbrace{P(X_{d-2} | X_{d-1})}_{2^1 \text{ parameters}} \times \underbrace{P(X_{d-1})}_{2^0 \text{ parameters}}, \end{aligned} \tag{1.1}$$

for a total of $\sum_{i=0}^{d-1} 2^i = 2^d - 1$ parameters to be learned, a number that grows exponentially in the number of random variables and quickly becomes intractable. Fortunately, simplifications occur in the presence of conditional independencies among random variables: for instance, if X_0 is conditionally independent from variables $X_1, \ldots, X_{d-2}$ given variable X_{d-1} $\big($we will write $X_0 \perp\!\!\!\perp \{X_1, \ldots, X_{d-2}\} \mid X_{d-1}\big)$, then the corresponding conditional probability distribution simplifies: $P(X_0 | X_1, \ldots, X_{d-1}) = P(X_0 | X_{d-1})$ and the number of parameters for this particular factor goes down from 2^{d-1} to only 2^1.

Probabilistic graphical models are generative machine learning models encoding a joint probability distribution in a *compact* form, in the sense that these models include a graphical structure encompassing conditional independencies among random variables, thus enabling aforementioned simplification of the conditional probability distribution factors. Two major classes of probabilistic graphical models are Markov random fields (MRFs [Kindermann, 1980]) and Bayesian networks (BNs [Neapolitan, 1990; Pearl, 1989]), which importantly differ in that the former uses undirected graphs whereas the latter uses directed acyclic graphs (DAGs) to encode conditional independencies; consequently, MRFs and BNs are used to model different types of dependencies among random variables. The interested reader may refer to Koller and Friedman [2009]'s book for a complete tour on probabilistic graphical models.

1.1 Bayesian networks

In this work, we are interested in the study of Bayesian networks. Formally, a BN is a mathematical object consisting of two components:

- A graphical *structure* (more precisely, a DAG) encoding conditional independencies among random variables: every random variable is represented by a node and two random variables are conditionally independent given a set of other random variables if there is no arc (directed edge) between the corresponding two nodes. Elementary structures in DAGs are depicted in Figure 1.2. The acyclicity property of DAGs is discussed in Section 2.1.

- *Parameters* (defining a set of conditional probability distributions): every random variable is parameterized by a probability distribution that is conditioned on its parent variables in the DAG.

A classical example of a Bayesian network is the *Sprinkler* model depicted in Figure 1.1. Several DAGs may exist factoring the joint probability distribution while encoding the same set of conditional independencies and such structures are said to be *Markov equivalent* (the two elementary Markov equivalence classes are depicted in Figure 1.2); the set of all possible chain rule decompositions (see Figure 1.3) forms a Markov equivalence class whose cardinality grows as a factorial of the number of nodes. As implied earlier, some dependencies among random variables cannot be expressed using a BN: Figure 1.5 gives an example where no DAG exists encoding a certain set of conditional independencies.

Although this may seem counter-intuitive owing to their directed graphical representation, BNs are not necessarily causal: an arc going from variable A to variable B does not imply that A causes B, but merely that the two random variables *may* be dependent (note that the "may" matters: one can always construct a BN where two nodes A and B are connected via an arc in the DAG, then set the parameters so as to ensure that $P(A|B) = P(A)$, such that A and B become independent). More precisely, Bayesian networks satisfy the so-called *Markov property*: "every variable is conditionally independent from its non-descendants given its parents". The exact mechanism revolves around a graphical concept named *d-separation*: given any *V-structure* (the blue structure in Figure 1.2), call the middle node a *collider*; now, given a DAG and three disjoint sets of nodes $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$, the sets $\mathcal{S}_1$ and $\mathcal{S}_2$ are said to be d-separated by the *separating set* $\mathcal{S}_3$ (we will write $\mathcal{S}_1 \perp\!\!\!\perp_d \mathcal{S}_2 \mid \mathcal{S}_3$) if all paths connecting nodes in $\mathcal{S}_1$ to nodes in $\mathcal{S}_2$ are *blocked* by $\mathcal{S}_3$, where a path is blocked if one of the following two assertions holds:

- At least one collider in the path and all its descendants are not in $\mathcal{S}_3$.
- At least one non-collider in the path is in $\mathcal{S}_3$.

Figure 1.4 illustrates d-separation. The Markov property satisfied by BNs essentially states that d-separation entails conditional independence:

$$\forall \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3 \text{ disjoints}, \quad [\mathcal{S}_1 \perp\!\!\!\perp_d \mathcal{S}_2 \mid \mathcal{S}_3 \implies \mathcal{S}_1 \perp\!\!\!\perp \mathcal{S}_2 \mid \mathcal{S}_3]\,. \tag{1.2}$$

Very importantly, the converse statement (the so-called *faithfulness assumption*) does not hold in general. When both the Markov property and the faithfulness assumption are satisfied, it is known that the true Markov equivalent class can be inferred from

samples drawn from the joint probability distribution as the sample size reaches infinity; the GES [Chickering, 2002; Meek, 1997] and PC [Spirtes et al., 2000] algorithms are two existing structure learning algorithms satisfying this guarantee (the former also requires the embedded scoring function to be consistent and the property that all DAGs in an equivalence class have the same number of parameters, while the latter additionally requires the absence of latent confounders).

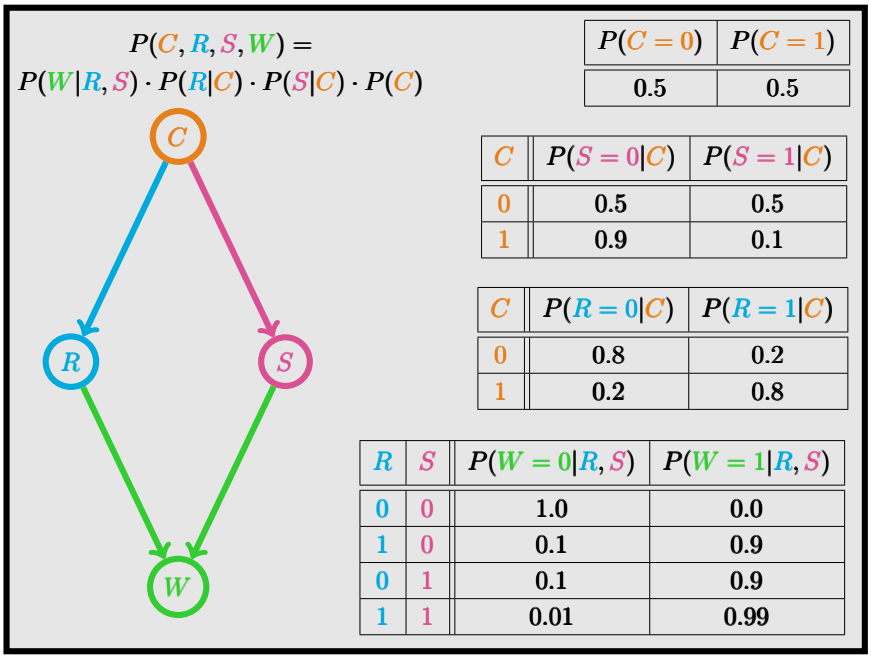

Figure 1.1: The *Sprinkler* BN is a simple Bayesian network consisting of four random binary variables: *clouds* (C), *rain* (R), *sprinkler* (S) and *wet* (W). Its structure encodes the two conditional independencies $W \perp\!\!\!\perp C \mid \{R, S\}$ and $R \perp\!\!\!\perp S \mid C$, i.e. the following simplifications occur: $P(W|C, R, S) = P(W|R, S)$ and $P(R|C, S) = P(R|C)$. The DAG provides a convenient way of writing the factorization of the joint probability distribution in a bottom-up fashion: reading nodes from the *bottom* up to the *top*, write the probability distribution factor of that node conditioned on its parents. Parameters correspond to the probability entries in the conditional probability tables.

Bayesian networks were largely studied and popularized in the celebrated work of Judea Pearl [1989] and they form a general framework encompassing widely used machine learning models, including naive Bayes [Zhang, 2004] (see Figure 1.6), hidden Markov models [Baum and Petrie, 1966], latent Dirichlet allocation [Blei et al., 2003] and mixture models [Lindsay, 1995]. Since their introduction, Bayesian networks have become a framework of choice for problems that require reasoning under uncertainty. Indeed, humans alone do not handle reasoning in systems with limited or conflicting information very well, hence the need for machine learning frameworks designed for uncertainty management.

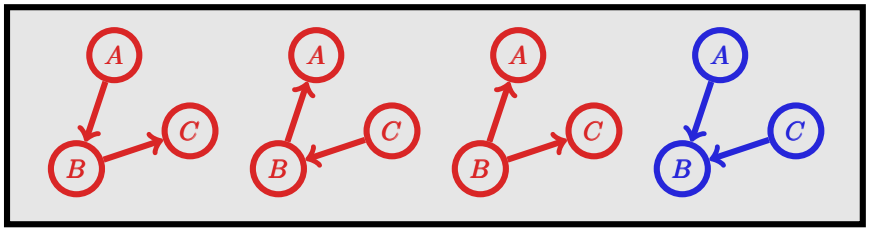

Figure 1.2: Elementary structures in a DAG: DAGs consist of four possible elementary structures corresponding to two elementary Markov equivalence classes. Structures in red are Markov equivalent i.e. they entail the same set of conditional independencies $\{A \perp\!\!\!\perp C \mid B\}$. The structure in blue is called a *V-structure* (sometimes called *immorality* instead) and is the unique member of its Markov equivalence class which entails the set of conditional independencies $\{A \perp\!\!\!\perp C \mid \emptyset\}$.

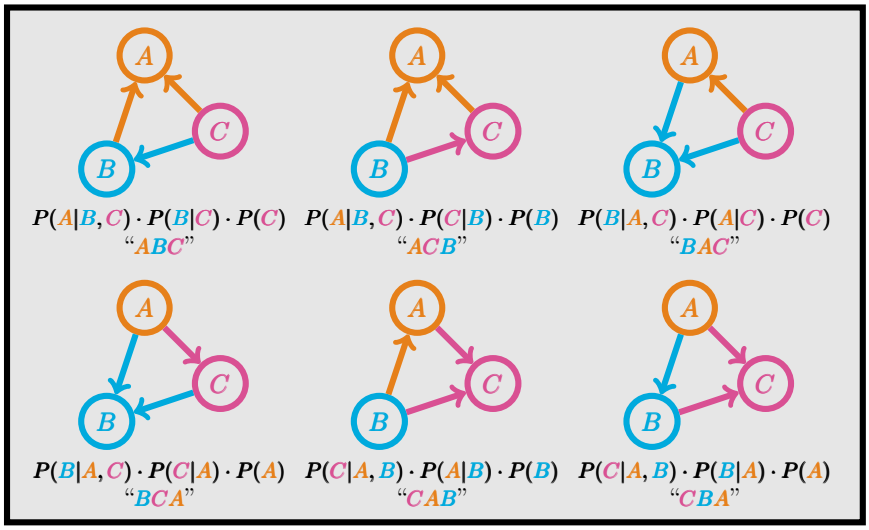

Figure 1.3: Chain rule decomposition viewed as DAGs: applying the chain rule is equivalent to choosing an ordering of the nodes and can be represented as a dense DAG (also called acyclic tournament, see Section 2.1). Assuming there are d random variables, there are $d!$ possible acyclic tournaments/chain rule decompositions. These DAGs are Markov equivalent since they encode the same set of conditional independencies: the empty set.

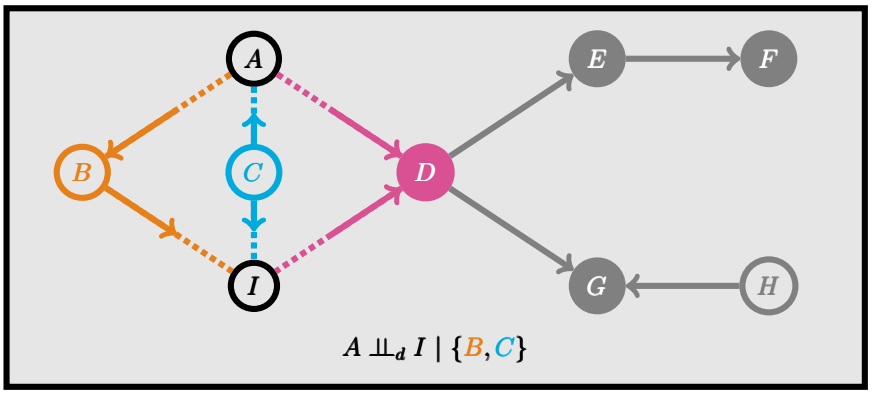

Figure 1.4: Illustration of d-separation: there are three (colored) paths connecting nodes A and I in the depicted DAG and all must be blocked to deduce a set of nodes d-separating A and I: in the orange path $A \cdots\rightarrow B \rightarrow\cdots I$ (respectively the cyan path $A \cdots\leftarrow C \rightarrow\cdots I$), node B (respectively C) is a non-collider hence adding it to the separating set will block that path; in the magenta path $A \cdots\rightarrow D \leftarrow\cdots I$, node D is a collider hence removing it and all its descendants E, F, G (i.e. removing the shaded nodes) from the separating set will block that path. We note that here node H has no influence: the d-separation statements $A \perp\!\!\!\perp_d I \mid \{B, C\}$ and $A \perp\!\!\!\perp_d I \mid \{B, C, H\}$ are both encoded by the DAG.

BNs can deal with various tasks such as prediction, reasoning, anomaly detection, diagnostics, automated insight and decision making under uncertainty. Bayesian networks have been applied for spam filters [Sahami et al., 1998], information retrieval [de Campos et al., 2004], semantic search [Koumenides, 2013], computational biology [Su et al., 2013], document classification [Denoyer and Gallinari, 2004], turbo code [McEliece et al., 1998], among many others.

Once a Bayesian network has been learned, it can be used for *inference*. Denoting V the node set of a BN, inference is the act of computing, given a set of *queried* variables $Q \subset V$ and a (disjoint) set of *observed* variables $O \subset V$, conditional probabilities of the form $P(Q = q|O = o)$. Since learning a BN corresponds to learning the joint probability distribution, one can use a learned BN to infer every possible conditional probability of that form. Unfortunately, inference in Bayesian networks in general is a NP-hard problem both in its exact [Cooper, 1990] and approximate [Dagum and Luby, 1993] form. Classic inference algorithms are the Variable Elimination and the Junction Tree algorithms for exact inference and the Belief Propagation [Pearl, 1982] algorithm for approximate inference (note that Belief Propagation is exact for polytrees). These algorithms are known to be fixed-parameter tractable in the DAG's treewidth, i.e. more *efficient* inference can be achieved for those BNs whose structure have lower treewidth.

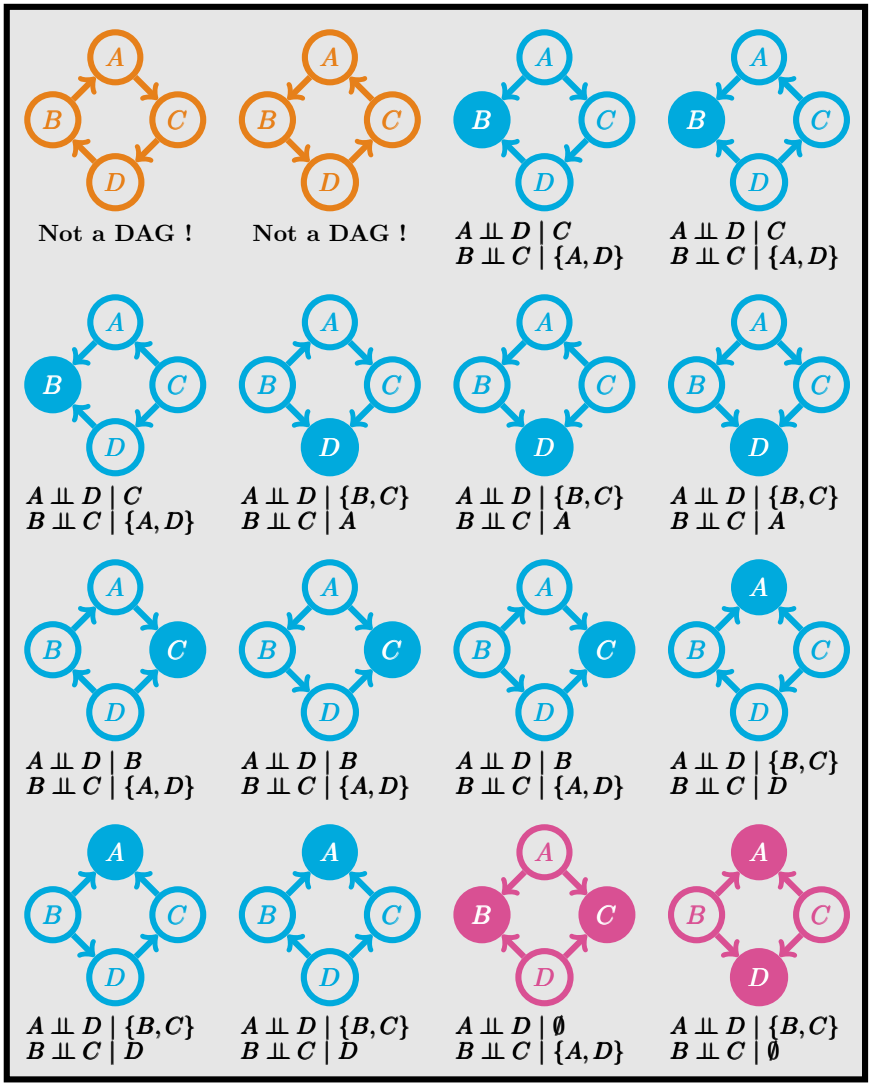

Figure 1.5: BNs cannot represent all dependencies: the set of conditional independencies $\{A \perp\!\!\!\perp D \mid \{B, C\}, B \perp\!\!\!\perp C \mid \{A, D\}\}$ is not encoded by any DAG. Such structure would have no arc between A and D and between B and C. This leaves $2^4 = 16$ possible orientations of the remaining 4 arcs. Orange, cyan, magenta structures respectively contain 0, 1, 2 collider(s) (colliders are the shaded nodes).

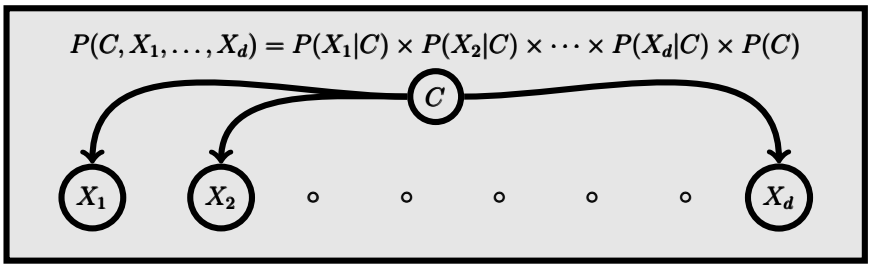

Figure 1.6: The naive Bayes' DAG: naive Bayes is a probabilistic classification model which is based on the *naive* assumption that features $(X_i)_{i\in[\![1,d]\!]}$ describing an instance are conditionally mutually independent given the instance's class C. It is a special case of Bayesian network where the DAG encodes the following set of conditional independencies: $\left\{X_i \perp\!\!\!\perp \{X_j\}_{j\neq i} \mid C\right\}_{i\in[\![1,d]\!]}$. Since every probability distribution factor $P(X_i|C)$ is conditioned on a single random variable (the class), the number of parameters describing the model grows linearly rather than exponentially in the number of features d, making naive Bayes very scalable.

1.2 Learning Bayesian networks

The process of learning a Bayesian network is twofold: first, one searches for a DAG structure fitting best the learning data, which is referred to as *structure learning*; second, one aims to estimate the parameters for every conditional probability distribution fitting best the learning data and the learned DAG, which is referred to as *parameter learning*. This twofold process is summarized by the following factorization of the Bayesian network's posterior probability given some data:

$$\begin{aligned} \underset{\mathcal{B}}{\text{Maximize}}\ P(\mathcal{B}|\mathcal{D}) &= \overbrace{P(\mathcal{S},\Theta|\mathcal{D})}^{\text{full learning}} \\ &= \underbrace{P(\Theta|\mathcal{D},\mathcal{S})}_{\text{parameter learning}} \times \underbrace{P(\mathcal{S}|\mathcal{D})}_{\text{structure learning}}, \end{aligned} \tag{1.3}$$

where $\mathcal{D}$ represents the learning data and $\mathcal{B} = (\mathcal{S}, \Theta)$ is the Bayesian network composed of its DAG structure $\mathcal{S}$ and its parameters Θ.

Learning parameters is a well-understood counting problem which can be solved adopting both the frequentist (maximum likelihood computation) or the Bayesian point of view (maximum a-posteriori computation); in the presence of missing data, the Expectation-Maximization [Dempster et al., 1977] algorithm can be used. Learning the structure of a BN on the other hand is a difficult task: it was proved to be NP-hard even for networks constrained to have in-degree at most 2 [Chickering, 1995] and for polytrees (singly-connected DAGs) [Dasgupta, 1999]. In the current literature, two predominant

paradigms exist for the *Bayesian network structure learning* (BNSL) problem, namely: *constraint-based* and *score-based* methods (combining these paradigms is possible, in which case the method is referred to as *hybrid*). We give a short summary of these paradigms.

1.2.1 Constraint-based methods for structure learning

Constraint-based methods attempt to learn the structure of a BN through a set of statistical tests performed on the learning data and referred to as *constraints*. They originate from the so-called *Inductive Causation* (IC) algorithm developed by Pearl [1989] which then led to the development of the now famous PC algorithm [Spirtes et al., 2000] named after its two first authors: Peter and Clark. These methods rely on the faithfulness assumption, which in conjunction with the Markov property entails there is equivalence between d-separation and conditional independence. As a consequence, statistical tests can be used to assess conditional independencies among random variables and the structure of the DAG is then deduced. Commonly used statistical tests include linear correlation, mutual information, G-tests, Pearson [1900]'s χ^2 test, Fisher [1992]'s exact test as well as Barnard [1945]'s and Boschloo [1970]'s tests.

Major advances in constraint-based structure learning have usually taken the form of extensions or variants of the original IC algorithm: the Fast Causal Inference [Spirtes et al., 1995] algorithm is designed to handle latent variables; the Grow-Shrink [Margaritis, 2003], Incremental Association [Tsamardinos and Aliferis, 2003] and Min-Max [Tsamardinos et al., 2003] Markov Blanket algorithms all estimate the so-called *Markov blanket* of each node (that is, the minimal set of nodes d-separating that node from all other nodes that are not in that set), which is analogous to selecting strongly relevant features and is reported to yield improved scalability while achieving higher structural accuracy in practice compared to the regular PC algorithm [Pellet and Elisseeff, 2008]; order independence was investigated [Colombo and Maathuis, 2014] by detailing every possible occurrence of order dependence in the IC algorithm due to violations of the faithfulness assumption and by proposing small modifications of the original code to remove order dependence at each stage of the algorithm; parallelization was studied [Scutari, 2017] as a mean to get rid of the backtracking optimization commonly used in IC-based algorithms, thus increasing stability while providing significant speedup on parallel architectures; recently, the Dual PC [Giudice et al., 2022] algorithm changed the traditional ordering by which conditional independence tests are performed (low to high cardinality): the algorithm processes conditioning sets starting first with sets of minimal and maximal cardinality and finishing with average-sized conditioning sets. Many

constraint-based algorithms have been implemented in the bnlearn [Scutari, 2010] package.

1.2.2 Score-based methods for structure learning

Score-based methods try to find a DAG which maximises a criterion given some learning data. They rely on two components:

1. A search scheme to efficiently select the different DAG candidates, since high-dimensionality of the full DAG search space makes exhaustive search generally not feasible. Different search schemes may return exact or approximate solutions, either in the full DAG space or in smaller spaces such as the space of Markov equivalence classes.

2. A scoring function to evaluate how well the candidate structures fit the learning data, with the property of being decomposable, that is the *global* score of a BN can be decomposed into a sum of *local* scores over elementary substructures (typically, node-parent set pairs). Scores are traditionally selected among Bayesian scoring functions (e.g. K2 [Cooper and Herskovits, 1992], BD/BDe/BDeu [Heckerman et al., 1995], BDs [Scutari, 2016]...) or among information-theoretic scoring functions (e.g. AIC [Akaike, 1974], BIC [Schwarz, 1978], MIT [de Campos, 2006], NML [Roos et al., 2008]...). The former score category computes the posterior distribution of the BN given a prior on the model's parameters, whereas the latter score category is akin to evaluating the achievable compression over the data with an optimal code induced by the DAG structure of the BN.

Generally speaking, score-based structure learning consists in first scoring local substructures, then combining these local substructures into a global structure constrained to be acyclic and whose global score is optimal with respect to the scoring function: assuming the local scores were precomputed, this second and most difficult part (illustrated in Figure 1.7) can be formalized as follows:

Definition 1. *Let V the node set of a BN and S a decomposable scoring function. For any node-parent set pair (I, j) where $j \in V$ and $I \subset V \setminus \{j\}$, denote $S(I, j) \in \mathbb{R}$ the local score of the substructure $I \to j = \{i \to j : i \in I\}$. Also denote $\mathcal{S}_j$ the set of available local scores for which $j \in V$ is the child node. Given a set of local scores $\mathcal{S} = \bigcup_{j \in V} \mathcal{S}_j$ such that $\forall j \in V$, $\mathcal{S}_j \neq \emptyset$, the* ***score-based Bayesian network structure learning***

(score-based BNSL) problem *is the following optimization problem:*

$$\begin{aligned} &\underset{\bigotimes_{j\in V}\{I_j:\, S(I_j,j)\in\mathcal{S}_j\}}{\textit{Maximize}} \overbrace{\sum_{j\in V} S(I_j,j)}^{\textit{global score}} \\ &\quad \textit{s.t.} \bigcup_{j\in V} (I_j \to j) \;\; \textit{is a DAG.} \end{aligned} \tag{1.4}$$

This optimization problem is non-convex in nature owing to the acyclicity property of DAGs. This non-convexity does not come as a surprise: learning the structure of a BN is NP-hard and so is non-convex optimization in general.

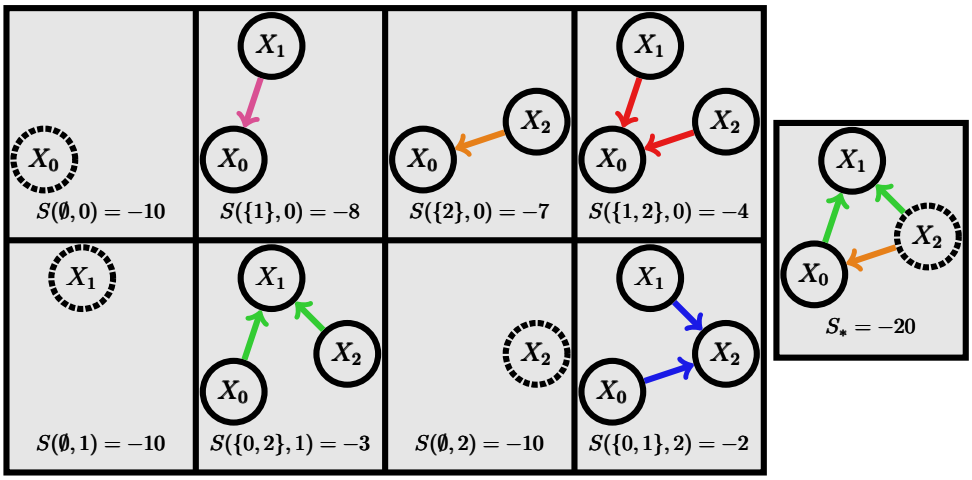

Figure 1.7: Illustration of scored-based structure learning from local scores: for every node, one must select a substructure with that node as a child, in such a way that combining these selected substructures yields a DAG whose sum of local scores is maximized. The list of precomputed local scores is represented on the left (it may not include all possible node-parent set pairs); the optimal DAG solution built from the set of available local scores is represented on the right along with its optimal global score, denoted S_*.

Extensive efforts have been made regarding score-based structure learning and many innovative methods have been developed. On the approximate side, greedy methods such as the K2 [Cooper and Herskovits, 1992], the GES [Meek, 1997] and the ubiquitous Hill-climbing heuristics are local search methods that explore efficiently the space of DAGs (or the space of Markov equivalent classes for GES). On the exact side, dynamic programming had been the go-to approach in the early 2000's following pioneering and independent works from Singh and Moore [2004], Ott et al. [2004] and Koivisto and Sood [2004] that led to further important contributions [Parviainen and Koivisto, 2010;

Silander and Myllymäki, 2006], until integer programming was successfully applied to score-based structure learning and became the new dominant exact approach for that problem. Integer programming methods express the problem of finding an optimal DAG structure as an optimization problem involving a linear objective function and a set of linear constraints on integral variables. The first efforts in this direction were made by Hemmecke et al. [2012] with the usage of binary vectors giving an algebraic representation of BNs (the so-called *characteristic imsets*), as well as Jaakkola et al. [2010] and Cussens [2011] who independently used the so-called *family-variable* representation. From a practical standpoint, while dynamic programming methods typically scale up to a few dozen of nodes, integer programming methods on the other hand can scale up to a few hundred nodes, albeit with significant structural restrictions (e.g. bounded in-degree). GOBNILP is an integer programming-based implementation of Cussens [2011]' method that has been under development for more than a decade and still achieves to this day state-of-the-art performance for exact score-based structure learning. Score-based methods addressing specifically the structure learning problem of continuous Bayesian networks exist as well: notably, the NOTEARS [Zheng et al., 2018] algorithm is based on continuous optimization and achieves state-of-the-art performance assuming linear dependencies among continuous random variables (the so-called *structural equation models* setting, discussed in more details in Section 3.2); it was later extended to handle non-linear dependencies [Zheng et al., 2020].

1.3 Outline

The present work fits into the scored-based structure learning paradigm and more precisely, aims at developing and analysing new heuristics that would improve the *scalability* of structure learning, that is the ability to learn larger networks (typically measured by the number of nodes). A common theme found in every publication produced in the context of this research is the integration of a classic combinatorial problem - the so-called *maximum acyclic subgraph problem* - as a key ingredient to achieve the desired increase in scalability. Beyond the present introductory chapter, the rest of this thesis is organized with the intent to instil the necessary mathematical tools required to properly understand the methods, techniques and design choices that led to the development of scalable (score-based) structure learning heuristics:

- Chapter 2 gives the necessary mathematical background to understand important notions such as (directed) acyclicity, maximum acyclic subgraph and convex optimization. These are core concepts in our publications.

- Chapter 3 summarizes the main contributions of this research by briefly introducing two new heuristics that we developed in the context of this research: **BNSL2MAS** (Article I) and **ProxiMAS** (Articles II, III).
- An appendix is provided in order to ensure that the reader fully grasps various concepts explored in the thesis and the published articles.

This thesis was written with the purpose of being a self-contained read: it complements the regular body of text with mathematical definitions, proofs and foundations, algorithms' pseudo-codes, as well as illustrating experiments and explanatory figures.

Chapter 2

Background

Score-based structure learning is a difficult problem in several aspects. The trained mathematician will understand this statement in terms of the NP-hardness property from complexity theory while the seasoned computer scientist will implement various algorithms and notice hard limitations both in scalability and in the ability to reach global extrema. An overlooked aspect of the inherent difficulty of the problem is its intersection between two orthogonal mathematical worlds: *graph theory* and *optimization*.

This background section aims first at providing the reader a tour encompassing key notions lying at the intersection of these worlds, what one could call *combinatorial optimization*. Specifically, Section 2.1 uses notions from graph theory to characterize acyclicity in directed graphs; Section 2.2 recalls how three combinatorial optimization problems involving acyclicity - maximum acyclic subgraph, minimum feedback arc set and maximum acyclic tournament problems - are interconnected; Section 2.3 follows and adds more context on the theoretical aspects behind the maximum acyclic subgraph and the minimum feedback arc set problems; Sections 2.4, 2.5 provide basic knowledge on integer programming and explain how maximum acyclic subgraph problems can be solved exactly based on that framework: this knowledge is at the core of Article I.

In a second time, this background section deviates from combinatorial optimization and explores in Section 2.6 greedy heuristics designed to approximately solve maximum acyclic subgraph problems, then sums up important notions from convex optimization in Sections 2.7, 2.8: those are the main ingredients for Articles II, III.

2.1 Acyclicity property for directed graphs

This section is devoted to the characterization of acyclicity, a key graph property at the core of the definition of Bayesian networks. The combinatorial nature of acyclicity is a primary reason behind the complexity (both theoretical and practical) in learning the structure of these probabilistic graphical models; understanding acyclicity is therefore an important step in order to develop new efficient learning strategies.

We begin with definitions for the graphical objects will we use throughout the thesis:

Definition 2. *A **graph** is a mathematical object representing connections (**edges**) between discrete entities (**nodes**). Formally, a **graph** with d nodes is a pair $\mathcal{G} = (V, E)$ where $V = [\![0, d-1]\!]$ is the **node set** and $E \subset V^2$ is the **edge set**. Given two nodes i, j:*

- *An **edge** $\{i, j\}$ is an undirected connection between nodes i and j $\big($i.e. $\{i, j\}$ and $\{j, i\}$ represent the same edge$\big)$.*
- *An **arc** (i, j) is a directed connection from the **parent** node i to the **child** node j $\big($i.e. (i, j) and (j, i) represent opposite arcs$\big)$.*

Convention 1. ***We will call graph instead of undirected graph any graph whose connections are exclusively edges. We will call digraph instead of directed graph any graph whose connections are exclusively arcs. In digraphs, the edge set will be called arc set.***

Definition 3. *The **skeleton** of a digraph is the graph constructed by adding $\{i, j\}$ to the graph's edge set if (i, j) or (j, i) are in the digraph's arc set.*

Definition 4. *Given an edge set (respectively arc set) E, an associated **score function** is a mapping $s : E \to \mathbb{R}$ encoding edge weights (respectively arc weights).*

Definition 5. *Graphs and digraphs can be expressed in matrix form as well:*

- *A graph (respectively digraph) with d nodes is equivalently represented by the so-called **adjacency matrix** $A \in \{0, 1\}^{d \times d}$, where $A[i, j] = 1$ if $\{i, j\}$ (respectively (i, j)) is in the graph's edge set (respectively the digraph's arc set) and $A[i, j] = 0$ otherwise.*
- *A weighted graph (respectively weighted digraph) with d nodes and an associated score function s is equivalently represented by the so-called **weighted adjacency matrix** $W \in \mathbb{R}^{d \times d}$, where $W[i, j] = s(i, j)$ if $\{i, j\}$ (respectively (i, j)) is in the graph's edge set (respectively the digraph's arc set) and $W[i, j] = 0$ otherwise.*

Definition 6. *Two edges are said to be* ***adjacent*** *if they share a single end-point; a* ***path*** *is a sequence of consecutive adjacent edges. Analogously in the directed case: two arcs are said to be* ***adjacent*** *if an arc's child node is the other arc's parent node; a* ***directed path*** *is a sequence of consecutive adjacent arcs. A* ***loop*** *(respectively* ***cycle****) is a path (respectively directed path) whose end-points are the same node. A loop (respectively cycle) consisting of l edges (respectively arcs) is said to have* ***length*** *l and is called l-****loop*** *(respectively l-****cycle****).*

Convention 2. ***In all that follows and unless stated otherwise, we will always assume digraphs to be simple, that is without any 1-cycle (equivalently, whose corresponding adjacency matrices have a diagonal full of zeros).***

Definition 7. *A* ***directed acyclic graph*** *or* ***DAG*** *is a digraph containing no cycle.*

Now that we have defined DAGs, we are ready to characterize them (Lemmas 2, 3). Intuitively, nodes in a DAG are *ordered*: they have a *top* and a *bottom* (see Lemma 1):

Definition 8. *In a digraph, a* ***source*** *(respectively* ***sink****) is a node without any parent (respectively child).*

Lemma 1. *Let $\mathcal{G} = (V, E)$ a DAG. Then $\mathcal{G}$ has a source and a sink.*

Proof. We conduct the proof for the source and note that the exact same line of reasoning can be used for the sink. Clearly, if $\mathcal{G}$ contains no arc the claim trivially follows i.e. we can assume without loss of generality that there is an arc $(v_1, v_0) \in E$ (where v_0 and v_1 are distinct). By way of contradiction, suppose $\mathcal{G}$ does not have a source. A digraph with $d = 2$ nodes and no source is a 2-cycle, so we can assume without loss of generality that $V = [\![0, d-1]\!]$ where $d > 2$. Since v_1 is not a source, $\mathcal{G}$ is acyclic and $(v_1, v_0) \in E$, there must exist $v_2 \in V \setminus \{v_0, v_1\}$ such that $(v_2, v_1) \in E$, implying $v_2 \to v_1 \to v_0 \subset \mathcal{G}$. In fact, since no node is a source and $\mathcal{G}$ is acyclic we can use the same argument recursively and obtain that for all $i \in [\![2, d-1]\!]$:

$$v_{i-1} \to \cdots \to v_0 \subset \mathcal{G} \implies \exists v_i \in V \setminus \{v_0, \ldots, v_{i-1}\} : \; v_i \to \cdots \to v_0 \subset \mathcal{G}.$$

In particular, the path $v_{d-1} \to \cdots \to v_0$ spans $\mathcal{G}$. Furthermore, v_{d-1} is not a source so there exists $j \in [\![0, d-2]\!]$ such that $(v_j, v_{d-1}) \in E$. But then one would get that the cycle $v_j \to v_{d-1} \to \cdots \to v_j$ is contained in $\mathcal{G}$, a contradiction. □

Definition 9. *Let $\mathcal{G} = (V, E)$ a digraph. We say that $\mathcal{G}$ has a* ***topological order*** *if there is a permutation operator $\pi : V \to V$ such that:*

$$\forall (i, j) \in E, \; \pi(i) < \pi(j). \tag{2.1}$$

Remark 1. *We note that Definition 9 can be equivalently restated with a permutation operator $\pi' : V \to V$ such that:*

$$\forall (i,j) \in E,\ \pi'(i) > \pi'(j). \tag{2.2}$$

Indeed, assume (without loss of generality) that $V = [\![0, d-1]\!]$ and consider the permutation operator reversing nodes order, i.e. $\overleftarrow{\pi} : i \in [\![0, d-1]\!] \mapsto d-1-i$. Then $\pi' \coloneqq \overleftarrow{\pi} \circ \pi$ is a permutation of V if and only if π is a permutation of V. Moreover, for all $(i,j) \in E$:

$$\begin{aligned} \pi(i) < \pi(j) &\iff d-1-\pi(i) > d-1-\pi(j) \\ &\iff \overleftarrow{\pi} \circ \pi(i) > \overleftarrow{\pi} \circ \pi(j) \\ &\iff \pi'(i) > \pi'(j). \end{aligned}$$

Lemma 2. *Let $\mathcal{G} = (V, E)$ a digraph. Then $\mathcal{G}$ is a DAG if and only if $\mathcal{G}$ has a topological order.*

Proof. In what follows we will assume without loss of generality that $\mathcal{G}$ has d nodes, i.e. $V = [\![0, d-1]\!]$.

$\Rightarrow$. We construct the topological order recursively. First, $\mathcal{G}$ is a DAG hence Lemma 1 guarantees it has a source, let us call it $v_0 \in V$. Now, consider the digraph $\mathcal{G} \setminus \{v_0\}$ constructed by removing from $\mathcal{G}$ its source v_0 and every arc with v_0 as a parent. Since removing arcs cannot introduce any cycle, necessarily $\mathcal{G} \setminus \{v_0\}$ is itself a DAG and due to Lemma 1 it must have a source, let us call it $v_1 \in V \setminus \{v_0\}$. Let us then use the notations $V_i \coloneqq V \setminus \{v_0, \dots, v_{i-1}\}$ and $G_i \coloneqq G \setminus \{v_0, \dots, v_{i-1}\}$, with the convention $V_0 \coloneqq V$ and $G_0 \coloneqq G$. Applying the previous argument recursively, one gets that for all $i \in [\![1, d-1]\!]$:

$$G_{i-1} \text{ is a DAG with a source } v_{i-1} \in V_{i-1} \implies G_i \text{ is a DAG with a source } v_i \in V_i.$$

Now, consider the operator $\pi : i \in [\![0, d-1]\!] \mapsto v_i$. Clearly by construction, π is a permutation of V and so is its inverse π^{-1}; additionally, notice $(v_i, v_j) \in E \implies i < j$ (otherwise, v_j would not be a source at step j). We conclude that $\mathcal{G}$ has a topological order, since:

$$\forall (v_i, v_j) \in E,\ \pi^{-1}(v_i) = i < j = \pi^{-1}(v_j).$$

$\Leftarrow$. We proceed by way of contradiction. Suppose $\mathcal{G}$ has a topological order and there is a cycle $v_0 \to \cdots \to v_{l-1} \to v_0$ contained in $\mathcal{G}$, where $l \geq 2$ and $(v_0, \dots, v_{l-1}) \in V^l$. But then, since $\mathcal{G}$ has a topological order we know there is a permutation operator

$\pi : V \to V$ that causes a contradiction:

$$\pi(v_0) < \cdots < \pi(v_{l-1}) < \pi(v_0).$$

□

Lemma 3. *Let $\mathcal{G} = (V, E)$ a digraph with d nodes and A the adjacency matrix of $\mathcal{G}$. Then $\mathcal{G}$ is a DAG if and only if there is a permutation operator $\pi : V \to V$ and a strictly upper-triangular matrix T such that $A = \big(T[\pi(i), \pi(j)]\big)_{(i,j)\in[\![0,d-1]\!]^2}$.*

Proof. Due to Lemma 2, we can replace the statement "$\mathcal{G}$ is a DAG" by "$\mathcal{G}$ has a topological order".

⇒. Suppose $\mathcal{G}$ has a topological order, i.e. there is a permutation operator $\pi : V \to V$ such that for all $(i, j) \in E,\ \pi(i) < \pi(j)$. As a permutation, π is invertible and we can define the matrix $T \coloneqq \big(A[\pi^{-1}(i), \pi^{-1}(j)]\big)_{(i,j)\in[\![0,d-1]\!]^2}$. Clearly:

$$\begin{aligned}\forall (i,j) \in [\![0, d-1]\!]^2,\ A[i,j] &= A\big[\pi^{-1}\big(\pi(i)\big), \pi^{-1}\big(\pi(j)\big)\big] \\ &= T[\pi(i), \pi(j)].\end{aligned}$$

Besides, notice the topological order property satisfied by π with respect to $\mathcal{G}$ can be equivalently rewritten:

$$\forall (i,j) \in [\![0, d-1]\!]^2,\ \big[\pi(i) \geq \pi(j) \implies (i,j) \notin E\big].$$

It follows that for all $(i, j) \in [\![0, d-1]\!]^2$:

$$\begin{aligned} i \geq j &\iff \pi\big(\pi^{-1}(i)\big) \geq \pi\big(\pi^{-1}(j)\big) \\ &\implies \big(\pi^{-1}(i), \pi^{-1}(j)\big) \notin E \\ &\iff A[\pi^{-1}(i), \pi^{-1}(j)] = 0 \\ &\iff T[i,j] = 0, \end{aligned}$$

proving T is strictly upper-triangular as desired.

⇐. Conversely, suppose there is a permutation operator $\pi : V \to V$ and a strictly upper-triangular matrix T such that $A = \big(T[\pi(i), \pi(j)]\big)_{(i,j)\in[\![0,d-1]\!]^2}$. Then π clearly satisfies the topological order property with respect to $\mathcal{G}$, since we have that for

all $(i, j) \in [\![0, d-1]\!]^2$:

$$\begin{aligned}(i,j) \in E &\iff A[i,j] = 1 \\ &\iff T[\pi(i), \pi(j)] = 1 \\ &\implies \pi(i) < \pi(j).\end{aligned}$$

□

Remark 2. *We note that due to Remark 1, Lemma 3 can be equivalently restated with a strictly lower-triangular matrix T.*

Of particular interest are those DAGs that are *dense*: Lemma 4 states that their acyclicity property can be characterized looking at 2 and 3-cycles only:

Definition 10. *A **tournament** is a digraph with a complete skeleton and no 2-cycle, i.e. a digraph obtained by assigning a single direction to every edge in a complete graph.*

Lemma 4. *Let $\mathcal{G} = (V, E)$ a digraph with d nodes. Then $\mathcal{G}$ is an acyclic tournament if and only if the two following assertions hold:*

1. *$\mathcal{G}$ has a complete skeleton and no 2-cycle:*

$$\forall (i,j) \in [\![0, d-1]\!]^2 : i, j \text{ distinct}, \ \big[(i,j) \in E \iff (j,i) \notin E\big]. \tag{2.3}$$

2. *$\mathcal{G}$ has no 3-cycle:*

$$\forall (i,j,k) \in [\![0, d-1]\!]^3 : i, j, k \text{ distinct}, \\ \big[(i,j) \in E \text{ and } (j,k) \in E \implies (k,i) \notin E\big]. \tag{2.4}$$

Proof.

$\Rightarrow$. Trivial since 1. is by definition of tournament and 2. follows immediately from being acyclic.

$\Leftarrow$. By hypothesis we know that $\mathcal{G}$ contains no 2 or 3-cycle, i.e. all is left to prove is that $\mathcal{G}$ does not contain any l-cycle where $l > 3$. By way of contradiction, suppose there is an l-cycle $v_0 \to \cdots \to v_{l-1} \to v_0$ contained in $\mathcal{G}$, where $l > 3$ and $(v_0, \ldots, v_{l-1}) \in V^l = [\![0, d-1]\!]^l$. Notice $v_0 \to v_1 \to v_2 \subset \mathcal{G}$ such that $(v_2, v_0) \notin E$ (otherwise, we would have the 3-cycle $v_0 \to v_1 \to v_2 \to v_0$ contained in $\mathcal{G}$). This ensures we have $(v_0, v_2) \in E$ since by hypothesis the skeleton of $\mathcal{G}$ is complete.

Then, $(v_2, v_3) \in E$ entails $v_0 \to v_2 \to v_3 \subset \mathcal{G}$. Applying this argument recursively yields for all $i \in [\![2, l-2]\!]$:

$$\begin{aligned} v_0 \to v_{i-1} \to v_i \subset \mathcal{G} &\implies (v_i, v_0) \notin E \\ &\implies (v_0, v_i) \in E \\ &\implies v_0 \to v_i \to v_{i+1} \subset \mathcal{G}. \end{aligned}$$

In particular we deduce that $v_0 \to v_{l-2} \to v_{l-1} \subset \mathcal{G}$ holds, but then the 3-cycle $v_0 \to v_{l-2} \to v_{l-1} \to v_0$ is contained in $\mathcal{G}$ which is a contradiction. An illustration of this line of reasoning is provided in Figure 2.1.

□

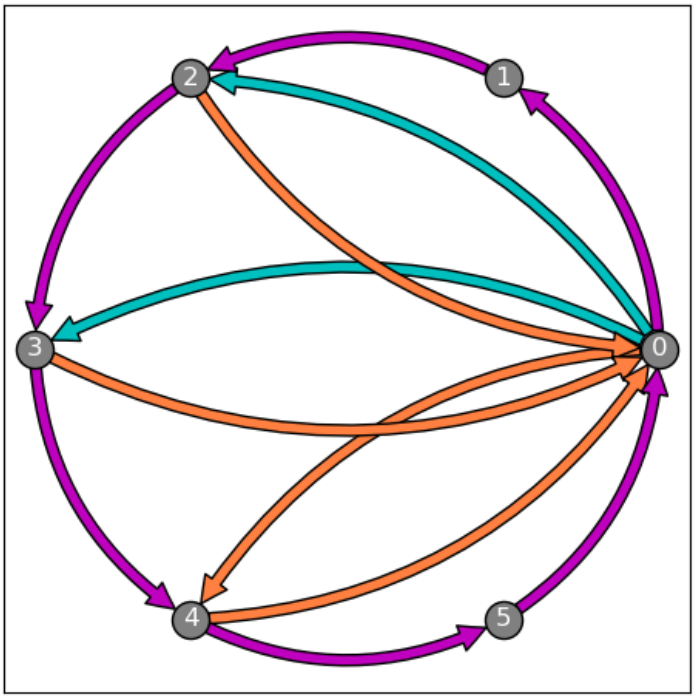

Figure 2.1: Illustration of Lemma 4: a tournament with no 3-cycle is necessarily acyclic. If (by way of contradiction) such type of tournament were to contain a cycle (represented in magenta), then one could iteratively construct two sequences of arcs, one included in and one disjoint from the tournament (represented in cyan and orange, respectively). Ultimately, the construction ends up in a contradiction: the two arcs in the orange 2-cycle would both induce a 3-cycle in the tournament, hence they cannot belong to the tournament, a contradiction since the skeleton of a tournament is complete.

2.2 Maximum acyclic subgraph and minimum feedback arc set

In this section we recall basic mathematical facts regarding two fundamental combinatorial problems, the so-called *maximum acyclic subgraph* (MAS) and *minimum feedback arc set* (FAS) problems, both belonging to the famous Karp [1972]'s 21 NP-hard problems. These two combinatorial problems are at the heart of this thesis, hence we take the time to precisely define them, see how they are related and even re-establish a connection with another combinatorial problem - the so-called *maximum acyclic tournament* (MAT) problem - giving more insight on the acyclicity property and its potential scalability. In this thesis, we are interested in the weighted versions of the aforementioned combinatorial problems.

We start off by defining the weighted MAS and FAS problems:

Definition 11. *Given a weighted digraph $\mathcal{G} = (V, E, s)$ as input, the **weighted maximum acyclic subgraph (weighted MAS) problem** is to find a weighted subgraph (V, E_*, s) of $\mathcal{G}$ whose sum of arc weights is maximized under the constraint that (V, E_*) is a DAG. We will say:*

- *A weighted digraph (V, E', s) is MAS($\mathcal{G}$)-admissible if $E' \subset E$, (V, E') is a DAG and $s_{|E'} > 0$.*
- *A weighted digraph (V, E_*, s) is MAS($\mathcal{G}$)-optimal if:*

$$E_* = \underset{E'}{argmax} \sum_{(i,j)\in E'} s(i,j) \quad s.t.\ (V, E', s) \text{ is MAS}(\mathcal{G})\text{-admissible.} \tag{2.5}$$

Remark 3. *We note that an optimal weighted MAS solution cannot contain an arc with a strictly negative weight, otherwise a strictly better DAG solution would be found by simply removing that arc (removing arcs cannot introduce any cycle). In Definition 11, one could have dropped the constraint $s_{|E'} > 0$, in which case the implicit constraint $s_{|E'} \geq 0$ would hold instead, resulting in an equally optimal solution but possibly less sparse. In machine learning, one usually seeks sparse solutions, hence our choice to explicitly enforce the constraint $s_{|E'} > 0$.*

Definition 12. *Given a weighted digraph $\mathcal{G} = (V, E, s)$ as input, the **weighted (minimum) feedback arc set (weighted FAS) problem** is to find a weighted subgraph (V, E_*, s) of $\mathcal{G}$ whose sum of arc weights is minimized under the constraint that $(V, E \backslash E_*)$ is a DAG. We will say:*

- *A weighted digraph (V, E', s) is FAS($\mathcal{G}$)-admissible if $E' \subset E$, $(V, E \setminus E')$ is a DAG and $s_{|E \setminus E'} > 0$.*
- *A weighted digraph (V, E_*, s) is FAS($\mathcal{G}$)-optimal if:*

$$E_* = \underset{E'}{argmin} \sum_{(i,j) \in E'} s(i,j) \quad s.t.\ (V, E', s) \text{ is FAS}(\mathcal{G})\text{-admissible.} \tag{2.6}$$

From an optimization standpoint, the weighted MAS and FAS problems are dual in the sense that a valid (respectively optimal) solution for one yields a valid (respectively optimal) solution for the other, as showed in Lemma 5:

Lemma 5. *Let $\mathcal{G} = (V, E, s)$ a weighted digraph. The following holds:*

1. *If (V, E_*, s) is MAS($\mathcal{G}$)-optimal, then $(V, E \setminus E_*, s)$ is FAS($\mathcal{G}$)-optimal.*
2. *If (V, E_*, s) is FAS($\mathcal{G}$)-optimal, then $(V, E \setminus E_*, s)$ is MAS($\mathcal{G}$)-optimal.*

Proof. Notice that given any $E' \subset E$, one also has $E \setminus E' \subset E$; in particular this allows us to write $E' = E \setminus (E \setminus E')$ and it clearly follows that:

$$\begin{cases} (V, E', s) \text{ is MAS}(\mathcal{G})\text{-admissible} & \implies (V, E \setminus E', s) \text{ is FAS}(\mathcal{G})\text{-admissible} \\ (V, E', s) \text{ is FAS}(\mathcal{G})\text{-admissible} & \implies (V, E \setminus E', s) \text{ is MAS}(\mathcal{G})\text{-admissible.} \end{cases}$$

1. Suppose by way of contradiction that (V, E_*, s) is MAS($\mathcal{G}$)-optimal and $(V, E \setminus E_*, s)$ is not FAS($\mathcal{G}$)-optimal. With (V, E_*, s) MAS($\mathcal{G}$)-admissible, $(V, E \setminus E_*, s)$ must be FAS($\mathcal{G}$)-admissible; then, $(V, E \setminus E_*, s)$ is not FAS($\mathcal{G}$)-optimal means there exists (V, E'_*, s) FAS($\mathcal{G}$)-admissible with strictly smaller sum of arc weights compared to $(V, E \setminus E_*, s)$, such that:

$$\begin{aligned} \sum_{(i,j) \in E} s(i,j) - \sum_{(i,j) \in E \setminus E'_*} s(i,j) &= \sum_{(i,j) \in E \setminus (E \setminus E'_*)} s(i,j) \\ &= \sum_{(i,j) \in E'_*} s(i,j) \\ &< \sum_{(i,j) \in E \setminus E_*} s(i,j) \\ &= \sum_{(i,j) \in E} s(i,j) - \sum_{(i,j) \in E_*} s(i,j). \end{aligned}$$

After simplification, we deduce:

$$\sum_{(i,j) \in E \setminus E'_*} s(i,j) > \sum_{(i,j) \in E_*} s(i,j).$$

Now, with (V, E'_*, s) FAS($\mathcal{G}$)-admissible, $(V, E \setminus E'_*, s)$ must be MAS($\mathcal{G}$)-admissible. But then $(V, E \setminus E'_*, s)$ is MAS($\mathcal{G}$)-admissible with strictly larger sum of arc weights compared to (V, E_*, s), contradicting the fact that (V, E_*, s) is MAS($\mathcal{G}$)-optimal.

2. The proof for this statement follows the exact same reasoning as in 1. (simply swap in the proof "MAS"/"FAS", "smaller"/"larger" and "<"/">").

□

We now define the weighted MAT problem:

Definition 13. *Given a weighted digraph $\mathcal{G} = (V, E, s)$ as input, the **weighted maximum acyclic tournament (weighted MAT) problem** is to find a weighted subgraph (V, E_*, s) of $\mathcal{G}$ whose sum of arc weights is maximized under the constraint that (V, E_*) is an acyclic tournament. We will say:*

- *A weighted digraph (V, E', s) is MAT($\mathcal{G}$)-admissible if $E' \subset E$ and (V, E') is an acyclic tournament.*
- *A weighted digraph (V, E_*, s) is MAT($\mathcal{G}$)-optimal if:*

$$E_* = \underset{E'}{argmax} \sum_{(i,j)\in E'} s(i,j) \quad s.t.\ (V, E', s)\ is\ MAT(\mathcal{G})\text{-}admissible. \tag{2.7}$$

Remark 4. *We note that the weighted MAS and MAT problems differ in two important ways, caused by the fact that tournaments have complete skeletons:*

- *The weighted MAS problem always has an admissible solution, whereas the weighted MAT problem has none if the skeleton of the input weighted digraph is not complete.*
- *Unlike the weighted MAS problem, the weighted MAT problem does not implicitly enforce the constraint $s_{|E'} \geq 0$. If a weighted digraph has a complete skeleton and all its arc weights are non-negative (strictly if the constraint $s_{|E'} > 0$ is enforced in the weighted MAS problem), then the two problems share an identical optimal solution given this weighted digraph as input (see Figure 2.2).*

The rest of the section aims at establishing an important connection between the weighted MAS and MAT problems: Lemma 6 first shows how a DAG with strictly non-negative weights can be extended into (or recovered from) an acyclic tournament with non-negative weights in such a way that the sum of arc weights in the acyclic tournament exceeds the sum of arc weights in the DAG; Lemma 7 finally establishes that solving the weighted MAS problem reduces to solving a related weighted MAT problem:

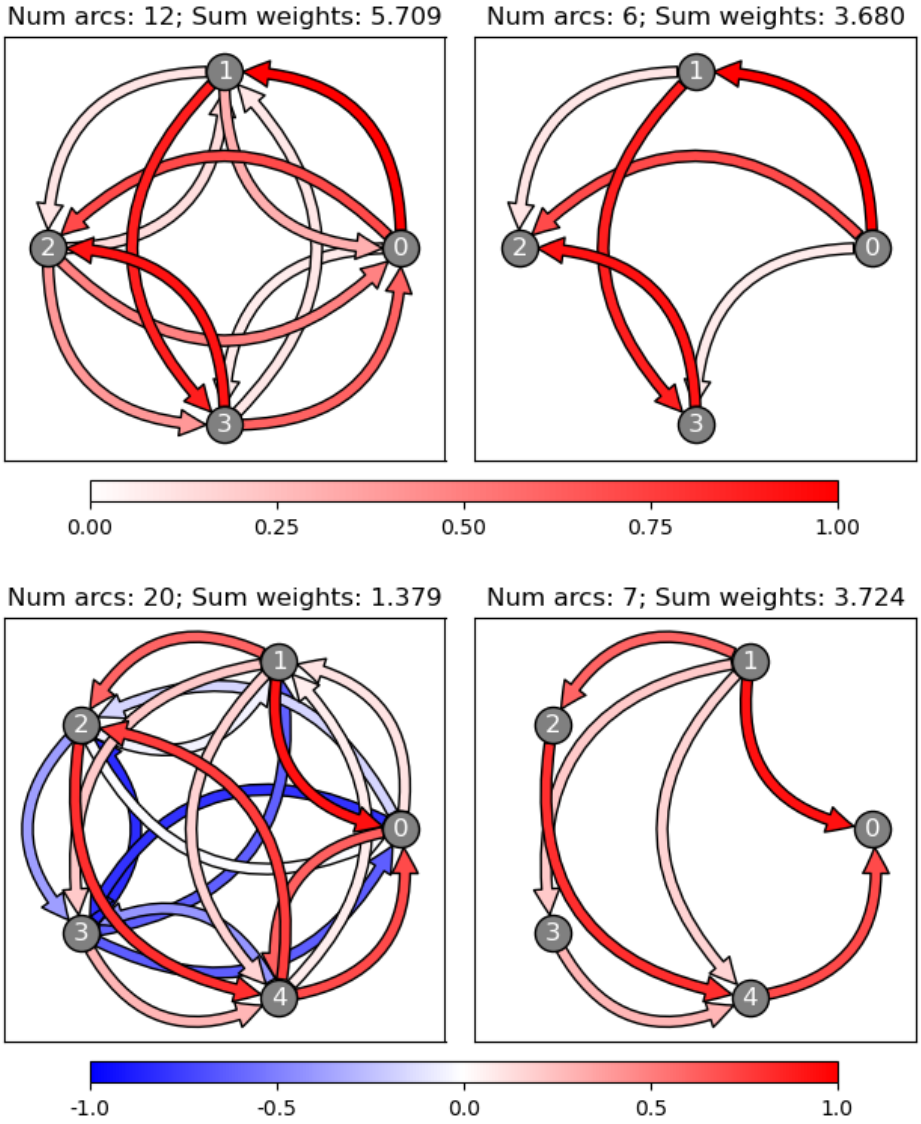

Figure 2.2: Illustration of the weighted MAS problem (left figures: complete weighted digraphs; right figures: corresponding weighted MAS optimal solutions). Given as input a weighted digraph with d nodes having a complete skeleton: if all arc weights are strictly non-negative, a weighted MAS optimal solution must contain $\frac{d(d-1)}{2}$ arcs (it is an acyclic tournament) and its sum of arc weights is strictly smaller than the sum of arc weights in the input (top figures); if some of the arc weights are negative or null, a weighted MAS optimal solution may contain less than $\frac{d(d-1)}{2}$ arcs and its sum of arc weights may be larger than the sum of arc weights in the input (bottom figures), due to the fact that a weighted MAS optimal solution cannot contain an arc with negative or null weight.

Lemma 6. *Let $\mathcal{G} = (V, E, s)$ a weighted digraph containing at least one arc with strictly non-negative weight. Construct the complete weighted digraph $\overline{\mathcal{G}} = (V, V^2, \overline{s})$ as follows:*

$$\forall (i,j) \in V^2,\ \overline{s}(i,j) = \begin{cases} \max\big(0, s(i,j)\big) & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

Then:

1. *For every acyclic tournament $(V, \overline{E}')$ such that $\overline{E}' \subset V^2$ and $\overline{s}_{|\overline{E}'} \neq 0$, one can construct a DAG (V, E') such that $E' \subset E$ and $s_{|E'} > 0$, satisfying:*

$$E' \subset \overline{E}' \quad \text{and} \quad \sum_{(i,j)\in E'} s(i,j) = \sum_{(i,j)\in \overline{E}'} \overline{s}(i,j). \tag{2.9}$$

2. *For every DAG (V, E') such that $E' \subset E$ and $s_{|E'} > 0$, one can construct an acyclic tournament $(V, \overline{E}')$ such that $\overline{E}' \subset V^2$ and $\overline{s}_{|\overline{E}'} \neq 0$, satisfying:*

$$E' \subset \overline{E}' \quad \text{and} \quad \sum_{(i,j)\in E'} s(i,j) \leq \sum_{(i,j)\in \overline{E}'} \overline{s}(i,j). \tag{2.10}$$

Proof. We start with the important remark that by construction, $\overline{s} \geq 0$. Additionally, due to our assumption that $\mathcal{G}$ contains an arc with strictly non-negative weight, we know as well that $\overline{s} \neq 0$ must hold.

1. Let $(V, \overline{E}')$ an acyclic tournament such that $\overline{E}' \subset V^2$ and $\overline{s}_{|\overline{E}'} \neq 0$. Now, consider the set of arcs $E' \coloneqq \big\{(i,j) \in \overline{E}' :\ \overline{s}(i,j) > 0\big\} \subset \overline{E}'$ (we remark that E' is non-empty due to the properties $\overline{s}_{|\overline{E}'} \neq 0$ and $\overline{s} \geq 0$). Since $\overline{s} \geq 0$, by construction both $\overline{s}_{|\overline{E}'\setminus E'} = 0$ and $\overline{s}_{|E'} > 0$ are satisfied. By hypothesis we know as well that $\overline{s}_{|V^2\setminus E} = 0$, which combined with $\overline{s}_{|E'} > 0$ necessarily yields $E' \subset E$. Then, for all $(i,j) \in E' \subset E$:

$$0 < \overline{s}(i,j) = \max\big(0, s(i,j)\big) \implies \overline{s}(i,j) = s(i,j) > 0,$$

   i.e. $\overline{s}_{|E'} = s_{|E'} > 0$ holds; along with $\overline{s}_{|\overline{E}'\setminus E'} = 0$, we deduce:

$$\begin{aligned} \sum_{(i,j)\in\overline{E}'} \overline{s}(i,j) &= \sum_{(i,j)\in\overline{E}'\setminus E'} \overline{s}(i,j) + \sum_{(i,j)\in E'} \overline{s}(i,j) \\ &= 0 + \sum_{(i,j)\in E'} \overline{s}(i,j) \\ &= \sum_{(i,j)\in E'} s(i,j). \end{aligned}$$

Finally, proving that (V, E') is a DAG is trivial: it is a subgraph of the acyclic tournament $(V, \overline{E}')$ and removing arcs cannot introduce any cycle.

2. Let (V, E') a DAG such that $E' \subset E$ and $s_{|E'} > 0$. Without loss of generality, let us assume (V, E') has d nodes. Denoting A' the adjacency matrix of the DAG (V, E'), Lemma 3 ensures there is a permutation operator $\pi : V \to V$ and a strictly upper-triangular matrix T satisfying $A' = \big(T[\pi(i), \pi(j)]\big)_{(i,j)\in[\![0,d-1]\!]^2}$. Now, define the following matrices:

$$\overline{T} \coloneqq \begin{bmatrix} 0 & 1 & \dots & 1 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 1 \\ 0 & \dots & \dots & 0 \end{bmatrix} \quad \text{and} \quad \overline{A}' \coloneqq \big(\overline{T}[\pi(i), \pi(j)]\big)_{(i,j)\in[\![0,d-1]\!]^2}.$$

With $\overline{T}$ strictly upper-triangular, Lemma 3 ensures the digraph $(V, \overline{E}')$ with adjacency matrix $\overline{A}'$ is a DAG (where $\overline{E}' \subset V^2$). It is in fact an acyclic tournament since for all $(i, j) \in [\![0, d-1]\!]^2$ where i, j are distinct:

$$\begin{aligned} \overline{A}'[i,j] = 0 &\iff \overline{T}[\pi(i), \pi(j)] = 0 \\ &\iff \overline{T}[\pi(j), \pi(i)] = 1 \\ &\iff \overline{A}'[j,i] = 1. \end{aligned}$$

Notice as well that for all $(i, j) \in [\![0, d-1]\!]^2$:

$$\begin{aligned} A'[i,j] = 1 &\iff T[\pi(i), \pi(j)] = 1 \\ &\implies \overline{T}[\pi(i), \pi(j)] = 1 \\ &\iff \overline{A}'[i,j] = 1, \end{aligned}$$

i.e. $E' \subset \overline{E}'$. Finally, with $E' \subset E$ and $s_{|E'} > 0$ it is clear by construction of $\overline{s}$ that $\overline{s}_{|E'} = s_{|E'} > 0$ holds (and the latter combined with $E' \subset \overline{E}'$ also ensures that $\overline{s}_{|\overline{E}'} \neq 0$); along with $\overline{s} \geq 0$, we conclude:

$$\begin{aligned} \sum_{(i,j)\in\overline{E}'} \overline{s}(i,j) &= \sum_{(i,j)\in\overline{E}'\setminus E'} \overline{s}(i,j) + \sum_{(i,j)\in E'} \overline{s}(i,j) \\ &\geq \sum_{(i,j)\in E'} \overline{s}(i,j) \\ &= \sum_{(i,j)\in E'} s(i,j). \end{aligned}$$

□

Lemma 7. *Let $\mathcal{G} = (V, E, s)$ a weighted digraph containing at least one arc with strictly non-negative weight. Construct the complete weighted digraph $\overline{\mathcal{G}} = (V, V^2, \overline{s})$ as follows:*

$$\forall (i,j) \in V^2,\ \overline{s}(i,j) = \begin{cases} \max\big(0, s(i,j)\big) & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{2.11}$$

Also let $(V, \overline{E}_*, \overline{s})$ MAT$\big(\overline{\mathcal{G}}\big)$-optimal (which is certain to exist since $\overline{\mathcal{G}}$ is complete). Then, one can construct (V, E_*, s) MAS$(\mathcal{G})$-optimal satisfying:

$$E_* \subset \overline{E}_* \quad \text{and} \quad \sum_{(i,j)\in E_*} s(i,j) = \sum_{(i,j)\in \overline{E}_*} \overline{s}(i,j). \tag{2.12}$$

Proof. Due to the fact that $(V, \overline{E}_*, \overline{s})$ is MAT$\big(\overline{\mathcal{G}}\big)$-optimal, we get in particular that $(V, \overline{E}_*, \overline{s})$ is MAT$\big(\overline{\mathcal{G}}\big)$-admissible and $\overline{s}|_{\overline{E}_*} \neq 0$: to see the latter, suppose by way of contradiction that $\overline{s}|_{\overline{E}_*} = 0$; but then one could construct an acyclic tournament containing one of those arcs in $\mathcal{G}$ with strictly non-negative weight, effectively contradicting the optimality of $(V, \overline{E}_*, \overline{s})$. The first clause of Lemma 6 then ensures one can construct (V, E_*, s) MAS$(\mathcal{G})$-admissible satisfying:

$$E_* \subset \overline{E}_* \quad \text{and} \quad \sum_{(i,j)\in E_*} s(i,j) = \sum_{(i,j)\in \overline{E}_*} \overline{s}(i,j).$$

All is left is to show that (V, E_*, s) is in fact optimal for the weighted MAS problem given $\mathcal{G}$ as input. Suppose by way of contradiction that (V, E_*, s) is not MAS$(\mathcal{G})$-optimal, i.e. there exists (V, E'_*, s) MAS$(\mathcal{G})$-admissible with strictly larger sum of arc weights compared to (V, E_*, s). Now, the second clause of Lemma 6 ensures one can construct $(V, \overline{E}'_*, \overline{s})$ MAT$\big(\overline{\mathcal{G}}\big)$-admissible satisfying:

$$E'_* \subset \overline{E}'_* \quad \text{and} \quad \sum_{(i,j)\in E'_*} s(i,j) \leq \sum_{(i,j)\in \overline{E}'_*} \overline{s}(i,j).$$

Combining results, we thus get:

$$\begin{aligned} \sum_{(i,j)\in \overline{E}_*} \overline{s}(i,j) &= \sum_{(i,j)\in E_*} s(i,j) \\ &< \sum_{(i,j)\in E'_*} s(i,j) \\ &\leq \sum_{(i,j)\in \overline{E}'_*} \overline{s}(i,j). \end{aligned}$$

But then, $(V, \overline{E}'_*, \overline{s})$ is MAT$\big(\overline{\mathcal{G}}\big)$-admissible with strictly larger sum of arc weights compared to $(V, \overline{E}_*, \overline{s})$, a contradiction since $(V, \overline{E}_*, \overline{s})$ is MAT$\big(\overline{\mathcal{G}}\big)$-optimal. □

2.3 On the theoretical complexity of MAS and FAS

In light of Lemma 2 which equivalently restates acyclicity as having a topological order, it is clear that at their core both MAS and FAS are permutation problems: given an input digraph (V, E), a brute-force strategy solving these problems would involve testing every possible permutation of the nodes for a naive complexity of $\mathcal{O}\big(|V|!\big)$.

Non-trivial exact algorithms are fortunately known: it was showed by Bodlaender et al. [2012] that several vertex ordering problems (including FAS) could be solved in $\mathcal{O}^*\big(2^{|V|}\big)$ time and $\mathcal{O}^*\big(2^{|V|}\big)$ space using dynamic programming (similar to the work of Held and Karp [1961] for the travelling salesman problem), or $\mathcal{O}^*\big(4^{|V|}\big)$ time and polynomial space using divide-and-conquer (similar to the work of Gurevich and Shelah [1987] for the travelling salesman problem); it was also showed by Raman and Saurabh [2007] that MAS and FAS could be solved in $\mathcal{O}^*\big(2^{|V|}\big)$ time and $\mathcal{O}^*\big(2^{|V|}\big)$ space. In fact, a trade-off between time and space exists for permutation problems (including MAS and FAS) [Koivisto and Parviainen, 2010]: more precisely, the authors showed that permutation problems (on n elements) could be solved in $\mathcal{O}^*\big(T^n\big)$ time and $\mathcal{O}^*\big(S^n\big)$ space where $TS < 4$ and $\sqrt{2} < S < 2$. Importantly, an exact algorithm solving the *minimum feedback vertex set* (FVS) problem in $\mathcal{O}^*\big(1.9977^{|V|}\big)$ time and polynomial space was introduced [Razgon, 2007], which readily gives a $\mathcal{O}^*\big(1.9977^{|E|}\big)$ time and polynomial space exact algorithm for FAS, via a linear time reduction from FVS to FAS [Festa et al., 1999].

From the perspective of parameterized complexity (see Downey and Fellows [2013]'s book for further reference on this topic), we note that fixed-parameter tractable algorithms exist for FAS (parameterized by the size of the minimum feedback arc set [Chen et al., 2008] or by treewidth [Bonamy et al., 2018]) and for MAS (parameterized by the size of the maximum acyclic subgraph [Fernau and Raible, 2008; Raman and Saurabh, 2007]). More singular is the existence of Monte-Carlo randomized algorithms solving FAS in polynomial time with arbitrary probability [Kudelic, 2016; Kudelic and Ivkovic, 2019].

Special classes of digraphs exist for which solving MAS and FAS is easier: polynomial time algorithms solving FAS were found for planar digraphs [Lucchesi and Younger, 1978], weakly acyclic digraphs [Grötschel et al., 1985] reducible flow digraphs [Ramachandran, 1988] and more recently on resolvable digraphs [Hecht, 2018]. Moreover, an exact algorithm solving MAS for $(1, n)$-graphs was proposed [Fernau and Raible, 2008] with an advantageous time complexity of $\mathcal{O}^*\big(1.1871^{|E|}\big)$.

In addition to being NP-hard problems [Karp, 1972], MAS and FAS are known to be APX-hard [Papadimitriou and Yannakakis, 1991]; remember that unless P = NP, no APX-hard problem (including MAS and FAS) has a *polynomial time approximation*

scheme (PTAS) [Ausiello et al., 1995]. We note however that a PTAS exists for FAS restricted to tournaments [Kenyon-Mathieu and Schudy, 2006]. Best known approximation algorithms for MAS and FAS have an approximation factor of $\frac{1}{2} + \Omega\big(\log(|V|)^{-1}\big)$ [Charikar et al., 2007] and $\mathcal{O}\big(\log(|V|)\log\log(|V|)\big)$ [Seymour, 1995], respectively.

2.4 Basics on integer linear programming

This short section gives a synthetic presentation of an important class of optimization problems, the so-called *mixed-integer linear programs* (MILPs), which given a matrix $A \in \mathbb{R}^{m,n}$ and two vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ take the following form:

$$\underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \; c^t x \qquad (2.13)$$
$$\text{s.t.} \begin{cases} Ax = b \\ x = [x_{\text{real}} | x_{\text{int}}] \\ x_{\text{real}} \subset \mathbb{R}, x_{\text{int}} \subset \mathbb{Z}. \end{cases}$$

Essentially, a MILP is a *linear program* (LP) where some of the variables are constrained to take integer values only (when all variables are constrained that way, we simply call it an ILP). This difference is critical: while it is well known that regular linear programming is solvable in polynomial time [Karmarkar, 1984; Khachiyan, 1979], (mixed-)integer linear programming on the other hand is NP-hard (the particular case of 0-1 integer linear programming in fact belongs to Karp [1972]'s 21 NP-hard problems).

A competitive strategy for solving MILPs is to use a *branch and cut* procedure (a *branch and bound* algorithm [Land and Doig, 2010] combined with *cutting planes*) embedding an efficient LP solver. A high level and simplified description of the procedure follows: first the integrality constraint on the integer variables x_{int} is relaxed which gives a linear program efficiently solvable (two state-of-the-art LP solvers are the simplex [Dantzig, 1990] and certain interior point [Gondzio, 2012] algorithms); second, every relaxed integer variable $x_{\text{int}}[i]$ taking value $\overset{\circ}{x}_{\text{int}}[i]$ is inspected. If this value is fractional, linear constraints are added to remove it from the search space, which corresponds to the *cutting* part:

$$\overset{\circ}{x}_{\text{int}}[i] \notin \mathbb{Z} \implies \text{Add linear constraints} \begin{cases} x_{\text{int}}[i] \leq \left\lfloor \overset{\circ}{x}_{\text{int}}[i] \right\rfloor \\ \\ x_{\text{int}}[i] \geq \left\lceil \overset{\circ}{x}_{\text{int}}[i] \right\rceil . \end{cases} \qquad (2.14)$$

Since these two constraints cannot hold at the same time, two new linear programs are created, each receiving one of the two constraints: this is the *branching* part. Repeat-

ing this branching process yields a search tree of LPs that are relaxations of the original MILP. As a byproduct, assuming the MILP is a minimization problem, the largest optimal score among LP relaxations in the search tree is the tightest lower bound available on the MILP's optimal score; besides, when all relaxed integer variables do take integral values, a valid integral solution is found which yields an upper bound on the MILP's optimal score. The algorithm stops when the lower bound and the upper bound on the MILP's optimal score coincide, that is the algorithm is exact.

2.5 ILPs for weighted MAS and FAS problems

This section builds upon Section 2.2, translating the mathematical framework behind the weighted maximum acyclic subgraph problem into various exact optimization strategies. In order for the reader to better assess the practical cost of acyclicity, these different strategies were implemented as integer linear programs (briefly introduced in Section 2.4) and a scalability experiment was conducted (see Figure 2.5).

Weighted MAS problems (as presented in Definition 11) can be formulated as ILPs. Formally, given a weighted digraph $\mathcal{G} = (V, E, s)$ and denoting E_+ the set of arcs with strictly non-negative weights, we create the set of binary variables $I = \big\{I(i,j) \in \{0,1\} : (i,j) \in E_+\big\}$, where $I(i,j)$ takes value 1 if the corresponding arc is in the solution, 0 otherwise. Then, the weighted MAS problem as formulated in Equation 2.5 (given $\mathcal{G}$ as input) can be equivalently rewritten:

$$\begin{aligned} I_* \;=\; & \underset{I}{\operatorname{argmax}} \; \textstyle\sum_{(i,j)\in E_+} s(i,j) I(i,j) \\ & \text{s.t. } I \text{ represents a DAG,} \end{aligned} \tag{2.15}$$

where the global constraint "I represents a DAG" must be formalized into a set of linear constraints. Based on Definition 7, an immediate formalization is to add constraints removing every existing cycle in the input digraph. Formally, let $\mathcal{C}(\mathcal{G})$ denote the set of all cycles contained in $\mathcal{G}$; given a l-cycle of the form $v_0 \to \cdots \to v_{l-1} \to v_0 \in \mathcal{C}(\mathcal{G})$, one can ensure this cycle is cut from the search space by enforcing the following integer linear constraint:

$$\sum_{k=0}^{l-1} I(v_k, v_{k+1 \bmod l}) \leq l - 1. \tag{2.16}$$

This follows immediately from the fact that the left hand-side in the previous constraint is a sum of l binary variables, such that at least one of these variables is forced to take value 0; but then, $I(v_k, v_{k+1 \bmod l}) = 0$ means exactly that the arc $(v_k, v_{k+1 \bmod l})$ is not in the solution, effectively *cutting* the cycle $v_0 \to \cdots \to v_{l-1} \to v_0$. Although easily formalized,

this approach has an important drawback in practice: the number of cycles in a digraph may grow super-exponentially with the number of nodes (see Figure 2.3 in which the extreme case of complete digraphs is discussed). In fact, even sparse digraphs may contain a prohibitive amount of cycles, as emphasized in Figure 2.4. It is well-known that counting cycles in a digraph is a #P-hard problem [Valiant, 1979] and the best available algorithms for cycle enumeration have time complexity $\mathcal{O}\big((|V|+|E|)\times(|C|+1)\big)$ given an input digraph with $|V|$ nodes, $|E|$ arcs and $|C|$ (elementary) cycles [Johnson, 1975; Tarjan, 1973].

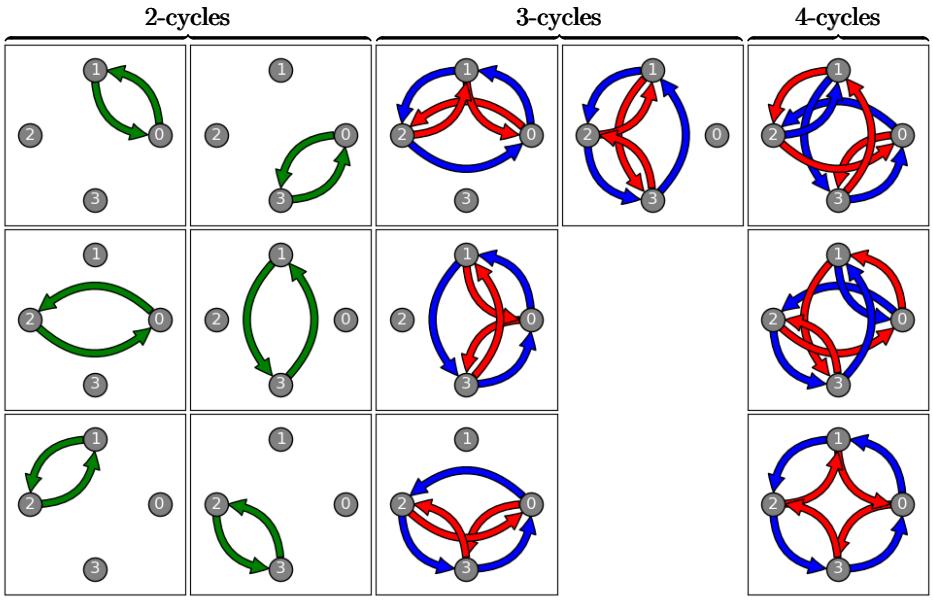

Figure 2.3: Visualization of every cycle in a complete digraph with 4 nodes. In general, a complete digraph with d nodes has $(l-1)!\cdot\binom{d}{l}$ l-cycles. If $l\leq 2$, $(l-1)!=1$ i.e. l-cycles where $l\leq 2$ have a single orientation (represented in green for $l=2$); otherwise, 2 is a divisor of $(l-1)!$ i.e. l-cycles where $l>2$ have two orientations (represented in red and blue for $l=3,4$). The total number of cycles grows super-exponentially in the number of nodes. Assuming all orientations are depicted on the same subplot, one would need in order to visualize every cycle in a complete digraph with 5 (respectively 6) nodes a total of 47 (respectively 212) subplots.

Fortunately, one does not need to enforce cycle constraints for every cycle in the input digraph in order to guarantee acyclicity, rendering exhaustive cycle enumeration unnecessary: indeed, state-of-the-art ILP solvers [Bestuzheva et al., 2021; Gurobi Optimization, LLC, 2023; ILOG CPLEX Optimization Studio, 2022; MOSEK ApS, 2022] offer the possibility to *lazily* add constraints to the optimization model during the optimization process. In practice, without the global constraint "I represents a DAG", an

ILP solver may return an optimal solution to Equation 2.15 that contains a fraction of the cycles found in the input digraph; when this happens, it suffices to invalidate the current optimal solution by injecting linear constraints (as described in Equation 2.16) cutting this fraction of the cycles. The optimization process is then resumed with the newly added constraints; this strategy is repeated until an acyclic optimal solution is returned by the ILP solver. This so-called *lazy* strategy was recently investigated [Baharev et al., 2021; Grötschel et al., 2022] and proved to scale significantly better than the classical approach relying on exhaustive cycle enumeration.

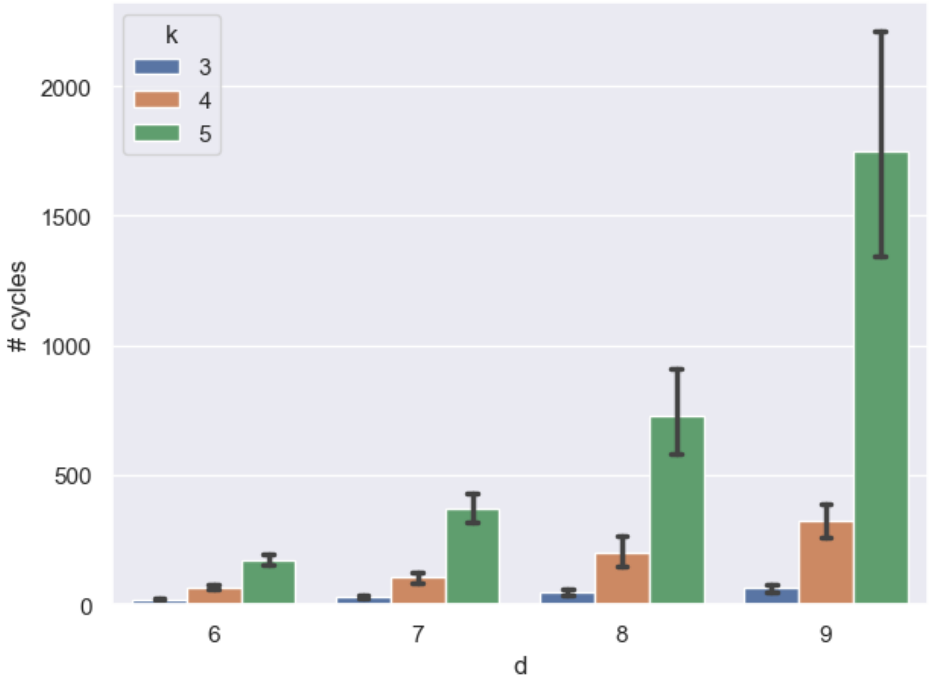

Figure 2.4: Visualization of the number of cycles in randomly generated digraphs. Digraphs are generated starting from a complete digraph with d nodes; for every arc in the complete digraph, that arc has probability $\frac{k}{d}$ to be kept, resulting in sparse digraphs with average in-degree converging to k as d increases. Bar plots and error bars represent the mean and the standard deviation of the number of cycles in 50 randomly generated digraphs for every tested pair (d, k). We note $v_0 \to \cdots \to v_{l-1} \to v_0$ and $v_i \to \cdots \to v_{l-1} \to v_0 \to \cdots \to v_i$ are counted as a single cycle. Both the mean and the standard deviation quickly explode even when the number of nodes is small and the generated digraphs are sparse.

It turns out that one can in fact model weighted MAS problems as ILPs with a number of linear constraints independent from the number of cycles contained in the input digraph.

This other implementation relies on the fact that solving a weighted MAS instance reduces to solving a related weighted MAT instance, as demonstrated in Lemma 7. Formally, given a weighted digraph $\mathcal{G} = (V, E, s)$ we construct the complete weighted digraph $\overline{\mathcal{G}} = (V, V^2, \overline{s})$ satisfying Equation 2.11; we also create the set of binary variables $\overline{I} = \left\{\overline{I}(i,j) \in \{0,1\} : (i,j) \in V^2\right\}$, where $\overline{I}(i,j)$ takes value 1 if the corresponding arc is in the solution, 0 otherwise. Then, the weighted MAT problem as formulated in Equation 2.7 (given $\overline{\mathcal{G}}$ as input) can be equivalently rewritten:

$$\begin{aligned} \overline{I}_* \;=\; & \underset{\overline{I}}{\operatorname{argmax}} \textstyle\sum_{(i,j)\in V^2} \overline{s}(i,j)\overline{I}(i,j) \\ & \text{s.t. } \overline{I} \text{ represents an acyclic tournament,} \end{aligned} \tag{2.17}$$

where the global constraint "$\overline{I}$ represents an acyclic tournament" must be formalized into a set of linear constraints. This set is readily given by Lemma 4, which tells us that acyclic tournaments are exactly those digraphs with a complete skeleton and no 2 or 3-cycle. The following set of linear constraints ensures these properties are satisfied when $\overline{\mathcal{G}}$ has d nodes:

$$\begin{cases} \forall (i,j) \in [\![0, d-1]\!]^2 : i,j \text{ distinct}, & \overline{I}(i,j) + \overline{I}(j,i) = 1 \\ \forall (i,j,k) \in [\![0, d-1]\!]^3 : i,j,k \text{ distinct}, & \overline{I}(i,j) + \overline{I}(j,k) + \overline{I}(k,i) \leq 2. \end{cases} \tag{2.18}$$

As desired, the cardinality of the previous set of constraints is independent from the number of cycles in $\overline{\mathcal{G}}$ (and $\mathcal{G}$). It is however dominated by the number of 3-cycles in a complete digraph with d nodes, precisely $2 \cdot \binom{d}{3}$ (i.e. resulting in a number of constraints that is cubic in the number of nodes and irrespective of the sparsity of the initial digraph $\mathcal{G}$; see Table 2.1). Now, Lemmas 6, 7 ensure that an optimal solution for the weighted MAS problem given $\mathcal{G}$ as input is $I_* = \left\{I_*(i,j) \coloneqq \overline{I}_*(i,j) \cdot 1_{|\overline{s}(i,j)>0} : (i,j) \in V^2\right\}$.

MAS ILP type	**# Binary variables**	**# Linear constraints**
Acyclic tournament	$\lvert V\rvert^2 - \lvert V\rvert$	$2 \cdot \binom{\lvert V\rvert}{3} + \binom{\lvert V\rvert}{2}$
Full cycle set cover	$\lvert E_+\rvert$	$\lvert C\rvert$
Lazy cycle set cover	$\lvert E_+\rvert$	$\leq \lvert C\rvert$

Table 2.1: Number of binary variables and linear constraints needed to model various ILPs solving a weighted MAS problem given as input a weighted digraph with node set V, strictly non-negative arc set E_+ and cycle set cover C.

Figure 2.5 illustrates the practical difference between the acyclic tournament and the lazy cycle set cover MAS ILP formulations. We empirically showed that:

- Setting the acyclic tournament MAS ILP formulation scales very unfavorably compared to setting the lazy cycle set cover MAS ILP formulation.

- Solving the acyclic tournament MAS ILP formulation scales poorly compared to solving the lazy cycle set cover MAS ILP formulation, although the difference in scalability becomes less pronounced as input digraphs become denser.

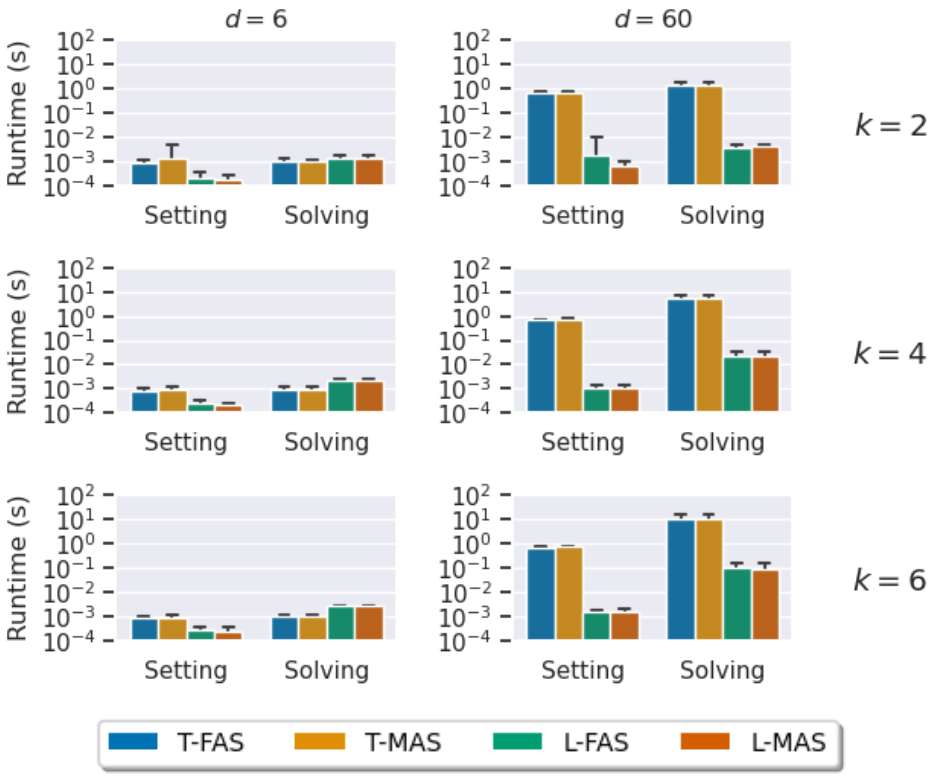

Figure 2.5: Temporal scalability assessment of ILPs for solving weighted MAS and FAS problems. Both the tournament-based approach ("T") and the lazy cycle set cover approach ("L") were compared. These ILPs were applied on randomly generated weighted digraphs with d nodes, $k \times d$ arcs on average (where the hyperparameter k controls sparsity) and arc weights sampled from a uniform distribution with range $[0, 1]$ $\big($50 instances for every pair $(d, k)\big)$. Bar plots represent the runtime (in seconds) for setting and solving ILPs averaged across instances and error bars depict the variance; ordinate axis is represented in logarithmic scale for clarity. When the number of nodes d increases by an order of magnitude, both the setting and solving runtime in the tournament-based variant increase by 3 to 4 orders of magnitude, whereas the setting and solving runtime in the lazy cycle set cover variant merely increase by 1 to 2 orders of magnitude. ILPs for MAS and FAS exhibit very similar temporal scalability. Hardware: Intel Core i7-9750H CPU 2.6GHz $\times$ 12; Software: Gurobi 10.0.1.

2.6 Greedy heuristics for weighted MAS and FAS problems

The exponential complexity of exact algorithms for MAS and FAS (see Section 2.3) renders these algorithms unsuited for large input digraphs. Instead, one typically resorts to using heuristics [Simpson et al., 2016]: these include DFS-based [Even and Even, 2012] and sort-based [Ailon et al., 2008; Brandenburg and Hanauer, 2011] heuristics, but also the well known Berger and Shor [1990] and Eades et al. [1993] heuristics. The latter (Eades') is a class of greedy algorithms approximately solving FAS and the original heuristic [Eades et al., 1993] is guaranteed to return solutions with at most $\frac{1}{2}|E| - \frac{1}{6}|V|$ arcs given an input (unweighted) digraph (V, E), although Simpson et al. [2016] report that greedy heuristics tend to perform significantly better than the worst case in practice.

Greedy heuristics for solving weighted MAS and FAS problems build upon the idea that the acyclicity property in digraphs can be equivalently restated as identifying a permutation of the nodes and a strictly upper-triangular matrix obtained by permuting the adjacency matrix of the digraph with that permutation (see Lemma 3). In other words, one will be able to identify a large acyclic subgraph (respectively a small feedback arc set) by finding a permutation of the nodes such that the image of the weighted adjacency matrix by this permutation would mostly have arcs with large weights (respectively small weights) in the top-right corner, also called forward arcs: this first step is described in Algorithms 2.1-2.4, where the permutation is sequentially constructed based on locally optimal decisions as illustrated in Figure 2.6; then, as a consequence of Lemma 3 (and Remark 2) it suffices to project on the top-right corner (respectively bottom-left corner) in order to recover the approximate maximum acyclic subgraph: this second step is described in Algorithm 2.11. Notice that in the aforementioned first step, Algorithms 2.1, 2.2 greedily construct the permutation by inspecting the sum of weights of outgoing arcs (outscores) whereas Algorithms 2.3, 2.4 greedily construct the permutation by inspecting the sum of weights of incoming arcs (inscores).

Alternatively, a more involved procedure can be used for the first step: by attempting to maximize the absolute difference between outscores and inscores as in Algorithm 2.5, one can hope to identify more efficiently these structures in the input weighted digraph where breaking a potential cycle can be achieved while retaining most of the arc weights (see Figure 2.7 for an illustration of this idea). Although very similar in their implementation, Algorithms 2.1-2.5 result in heuristics whose practical behavior may be surprisingly different depending on the properties of the weighted digraph they receive as input, as illustrated in Figure 2.8. In short, greedy FAS variants (Algorithms 2.2, 2.4) noticeably outperform other greedy heuristics given sufficiently sparse instances. This matters in

the context of Bayesian network structure learning, where sparsity is often a desired property.

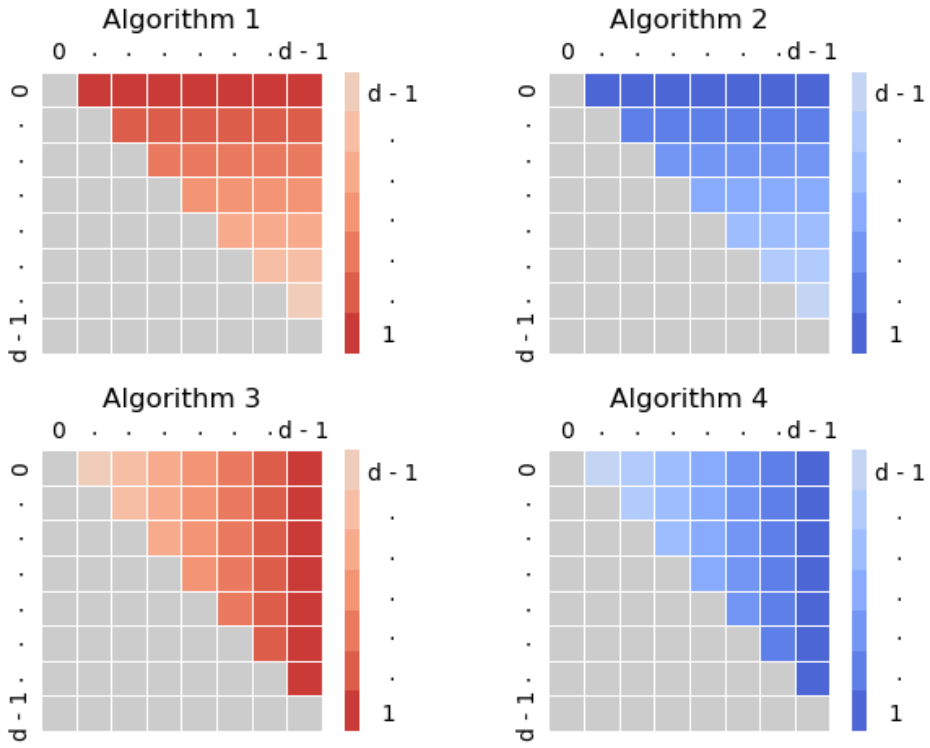

Figure 2.6: Visualization of the four greedy MAS and FAS variants whose pseudo-codes are given in Algorithms 2.1-2.4. MAS (respectively FAS) variants attempt to fill as much (respectively little) weight inside the top-right corner of the matrix (representing the input weighted digraph permuted by the permutation operator greedily constructed); they are depicted on the left (respectively on the right) with a red (respectively blue) colormap. Colormaps indicate the order in which greedy heuristics fill the matrix based on the iteration (there are always $d-1$ iterations given an input digraph with d nodes): outscore-based (respectively inscore-based) variants are depicted at the top (respectively at the bottom), filling the top-right corner of the matrix from the top rows up to the bottom ones (respectively from the right columns up to the left ones).

Algorithm 2.1 Greedy permutation for weighted MAS (outscore-based)

Input: $\mathcal{G} = (V, E, s) : \ V = [\![0, d-1]\!]$
Output: Permutation $\pi : V \to V$

1: $V_1 = V$
2: **for** r from 1 to d **do**
3: $\quad v_r = \underset{i \in V_r}{\operatorname{argmax}} \sum_{j \in V_r \setminus \{i\}} s(i,j)$
4: $\quad \pi : v_r \mapsto r - 1$
5: $\quad V_{r+1} = V_r \setminus \{v_r\}$
6: **end for**
7: **return** π

Algorithm 2.2 Greedy permutation for weighted FAS (outscore-based)

Input: $\mathcal{G} = (V, E, s) : \ V = [\![0, d-1]\!]$
Output: Permutation $\pi : V \to V$

1: $V_1 = V$
2: **for** r from 1 to d **do**
3: $\quad v_r = \underset{i \in V_r}{\operatorname{argmin}} \sum_{j \in V_r \setminus \{i\}} s(i,j)$
4: $\quad \pi : v_r \mapsto r - 1$
5: $\quad V_{r+1} = V_r \setminus \{v_r\}$
6: **end for**
7: **return** π

Algorithm 2.3 Greedy permutation for weighted MAS (inscore-based)

Input: $\mathcal{G} = (V, E, s) : \ V = [\![0, d-1]\!]$
Output: Permutation $\pi : V \to V$

1: $V_1 = V$
2: **for** r from 1 to d **do**
3: $\quad v_r = \underset{j \in V_r}{\operatorname{argmax}} \sum_{i \in V_r \setminus \{j\}} s(i,j)$
4: $\quad \pi : v_r \mapsto d - r$
5: $\quad V_{r+1} = V_r \setminus \{v_r\}$
6: **end for**
7: **return** π

Algorithm 2.4 Greedy permutation for weighted FAS (inscore-based)

Input: $\mathcal{G} = (V, E, s) : \ V = [\![0, d-1]\!]$
Output: Permutation $\pi : V \to V$

1: $V_1 = V$
2: **for** r from 1 to d **do**
3: $\quad v_r = \underset{j \in V_r}{\operatorname{argmin}} \sum_{i \in V_r \setminus \{j\}} s(i,j)$
4: $\quad \pi : v_r \mapsto d - r$
5: $\quad V_{r+1} = V_r \setminus \{v_r\}$
6: **end for**
7: **return** π

We remark that the naive algorithmic complexity of Algorithms 2.1-2.5 is $\mathcal{O}\big(|V|^2\big)$, since these heuristics consist of $|V|$ iterations, where every iteration itself has complexity $\mathcal{O}\big(|V|\big)$. An implementation relying on efficient data structures was proposed [Simpson et al., 2016] to bring the complexity down to $\mathcal{O}\big(|V|+|E|\big)$ in both time and space, effectively taking advantage of the input digraph's sparsity (in terms of the number of arcs). When dealing with weighted instances however, it may happen that the input digraph is complete, yet only a small portion of arcs have significant weight. In this context, one should resort to using vectorized implementations (see vectorized Algorithms 2.6-2.10) whose scalability will only depend on the number of nodes in the input digraph, irrespective of its arc set; we stress that greedy heuristics can only be partially vectorized owing to their sequential nature. As a byproduct, the possibility to (partially) vectorize greedy heuristics enables to approximately solve combinatorial problems such as weighted MAS

and FAS on specialised hardware capable of handling parallelism, notably GPUs.

Algorithm 2.5 Advanced greedy permutation for weighted MAS

Input: $\mathcal{G} = (V, E, s) : \ V = [\![0, d-1]\!]$

Output: Permutation $\pi : V \to V$

$V_1 = V$, $r_{\text{start}} = 0$, $r_{\text{end}} = 0$
for r from 1 to d **do**
  $f_r : k \in V_r \mapsto \sum_{j \in V_r \setminus \{k\}} s(k,j) - \sum_{i \in V_r \setminus \{k\}} s(i,k)$
  $v_r = \underset{k \in V_r}{\text{argmax}} \, |f_r(k)|$
  if $f_r(v_r) > 0$ **then**
    $r_{\text{start}} \mathrel{+}= 1$
    $\pi : v_r \mapsto r_{\text{start}} - 1$
  else
    $r_{\text{end}} \mathrel{+}= 1$
    $\pi : v_r \mapsto d - r_{\text{end}}$
  end if
  $V_{r+1} = V_r \setminus \{v_r\}$
end for
return π

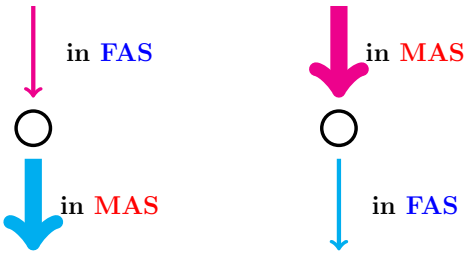

Figure 2.7: Illustration of the greedy strategy used in Algorithm 2.5: the heuristic prioritizes in finding those nodes for which the absolute difference between outscores (depicted in cyan) and inscores (depicted in magenta) is large. When the outscores dominate (left), outgoing arcs are placed inside the maximum acyclic subgraph whereas incoming arcs are placed inside the minimum feedback arc set; when the inscores dominate (right), incoming arcs are placed inside the maximum acyclic subgraph whereas outgoing arcs are placed inside the minimum feedback arc set. Doing so, any cycle passing through the node is cut while retaining as much weight as possible in the maximum acyclic subgraph.

Algorithm 2.6 Vectorized greedy permutation for weighted MAS (outscore-based)

Input: $W \in \mathbb{R}^{d \times d}$

Output: Permutation vector π

$\pi = \text{zeros}(d)$, $I = \text{identity}(d)$
$W[I] = 0$
$\text{outscores} = W.\text{sum}(\text{dim}{=}1)$
for r from 1 to d **do**
  $v_r = \text{argmax}(\text{outscores})$
  $\pi[r-1] = v_r$
  $\text{outscores}[v_r] = -\infty$
  $\text{outscores} \mathrel{-}= W[:, v_r]$
end for
return π

Algorithm 2.7 Vectorized greedy permutation for weighted FAS (outscore-based)

Input: $W \in \mathbb{R}^{d \times d}$

Output: Permutation vector π

$\pi = \text{zeros}(d)$, $I = \text{identity}(d)$
$W[I] = 0$
$\text{outscores} = W.\text{sum}(\text{dim}{=}1)$
for r from 1 to d **do**
  $v_r = \text{argmin}(\text{outscores})$
  $\pi[r-1] = v_r$
  $\text{outscores}[v_r] = +\infty$
  $\text{outscores} \mathrel{-}= W[:, v_r]$
end for
return π

Algorithm 2.8 Vectorized greedy permutation for weighted MAS (inscore-based)

Input: $W \in \mathbb{R}^{d \times d}$

Output: Permutation vector π

$\pi = \text{zeros}(d)$, $I = \text{identity}(d)$
$W[I] = 0$
$\text{inscores} = W.\text{sum}(\text{dim}{=}0)$
for r from 1 to d **do**
  $v_r = \text{argmax}(\text{inscores})$
  $\pi[d-r] = v_r$
  $\text{inscores}[v_r] = -\infty$
  $\text{inscores} \mathrel{-}= W[v_r, :]$
end for
return π

Algorithm 2.9 Vectorized greedy permutation for weighted FAS (inscore-based)

Input: $W \in \mathbb{R}^{d \times d}$

Output: Permutation vector π

$\pi = \text{zeros}(d)$, $I = \text{identity}(d)$
$W[I] = 0$
$\text{inscores} = W.\text{sum}(\text{dim}{=}0)$
for r from 1 to d **do**
  $v_r = \text{argmin}(\text{inscores})$
  $\pi[d-r] = v_r$
  $\text{inscores}[v_r] = +\infty$
  $\text{inscores} \mathrel{-}= W[v_r, :]$
end for
return π

Algorithm 2.10 Vectorized advanced greedy permutation for weighted MAS

Input: $W \in \mathbb{R}^{d \times d}$

Output: Permutation vector π

$\pi = \text{zeros}(d)$, $I = \text{identity}(d)$
$W[I] = 0$
$W_{\text{diff}} = W - \text{transpose}(W)$
scores $= W_{\text{diff}}.\text{sum}(\text{dim}{=}1)$
squared-scores $= \text{power}(\text{scores}, 2)$
$r_{\text{start}} = 0$, $r_{\text{end}} = 0$
for r from 1 to d **do**
  $v_r =$ argmax(squared-scores)
  if scores$[v_r] > 0$ **then**
    r_{start} += 1
    $\pi[r_{\text{start}} - 1] = v_r$
  else
    r_{end} += 1
    $\pi[d - r_{\text{end}}] = v_r$
  end if
  squared-scores$[v_r] = -\infty$
  $s = W_{\text{diff}}[:, v_r]$
  squared-scores −= multiply$(s, 2 \cdot \text{scores} - s)$
  scores −= s
end for
return π

Algorithm 2.11 Topological order-based acyclic projection

Input: $\mathcal{G} = (V, E)$: $V = [\![0, d-1]\!]$ with adjacency matrix $\widetilde{A}$, permutation operator $\pi : V \to V$, projection corner

Output: Acyclic projection A of $\widetilde{A}$

$T = \left(\widetilde{A}[\pi^{-1}(i), \pi^{-1}(j)]\right)_{(i,j) \in [\![0,d-1]\!]^2}$
if projection corner = top-right **then**
  $\forall i \geq j$: set $T[i, j] = 0$
else if projection corner = bottom-left **then**
  $\forall i \leq j$: set $T[i, j] = 0$
end if
$A = \left(T[\pi(i), \pi(j)]\right)_{(i,j) \in [\![0,d-1]\!]^2}$
return A

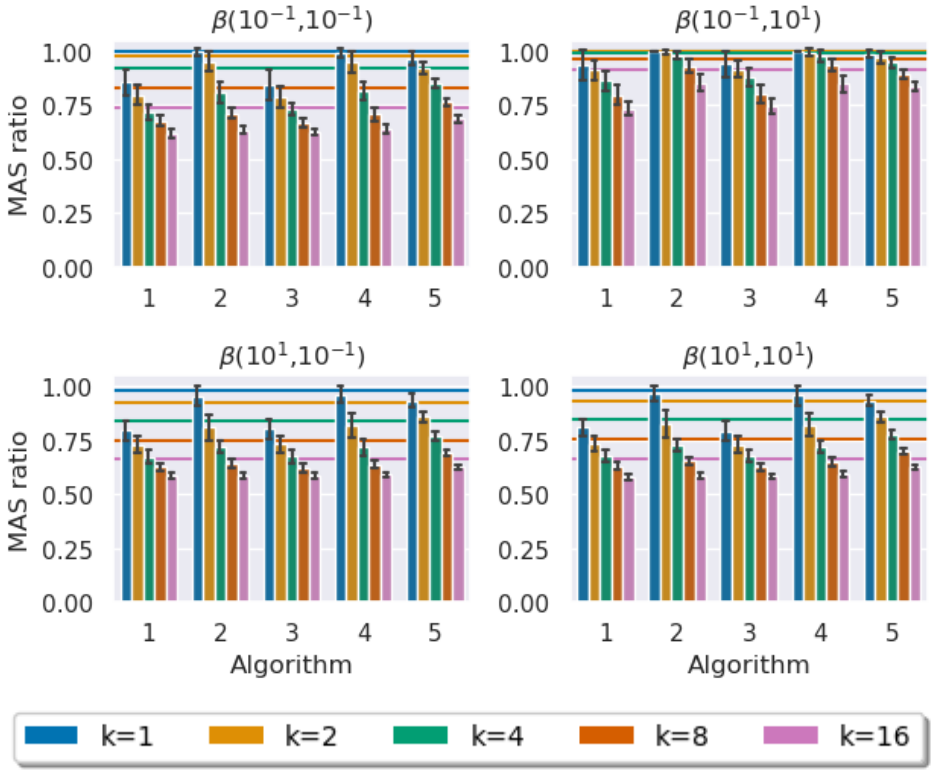

Figure 2.8: Empirical comparison of the five greedy MAS and FAS variants whose pseudocodes are given in Algorithms 2.1-2.5. These heuristics were applied on randomly generated weighted digraphs with $d = 50$ nodes, $k \times d$ arcs on average (where the hyperparameter k controls sparsity) and arc weights sampled from various Beta distributions (50 instances for every pair of k and Beta distribution). Heuristics were compared using the MAS ratio metric, that is the ratio between the sum of arc weights in the approximate MAS solution and the sum of arc weights in the input weighted digraph. Bar plots represent MAS ratios averaged across instances and error bars depict the variance; additionally, colored horizontal lines provide optimal MAS ratios averaged across instances and were obtained using an exact MAS algorithm. In extreme sparsity setting ($k = 1$), Algorithms 2.2, 2.4 (greedy FAS variants) significantly outperform Algorithms 2.1, 2.3 (greedy MAS variants) and slightly outperform Algorithm 2.5 (advanced greedy MAS variant). As k increases, the performance of Algorithms 2.1-2.4 becomes more similar, while Algorithm 2.5 takes the lead.

2.7 Proximal gradient descent

In this section, we provide theoretical background from convex optimization, more specifically the problem of minimizing a composite convex function where both components are convex but only one of the components is *smooth*. Such functions are typically encountered in machine learning in the presence of a non-smooth but convex regularizer, as in LASSO [Tibshirani, 1996] where the L1 norm is used to promote sparsity.

The *proximal gradient descent* algorithm is a tailored optimizer for aforementioned optimization problems and we recall some of its theoretical analysis; the famous *iterative shrinkage-thresholding algorithm* (ISTA, described in Algorithm 2.12) is the classical (non-accelerated) variant of proximal gradient descent. In particular, a key result that is used in this thesis (see Lemma 42) is the so-called *descent lemma* (Lemma 10), which states that the sequence generated by proximal gradient descent is guaranteed to decrease function value when minimizing a function of the aforementioned class.

This section assumes the reader is familiar with the field of convex analysis and notably with the notion of subgradient (represented by the functional operator $g \mapsto \partial g$), generalizing gradients to convex functions even at points of non-differentiability. A refresher on important notions from convex analysis is provided in Appendix (Sections 5.1-5.4).

Proximal gradient descent, much like gradient descent, is an iterative process for updating parameters in order to decrease function value. Every update consists in minimizing a least-squared-regularized subproblem, also called a *proximal operator*:

Definition 14. *Let $g : \mathbb{R}^n \to \mathbb{R}$ a convex function and $a \in \mathbb{R}^n$. The **proximal operator at point a** of g is defined as:*

$$\mathcal{P}rox(g)(a) = \underset{x\in\mathbb{R}^n}{argmin} \left(g(x) + \frac{1}{2}\|x - a\|^2 \right).$$

Definition 15. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a convex differentiable function with an L-Lipschitz continuous gradient and $g : \mathbb{R}^n \to \mathbb{R}$ a convex function. The **proximal gradient descent** algorithm is the optimizer generating new parameters x_k from previous parameters x_{k-1} as follows:*

$$x_k = \underset{x\in\mathbb{R}^n}{argmin} \left(g(x) + \frac{\gamma_k^{-1}}{2} \left\| x - \left(x_{k-1} - \gamma_k \nabla f(x_{k-1}) \right) \right\|^2 \right) \quad \text{where} \quad \gamma_k \in \left]0, L^{-1}\right]. \tag{2.19}$$

An equivalent definition is:

$$x_k = \mathcal{P}rox(\gamma_k g)\left(x_{k-1} - \gamma_k \nabla f(x_{k-1}) \right) \quad \text{where} \quad \gamma_k \in \left]0, L^{-1}\right]. \tag{2.20}$$

Lemmas 8, 9, quite technical, are needed for the convergence analysis of the (non-accelerated) proximal gradient descent:

Lemma 8. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a function differentiable at x_{k-1}, $g : \mathbb{R}^n \to \mathbb{R}$ a convex function and $\gamma_k > 0$. Then the following holds:*

$$x_k = \mathcal{P}rox(\gamma_k g)\big(x_{k-1} - \gamma_k \nabla f(x_{k-1})\big) \iff \gamma_k^{-1}(x_{k-1} - x_k) - \nabla f(x_{k-1}) \in \partial g(x_k). \quad (2.21)$$

Proof. Let us write $h : x \mapsto \frac{\gamma_k^{-1}}{2}\big\|x - \big(x_{k-1} - \gamma_k \nabla f(x_{k-1})\big)\big\|^2$. The function h is clearly convex and differentiable; its gradient at x_k is $\nabla h(x_k) = \gamma_k^{-1}\big(x_k - x_{k-1} + \gamma_k \nabla f(x_{k-1})\big)$. Now, by definition of the proximal operator and using the three first clauses from Lemma 27, we deduce:

$$\begin{aligned}
x_k = \mathcal{P}rox(\gamma_k g)\big(x_{k-1} - \gamma_k \nabla f(x_{k-1})\big) &\iff x_k = \operatorname*{argmin}_{x \in \mathbb{R}^n} \big(g(x) + h(x)\big) \\
&\iff 0 \in \partial(g + h)(x_k) \\
&\iff 0 \in \partial g(x_k) + \partial h(x_k) \\
&\iff 0 \in \partial g(x_k) + \{\nabla h(x_k)\} \\
&\iff 0 \in \partial g(x_k) \\
&\qquad + \big\{\gamma_k^{-1}\big(x_k - x_{k-1} + \gamma_k \nabla f(x_{k-1})\big)\big\} \\
&\iff -\gamma_k^{-1}\big(x_k - x_{k-1} + \gamma_k \nabla f(x_{k-1})\big) \in \partial g(x_k) \\
&\iff \gamma_k^{-1}(x_{k-1} - x_k) - \nabla f(x_{k-1}) \in \partial g(x_k).
\end{aligned}$$

□

Lemma 9. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a convex differentiable function with an L-Lipschitz continuous gradient and $g : \mathbb{R}^n \to \mathbb{R}$ a convex function. Consider $\phi = f + g$ and generate $x_k = \mathcal{P}rox(\gamma_k g)\big(x_{k-1} - \gamma_k \nabla f(x_{k-1})\big)$ where $\gamma_k \in]0, L^{-1}]$. Then the following holds:*

$$\forall x \in \mathbb{R}^n,\ \phi(x_k) \leq \phi(x) + \gamma_k^{-1}\langle x_{k-1} - x_k, x_{k-1} - x\rangle - \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2. \quad (2.22)$$

Proof. Fix $x \in \mathbb{R}^n$. Firstly, since f is differentiable with an L-Lipschitz continuous gradient we can use Lemma 33 and obtain:

$$\begin{aligned}
f(x_k) &\leq f(x_{k-1}) + \big\langle \nabla f(x_{k-1}), x_k - x_{k-1}\big\rangle + \frac{L}{2}\|x_k - x_{k-1}\|^2 \\
&\leq f(x_{k-1}) + \big\langle \nabla f(x_{k-1}), x_k - x_{k-1}\big\rangle + \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2.
\end{aligned}$$

Secondly, since f is convex and differentiable, Lemma 28 yields:

$$f(x) - f(x_{k-1}) \geq \big\langle \nabla f(x_{k-1}), x - x_{k-1}\big\rangle,$$

that is:

$$f(x_{k-1}) \leq f(x) + \big\langle \nabla f(x_{k-1}), x_{k-1} - x \big\rangle.$$

Combined with the previous result, we get:

$$\begin{aligned} f(x_k) &\leq f(x_{k-1}) + \big\langle \nabla f(x_{k-1}), x_k - x_{k-1} \big\rangle + \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2 \\ &\leq f(x) + \big\langle \nabla f(x_{k-1}), x_{k-1} - x \big\rangle + \big\langle \nabla f(x_{k-1}), x_k - x_{k-1} \big\rangle + \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2 \\ &= f(x) + \big\langle \nabla f(x_{k-1}), x_k - x \big\rangle + \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2. \end{aligned}$$

Thirdly, since f is differentiable and g is convex, Lemma 8 guarantees that one has $\gamma_k^{-1}(x_{k-1} - x_k) - \nabla f(x_{k-1}) \in \partial g(x_k)$. By definition of subgradient, this implies:

$$g(x) - g(x_k) \geq \big\langle \gamma_k^{-1}(x_{k-1} - x_k) - \nabla f(x_{k-1}), x - x_k \big\rangle,$$

that is:

$$g(x_k) \leq g(x) + \big\langle \gamma_k^{-1}(x_{k-1} - x_k) - \nabla f(x_{k-1}), x_k - x \big\rangle.$$

Combined with the previous result, we finally get:

$$\begin{aligned} \phi(x_k) &= f(x_k) + g(x_k) \\ &\leq f(x) + \big\langle \nabla f(x_{k-1}), x_k - x \big\rangle + \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2 \\ &\quad + g(x) + \big\langle \gamma_k^{-1}(x_{k-1} - x_k) - \nabla f(x_{k-1}), x_k - x \big\rangle \\ &= \phi(x) + \gamma_k^{-1}\langle x_{k-1} - x_k, x_k - x \rangle + \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2 \\ &= \phi(x) + \gamma_k^{-1}\big\langle x_{k-1} - x_k, x_{k-1} - x - (x_{k-1} - x_k) \big\rangle + \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2 \\ &= \phi(x) + \gamma_k^{-1}\big(\langle x_{k-1} - x_k, x_{k-1} - x \rangle - \langle x_{k-1} - x_k, x_{k-1} - x_k \rangle\big) + \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2 \\ &= \phi(x) + \gamma_k^{-1}\langle x_{k-1} - x_k, x_{k-1} - x \rangle - \gamma_k^{-1}\|x_k - x_{k-1}\|^2 + \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2 \\ &= \phi(x) + \gamma_k^{-1}\langle x_{k-1} - x_k, x_{k-1} - x \rangle - \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2. \end{aligned}$$

□

Lemma 10 is the very important descent lemma, guaranteeing decrease in function value:

Lemma 10. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a convex differentiable function with an L-Lipschitz continuous gradient and $g : \mathbb{R}^n \to \mathbb{R}$ a convex function. Consider $\phi = f + g$ and generate*

$x_k = \mathcal{P}rox(\gamma_k g)\big(x_{k-1} - \gamma_k \nabla f(x_{k-1})\big)$ *where* $\gamma_k \in]0, L^{-1}]$. *Then the following holds:*

$$\phi(x_k) \leq \phi(x_{k-1}) - \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2. \tag{2.23}$$

Proof. This trivially follows from Lemma 9 applied with $x = x_{k-1}$. □

Lemma 11 crucially bounds the difference between function value at the current parameters estimate and the global minimum by a telescopic term:

Lemma 11. *Let* $f : \mathbb{R}^n \to \mathbb{R}$ *a convex differentiable function with an L-Lipschitz continuous gradient and* $g : \mathbb{R}^n \to \mathbb{R}$ *a convex function. Consider* $\phi = f + g$ *and generate* $x_k = \mathcal{P}rox(\gamma_k g)\big(x_{k-1} - \gamma_k \nabla f(x_{k-1})\big)$ *where* $\gamma_k \in]0, L^{-1}]$. *Then, given any global minimizer* x_* *of* ϕ, *the following holds:*

$$\phi(x_k) - \phi(x^*) \leq \frac{\gamma_k^{-1}}{2}\big(\|x_{k-1} - x_*\|^2 - \|x_k - x_*\|^2\big). \tag{2.24}$$

Proof. Applying Lemma 9 with $x = x_*$ yields:

$$\begin{aligned}
\phi(x_k) - \phi(x_*) &\leq \gamma_k^{-1}\langle x_{k-1} - x_k, x_{k-1} - x_*\rangle - \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2 \\
&= \gamma_k^{-1}\langle x_{k-1} - x_k, x_{k-1} - x_k + x_k - x_*\rangle - \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2 \\
&= \gamma_k^{-1}\|x_{k-1} - x_k\|^2 + \gamma_k^{-1}\langle x_{k-1} - x_k, x_k - x_*\rangle - \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2 \\
&= \frac{\gamma_k^{-1}}{2}\|x_{k-1} - x_k\|^2 + \gamma_k^{-1}\langle x_{k-1} - x_k, x_k - x_*\rangle.
\end{aligned}$$

Now, we also have:

$$\begin{aligned}
\|x_{k-1} - x_*\|^2 - \|x_k - x_*\|^2 &= \|x_{k-1} - x_k + x_k - x_*\|^2 - \|x_k - x_*\|^2 \\
&= \|x_{k-1} - x_k\|^2 + 2\langle x_{k-1} - x_k, x_k - x_*\rangle + \|x_k - x_*\|^2 \\
&\quad - \|x_k - x_*\|^2 \\
&= \|x_{k-1} - x_k\|^2 + 2\langle x_{k-1} - x_k, x_k - x_*\rangle,
\end{aligned}$$

hence:

$$\begin{aligned}
\phi(x_k) - \phi(x_*) &\leq \frac{\gamma_k^{-1}}{2}\|x_{k-1} - x_k\|^2 + \gamma_k^{-1}\langle x_{k-1} - x_k, x_k - x_*\rangle \\
&= \frac{\gamma_k^{-1}}{2}\big(\|x_{k-1} - x_k\|^2 + 2\langle x_{k-1} - x_k, x_k - x_*\rangle\big) \\
&= \frac{\gamma_k^{-1}}{2}\big(\|x_{k-1} - x_*\|^2 - \|x_k - x_*\|^2\big).
\end{aligned}$$

□

We are now ready to re-establish in Lemma 12 the well known $\mathcal{O}(\frac{1}{k})$ convergence rate (where k is the number of iterations) of the (non-accelerated) proximal gradient descent algorithm described in Definition 15. More precisely, we focus on the ISTA optimization scheme with constant learning rate (Algorithm 2.12):

Algorithm 2.12 ISTA [Beck and Teboulle, 2009] (constant step)

Input: $\phi = f + g : \mathbb{R}^n \mapsto \mathbb{R}$ convex with global minimizer x_*, where f convex differentiable with L-Lipschitz continuous gradient, g convex; $x_0 \in \mathbb{R}^n$

Output: $x_k \underset{k\to+\infty}{\longrightarrow} x_*$

1: **for** k from 1 to $\dots$ **do**
2: $\quad z_k = x_{k-1} - L^{-1}\nabla f(x_{k-1})$
3: $\quad x_k = \mathcal{P}rox(L^{-1}g)(z_k)$
4: **end for**
5: **return** $(x_k)_k$

Lemma 12. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a convex differentiable function with an L-Lipschitz continuous gradient and $g : \mathbb{R}^n \to \mathbb{R}$ a convex function. Consider $\phi = f + g$, let x_* denote a global minimizer of ϕ and generate the parameters sequence $(x_k)_k$ with Algorithm 2.12. Then the following holds:*

$$\forall k > 0,\ \phi(x_k) - \phi(x_*) \leq \frac{L}{2k}\|x_0 - x_*\|^2. \tag{2.25}$$

Proof. Fix $k > 0$. Applying Lemma 11 for every $j \in [\![1, k]\!]$ yields:

$$\begin{aligned}\forall j \in [\![1, k]\!],\ \phi(x_j) - \phi(x_*) &\leq \frac{\gamma_j^{-1}}{2}\big(\|x_{j-1} - x_*\|^2 - \|x_j - x_*\|^2\big) \\ &= \frac{L}{2}\big(\|x_{j-1} - x_*\|^2 - \|x_j - x_*\|^2\big).\end{aligned}$$

Notice that the right-hand side in the last equation is a telescopic term; additionally, Lemma 10 clearly ensures that $\phi(x_k) \leq \phi(x_j)$ for all $j \in [\![1, k]\!]$. These two remarks

combined let us conclude:

$$\begin{aligned}\phi(x_k) - \phi(x_*) &\leq \frac{1}{k}\sum_{j=1}^{k}\big(\phi(x_j) - \phi(x_*)\big) \\ &\leq \frac{1}{k}\sum_{j=1}^{k}\frac{L}{2}\big(\|x_{j-1} - x_*\|^2 - \|x_j - x_*\|^2\big) \\ &= \frac{L}{2k}\big(\|x_0 - x_*\|^2 - \|x_k - x_*\|^2\big) \\ &\leq \frac{L}{2k}\|x_0 - x_*\|^2.\end{aligned}$$

□

2.8 Accelerated proximal gradient descent

In this brief section we summarize important achievements in terms of developing efficient (both theoretically and practically) optimization schemes for the problem of minimizing composite convex functions.

In a celebrated work, Nesterov [1983] was able to bring the convergence rate of gradient descent (the *smooth* special case of proximal gradient descent) algorithm down from $\mathcal{O}(\frac{1}{k})$ to $\mathcal{O}(\frac{1}{k^2})$ (where k is the number of iterations). Devising accelerated optimization schemes has since become a popular line of research in the optimization community. An other landmark is Beck and Teboulle [2009]'s acceleration scheme for the proximal gradient descent algorithm, achieving the optimal convergence rate of $\mathcal{O}(\frac{1}{k^2})$ in the non-smooth case of composite convex functions with their *fast iterative shrinkage-thresholding algorithm* (FISTA, described in Algorithm 2.13). With a slight modification of FISTA, Chambolle and Dossal [2015] managed to maintain the same (function value) convergence rate while proving the convergence of the parameters sequence, for which nothing was known with FISTA.

An important property that is exploitable to ensure a fast descent is for the objective function to be strongly convex (see Definition 24). Nesterov [2004] gives an optimization scheme that guarantees an optimal linear convergence rate when the objective function is strongly convex; this results was also extended to the non-smooth composite case [Liang et al., 2022] (described in Algorithm 2.14): when the composite objective function is ρ-strongly convex and its smooth component has L-Lipschitz continuous gradient, one achieves the linear convergence rate $\mathcal{O}\left(\left(1 - \sqrt{L^{-1}\rho}\right)^k\right)$. We note that in practice, strong convexity of the problem may be unknown or too expensive to estimate. In such

cases, one can resort instead to using adaptative variants of the FISTA algorithm and/or restarting techniques [Liang et al., 2022; O'Donoghue and Candès, 2015].

Algorithm 2.13 FISTA [Beck and Teboulle, 2009] (constant step)

Input: $\phi = f + g : \mathbb{R}^n \mapsto \mathbb{R}$ convex with global minimizer x_*, where f convex differentiable with L-Lipschitz continuous gradient, g convex; $x_0 \in \mathbb{R}^n$

Output: $x_k \underset{k\to+\infty}{\longrightarrow} x_*$

$t_0 = 1$, $y_0 = x_0$
for k from 1 to ... **do**
  $z_k = y_{k-1} - L^{-1}\nabla f(y_{k-1})$
  $x_k = \mathcal{P}rox(L^{-1}g)(z_k)$
  $t_k = \frac{1+\sqrt{1+4t_{k-1}^2}}{2}$
  $y_k = x_k + \frac{t_{k-1}-1}{t_k}(x_k - x_{k-1})$
end for
return $(x_k)_k$

Algorithm 2.14 Nesterov [Liang et al., 2022; Nesterov, 2004] (constant step)

Input: $\phi = f + g : \mathbb{R}^n \mapsto \mathbb{R}$ ρ-strongly convex with global minimizer x_*, where f convex differentiable with L-Lipschitz continuous gradient, g convex; $x_0 \in \mathbb{R}^n$

Output: $x_k \underset{k\to+\infty}{\longrightarrow} x_*$

$t_0 = 1$, $y_0 = x_0$
for k from 1 to ... **do**
  $z_k = y_{k-1} - L^{-1}\nabla f(y_{k-1})$
  $x_k = \mathcal{P}rox(L^{-1}g)(z_k)$
  $y_k = x_k + \frac{\sqrt{L}-\sqrt{\rho}}{\sqrt{L}+\sqrt{\rho}}(x_k - x_{k-1})$
end for
return $(x_k)_k$

The superiority of accelerated proximal gradient descent over its non-accelerated counterpart and gradient-based methods when minimizing composite convex functions (with a non-smooth component) was empirically validated in Figure 2.9.

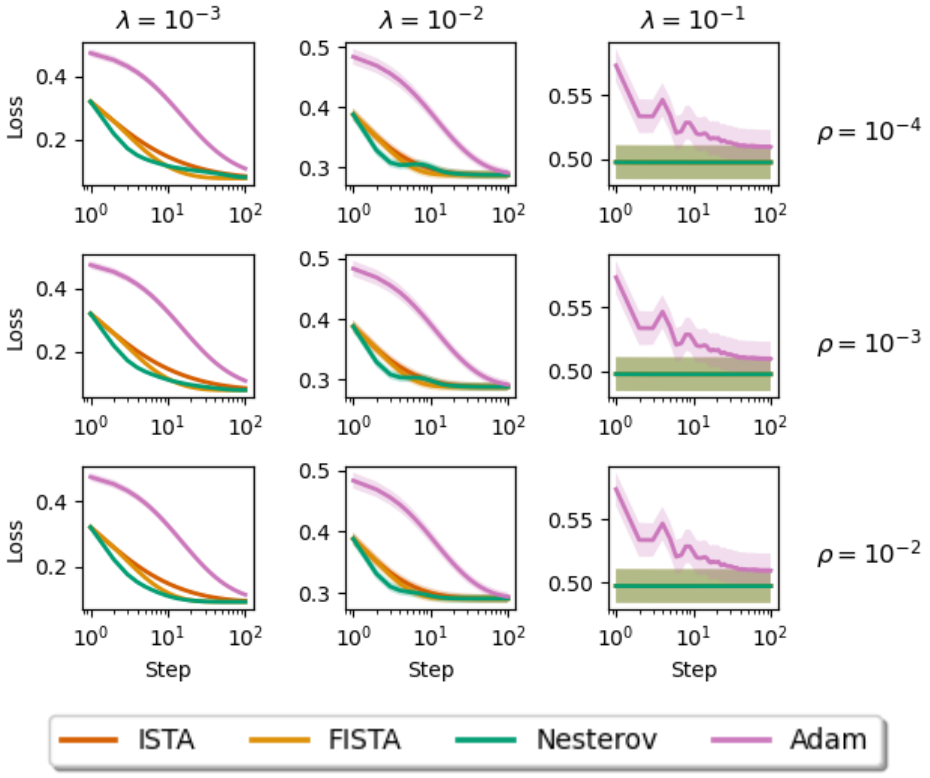

Figure 2.9: Visualization of various optimizers used to minimize composite convex functions of the form $\phi : w \in \mathbb{R}^d \mapsto \frac{1}{2n}\|Aw - b\|^2 + \lambda\|w\|_1 + \frac{\rho}{2}\|w\|^2$, where $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$ were randomly generated from Gaussian noise (50 instances). The displayed loss is averaged across instances and the variance is represented by the shaded regions. The hyperparameter λ controls how *non-smooth* the functions are at 0 and the hyperparameter ρ controls the degree of strong convexity of the functions. Tested solvers include the (non-accelerated) convex optimizer ISTA (Algorithm 2.12) as well as the accelerated convex optimizers FISTA (Algorithm 2.13) and Nesterov (Algorithm 2.14, designed to take advantage of the strong convexity property); the gradient-based adaptative optimizer Adam was also tested. Adam is always slower than convex optimizers at minimizing this class of function and fails at minimizing non-smooth functions (when λ is large); ISTA is always outperformed by its accelerated variants; FISTA is consistently fast, but Nesterov is the best alternative when strong convexity is important (when ρ is large).

Chapter 3

Scalable learning of BNs using FAS-based heuristics

In this short chapter, we summarize the main contributions of this thesis in the form of two different heuristics - **BNSL2MAS** and **ProxiMAS** - tailored for Bayesian network structure learning at large scale, both revolving around the use of maximum acyclic subgraph/minimum feedback arc set problems. The purpose of the section is to give a high level overview of these heuristics while providing more context about how they fit within the mathematical notions encountered in Chapter 2.

Article	**Conference**	**Heuristic**
I *Scalable Bayesian Network Structure Learning via Maximum Acyclic Subgraph*	PGM 2020	BNSL2MAS
II *Learning Large DAGs by Combining Continuous Optimization and Feedback Arc Set Heuristics*	AAAI 2022	ProxiMAS
III *Convergence of Feedback Arc Set-Based Heuristics for Linear Structural Equation Models*	PGM 2022	ProxiMAS

Table 3.1: Heuristics developed in this thesis and corresponding article(s).

3.1 BNSL2MAS (Article I)

In a seminal paper, Cussens [2011] proposed an ILP-based exact solver for the score-based structure learning problem, where for every node-parent set pair (having a local score) a binary variable is created and this variable takes value 1 if and only if the corresponding node-parent set pair is in the solution. Although mathematically elegant and easy to formalize, the large amount of binary variables (see Table 3.2) limits the

practical scalability of this approach to at most a few hundred nodes and only a few parents per node.

Scores set	**Cardinality w.r.t. # nodes d**
Local scores	$\mathcal{O}(d2^{d-1})$
Local scores (parent sets size $\leq k$)	$\mathcal{O}(d^{k+1})$
Arc scores	$\mathcal{O}(d^2)$

Table 3.2: Worst case cardinality of different sets of scores used in Bayesian network structure learning. The cardinality of local scores grows exponentially in the number of nodes, which is why restrictions on the number of parents are typically used in score-based structure learning. The cardinality of arc scores on the other hand is at most quadratic, independently of parent sets size.

The BNSL2MAS heuristic introduced in Article I aims at addressing the aforementioned scalability limitations and consists of two steps:

1. Local scores are *compactified* into a set of arc scores (always quadratic in cardinality), under the (strong) assumption that local scores are *additive* in nature: contributions from individual parents are assumed to add-up together to form local scores with parent sets of any size. This compactification step occurs in Algorithm 3.2: line 1 and relies on Algorithm 3.1.

2. Given the weighted digraph constructed from the AALS (in Algorithm 3.2: line 2), a maximum acyclic subgraph problem is solved exactly for that instance (this MAS step occurs in Algorithm 3.2: line 3).

Although the difficulty in enforcing acyclicity remains, the reduced number of binary variable effectively makes ILP MAS problems scale more favorably than ILP BNSL problems and this holds especially true when the maximum number of parents is large. The determinant factor in ensuring that BNSL2MAS scales itself favorably thus lies in an efficient computation of the AALS. Looking more closely, we note that the optimization problems being solved at every iteration of the outer for loop (in Algorithm 3.1: line 6) are in fact independent of one another, such that the corresponding models can be initialized and solved in parallel, increasing the scalability of the method. Common choices for the distance $\mathcal{D}$ passed as input in Algorithm 3.1 are the absolute loss $(x, y) \mapsto |x-y|$ and the squared loss $(x, y) \mapsto (x-y)^2$, resulting in linear optimization problems for the former and quadratic optimization problems for the latter.

Algorithm 3.1 AALS

Input: Local scores $\mathcal{S} = \bigcup_{j=0}^{d-1} \mathcal{S}_j$ (see Definition 1), distance $\mathcal{D}$

Output: Approximate additive local scores

1: **for** j in $[\![0, d-1]\!]$ **do**
2: Create continuous **bias** variable b_j
3: **for** i in $[\![0, d-1]\!] \setminus \{j\}$ **do**
4: Create continuous **arc score** variable $s_{i,j}$
5: **end for**
6: Solve the continuous optimization problem:

$$b_j^*, \{s_{i,j}^*\}_i = \underset{b_j, \{s_{i,j}\}_i}{\operatorname{argmin}} \sum_{I:\, S(I,j) \in \mathcal{S}_j} \mathcal{D}\big(S(I,j), \widehat{S}(I,j)\big)$$

$$\text{s.t. } \forall I : S(I,j) \in \mathcal{S}_j, \quad \begin{cases} \widehat{S}(I,j) = b_j + \sum_{i \in I} s_{i,j} \\ \widehat{S}(I,j) \geq S(I,j) \end{cases}$$

7: **end for**
8: **return** $\{b_j^*\}_j, \{s_{i,j}^*\}_{i,j}$

Algorithm 3.2 BNSL2MAS

Input: Local scores $\mathcal{S} = \bigcup_{j=0}^{d-1} \mathcal{S}_j$ (see Definition 1), distance $\mathcal{D}$

Output: Approximate BNSL solution, upper bound on the optimal BNSL score

1: Extract AALS: $\{b_j^*\}_j, \{s_{i,j}^*\}_{i,j} =$ Algorithm 3.1$(\mathcal{S}, \mathcal{D})$
2: Convert arc scores $\{s_{i,j}^*\}_{i,j}$ into a weighted digraph $\widetilde{\mathcal{G}}$
3: Solve exactly the weighted MAS problem given $\widetilde{\mathcal{G}}$ as input: $\overline{\mathcal{G}} = \mathrm{MAS}\big(\widetilde{\mathcal{G}}\big)$
4: Compute BNSL score upper bound: $\overline{S}_{\mathrm{BNSL}} = \sum_{j=0}^{d-1} b_j^* + \sum_{(i,j) \in \overline{\mathcal{G}}} s_{i,j}^*$
5: Extract best subset from $\overline{\mathcal{G}}$ w.r.t. $\mathcal{S}$:

$$\mathcal{G} = \underset{\bigotimes_{j=0}^{d-1} \{I_j:\, S(I_j,j) \in \mathcal{S}_j\}}{\operatorname{argmax}} \sum_{j=0}^{d-1} S(I_j, j)$$

$$\text{s.t. } \bigcup_{j=0}^{d-1} (I_j \to j) \subset \overline{\mathcal{G}}$$

6: **return** $\mathcal{G}, \overline{S}_{\mathrm{BNSL}}$

On the theoretical side, we proved in Article I that enforcing the constraints that AALS upper-bound true local scores is sufficient to ensure that the sum of arc weights in the optimal MAS solution obtained via BNSL2MAS provides an upper bound on the optimal

BNSL score (see Algorithm 3.2: line 4). On the practical side, the obtained optimal MAS solution tends to be overly dense but it is trivial to extract from it a subset (necessarily acyclic) with the highest possible BNSL score (this is achieved in Algorithm 3.2: line 5). Based on the empirical study in Article I, the resulting approximate BNSL solutions are comparable in quality to solutions returned by hill-climbing heuristics, limiting the attractiveness of the BNSL2MAS approach on the qualitative side.

3.2 OptiMAS and ProxiMAS (Articles II, III)

A crucial difference between general BNSL problems and MAS problems is that the latter's scoring function scores arcs whereas the former's scoring function scores node-parent set pairs which form a much richer set. This partly explains the difficulty in obtaining good BNSL solutions using the BNSL2MAS heuristic. The question then becomes: *Can we find a less general framework in which structure learning and MAS explore the space of DAGs using a similar scoring function, enabling us to bridge the two problems more efficiently ?*

The most simplistic framework satisfying this requirement is the so-called linear structural equation models (abbreviated linear SEMs) setting, in which one seeks to recover from data the structure of a continuous Bayesian network, where nodes are assumed to be linearly dependent on their parents plus some noise (see Figure 3.1): this is a classical example of a linear additive noise model.

An important question in structure learning concerns *identifiability*: *In which case is it possible to learn the true DAG from observational data only, that is without interventional experiments ?* Many results have been found in the case of continuous structure recovery: while non-linear additive noise models are known to be identifiable [Peters et al., 2011], linear additive models are not identifiable in general, with linear non-Gaussian models being identifiable [Shimizu et al., 2006] and linear Gaussian models requiring extra conditions (e.g. equal variance [Peters and Bühlmann, 2013] or approximately known variance [Loh and Bühlmann, 2014]) to become identifiable.

From an optimization perspective, given a data matrix $X \in \mathbb{R}^{n \times d}$ consisting of n samples obtained from a continuous Bayesian network with d nodes, linear SEMs structure recovery corresponds to the following minimization problem:

$$\begin{aligned} W_* \;=\; & \underset{W \in \mathbb{R}^{d \times d}}{\operatorname{argmin}} \tfrac{1}{2n} ||XW - X||^2 + \lambda g(W) \\ & \text{s.t. } W \text{ represents a DAG} \end{aligned} \tag{3.1}$$

where W is the weighted adjacency matrix of the DAG, g is a regularizer (usually enforcing sparsity) tuned by the hyperparameter λ and the quadratic term corresponds to the linear SEMs fitness and represents aforementioned linear dependencies between random variables. We note that using least squares for the fitness term is not mandatory: one can for instance use the (negative) log-likelihood instead [Ng et al., 2020] (under some assumption on the noise distribution).

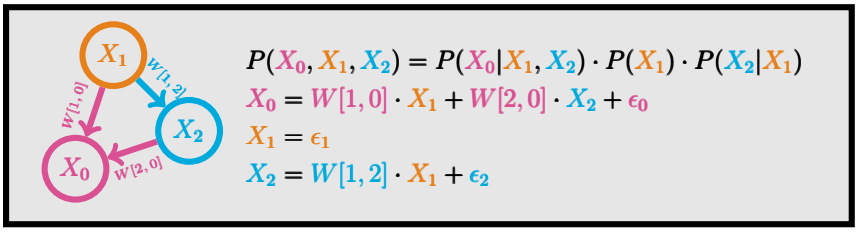

Figure 3.1: Illustration of linear structural equation models: in linear SEMs, random variables are continuous and every node is expressed as a linear combination of its parents plus some noise (represented by ϵ in the equations). Linear SEMs structure recovery is the problem of finding optimal arc weights $\big(W[i,j]\big)_{(i,j)\in E}$, that is finding the optimal weighted adjacency matrix W, under the constraint that W represents a DAG.

Solving optimally the minimization problem in Equation 3.1 is particularly challenging as it involves solving a mixed-integer quadratic program [Lazimy, 1982], where a quadratic objective function is being minimized while integer linear constraints are added to the model in order to enforce acyclicity much in the same way as in ILPs for MAS (see Section 2.5). Such coupling limits greatly the scalability of exact linear SEMs structure recovery in practice [Manzour et al., 2021], but a recent breakthrough gave a push on the practical side: Zheng et al. [2018] formulated the acyclicity constraint using a smooth function, effectively transferring the combinatorial nature of structure learning into a pure gradient-based optimization procedure. Although the resulting optimization problem remains highly non-convex, they managed to empirically validate their method when learning small-scale DAGs. Unfortunately, large scale learning remains out of reach with this approach as well, owing to the cubic complexity of the introduced acyclicity function (which involves computing a matrix exponential).

The OptiMAS heuristic introduced in Article II tries to overcome these difficulties in an attempt to achieve better scalability: first by decoupling the minimization of the loss from acyclicity enforcement; second by replacing the smooth acyclicity function [Zheng et al., 2018] by a quadratic acyclicity regularization term acting as a proxy that is much easier to compute and differentiate. More precisely, this works by constructing in an *online* fashion a sequence of loss functions with *stationary* properties (composite form, strong convexity and *smoothness*, see Lemma 41) that can be exploited, each loss differing only in a quadratic term that contains structural information of the previously

discovered DAG. At every iteration, the heuristic:

1. Makes an unconstrained optimization step on a newly constructed loss function (Algorithm 3.3: line 4).

2. Converts the obtained (cyclic) solution into a *close* acyclic solution by approximately solving a weighted maximum acyclic subgraph/minimum feedback arc set problem using a greedy heuristic (Algorithm 3.3: line 5, see Section 2.6 for details on greedy MAS/FAS heuristics).

3. Constructs a new loss function by adding the structural information of the previously found acyclic solution to the linear SEMs loss (Algorithm 3.3: line 3).

The scalability potential of this process was empirically established in Article II.

ProxiMAS corresponds to Algorithm 3.3 where the optimizer passed as input is a proximal gradient descent optimizer (for instance, one of Algorithms 2.12-2.14), effectively exploiting the composite convex form of the sequence of objective functions (see Algorithm 3.5: line 4). In Article III, a theoretical convergence analysis of the ProxiMAS algorithm was conducted assuming:

- The optimizer is the non-accelerated proximal gradient descent algorithm ISTA (Algorithm 2.12).

- The embedded weighted maximum acyclic subgraph greedy heuristic is the inscore-based weighted FAS variant (Algorithm 2.4 combined with a weighted version of Algorithm 2.11 where the projection corner is set to "bottom-left"), with the particularity that squared arc weights are used by Algorithm 2.4 in order to construct a permutation (rather than regular arc weights); this greedy MAS heuristic is summarized in Algorithm 3.4.

The ProxiMAS variant analyzed in Article III is detailed in Algorithm 3.5. The complete ProxiMAS stability proof from Article III can be found in Appendix (Section 5.7), where by stability, we mean here that ProxiMAS eventually constructs the same permutation with the embedded weighted MAS greedy heuristic, ensuring the same acyclic structures will be found after enough iterations.

Algorithm 3.3 OptiMAS

Input: $X \in \mathbb{R}^{n\times d}$, $\rho > 0$, $\lambda > 0$, optimizer

Output: Sequence of acyclic weighted adjacency matrices $(W_k)_k$

1: $W_0, \widetilde{W}_0 = \text{zeros}(d, d)$

2: **for** k from 1 to $\dots$ **do**

3: New objective function:

$$\phi_k : W \in \mathbb{R}^{d\times d} \mapsto \tfrac{1}{2n}\|XW - X\|^2 + \tfrac{\rho}{2}\|W - W_{k-1}\|^2 + \lambda\|W\|_1$$

4: Optimization step: $\widetilde{W}_k = \text{step}(\phi_k,\ \text{optimizer})$

5: MAS projection: $W_k = \text{weighted MAS}\big(\widetilde{W}_k\big)$

6: **end for**

7: **return** $(W_k)_k$

Algorithm 3.4 Greedy square-weighted MAS

Input: $\mathcal{G} = (V, E, s) :\ V = [\![0, d-1]\!]$ with weighted adjacency matrix $\widetilde{W}$

Output: Weighted acyclic projection W of $\widetilde{W}$ approximately maximizing $\|W\|^2$

1: $V_1 = V$

2: **for** r from 1 to d **do**

3: Greedy node selection based on squared weights:

$$\begin{aligned} v_r &= \operatorname*{argmin}_{j\in V_r} \sum_{i\in V_r\setminus\{j\}} \big(s(i,j)\big)^2 \\ &= \operatorname*{argmin}_{j\in V_r} \sum_{i\in V_r\setminus\{j\}} \big(\widetilde{W}[i,j]\big)^2 \\ \Big(&= \operatorname*{argmin}_{j\in V_r} \big\|\widetilde{W}\big[V_r\setminus\{j\}, j\big]\big\|^2 \text{ in vectorized form}\Big) \end{aligned}$$

4: $\pi : v_r \mapsto d - r \quad \big(\pi[d-r] = v_r \text{ in vectorized form}\big)$

5: $V_{r+1} = V_r \setminus \{v_r\}$

6: **end for**

7: $T = \big(\widetilde{W}[\pi^{-1}(i), \pi^{-1}(j)]\big)_{(i,j)\in[\![0,d-1]\!]^2}$

8: $\forall i \le j :$ set $T[i,j] = 0$

9: $W = \big(T[\pi(i), \pi(j)]\big)_{(i,j)\in[\![0,d-1]\!]^2}$

10: **return** W

Algorithm 3.5 ProxiMAS (analyzed in Article III)

Input: $X \in \mathbb{R}^{n\times d}$, $\rho > 0$, $\lambda > 0$

Output: Sequence of acyclic weighted adjacency matrices $(W_k)_k$

1: $W_0, \widetilde{W}_0 = \text{zeros}(d, d)$

2: Preprocessing (see Lemma 41): $L = \frac{1}{n}\|X^tX + n\rho I\|_*$

3: **for** k from 1 to ... **do**

4: New objective function:

$$\phi_k : W \in \mathbb{R}^{d\times d} \mapsto \underbrace{\overbrace{\frac{1}{2n}\|XW - X\|^2}^{\text{linear SEMs fitness}} + \overbrace{\frac{\rho}{2}\|W - W_{k-1}\|^2}^{\text{acyclicity regularizer}}}_{=: f_k \text{ (smooth, convex)}} + \underbrace{\overbrace{\lambda\|W\|_1}^{\text{sparsity regularizer}}}_{=: g \text{ (non-smooth, convex)}}$$

5: Non-accelerated proximal gradient descent step (see Definition 15):

i Update learning rate $\gamma_k \in]0, L^{-1}]$, then compute:

$$\begin{aligned} Z_k &= \widetilde{W}_{k-1} - \gamma_k \nabla f_k\big(\widetilde{W}_{k-1}\big) \\ &= \widetilde{W}_{k-1} - \gamma_k\Big(\frac{1}{n}X^t\big(X\widetilde{W}_{k-1} - X\big) + \rho\big(\widetilde{W}_{k-1} - W_{k-1}\big)\Big) \end{aligned}$$

ii Update parameters (apply Lemma 37):

$$\begin{aligned} \widetilde{W}_k &= \mathcal{P}rox(\gamma_k g)(Z_k) \\ &= \mathcal{P}rox\big(\gamma_k\lambda\|\cdot\|_1\big)(Z_k) \\ &= \Big(\mathcal{P}rox\big(\gamma_k\lambda|\cdot|\big)\big(Z_k[i,j]\big)\Big)_{(i,j)\in[\![0,d-1]\!]^2} \end{aligned}$$

i.e. $\forall (i,j) \in [\![0, d-1]\!]^2$:

$$\widetilde{W}_k[i,j] = \begin{cases} \Big(\big|Z_k[i,j]\big| - \gamma_k\lambda\Big) \cdot \frac{Z_k[i,j]}{\big|Z_k[i,j]\big|} & \text{if } \big|Z_k[i,j]\big| > \gamma_k\lambda \\ 0 & \text{if } \big|Z_k[i,j]\big| \leq \gamma_k\lambda \end{cases}$$

6: MAS projection: $W_k = \text{Algorithm 3.4}\big(\widetilde{W}_k\big)$

7: **end for**

8: **return** $(W_k)_k$

Chapter 4

Conclusion

In this thesis we tackled the problem of Bayesian network structure learning viewed under the score-based standpoint. More specifically, the goal was to develop scalable heuristics addressing this challenging problem, for which exact methods (traditionally relying on dynamic programming or integer linear programming) seldom scale beyond a few hundred nodes in practice and usually necessitate strong structural restrictions such as a small limit on the maximum number of parents in order to remain tractable.

A recurring theme in the research conducted for this thesis is the integration of maximum acyclic subgraph/minimum feedback arc set problems as key components of heuristics specifically designed with the purpose of approximately solving Bayesian network structure learning problem at significantly larger scale (beyond thousand nodes). Two such heuristics were developed in this thesis:

- **BNSL2MAS** (Article I) is a 2-step ILP-based heuristic for general BNSL, which 1. transforms a potentially very large set of local scores into a significantly more compact set of arc scores (of quadratic size) by solving a sequence of independent linear or quadratic problems (one per node in the DAG), these arc scores are then encapsulated in a weighted digraph; 2. solves exactly a weighted maximum acyclic subgraph problem given the constructed weighted digraph as input. We proved that the MAS score of an exact MAS solution of the aforementioned weighted MAS instance provides an upper-bound on the optimal BNSL score. Moreover, from the computed exact MAS solution, the best possible subgraph (with respect to the BNSL score function) can be trivially extracted, leading in practice to solutions comparable in quality to those returned by a hill-climbing heuristic.
- **ProxiMAS** (Articles II, III) is a non-convex optimization scheme for linear SEMs structure recovery. It is based on the idea that optimizing the loss function and

enforcing acyclicity should be decoupled in order to achieve superior scalability. It achieves this goal by repeating an unconstrained (acyclicity-wise) optimization step followed by an acyclic projection of the current cyclic solution using a greedy weighted maximum acyclic subgraph heuristic. The scalability potential of this process was empirically established in Article II and a theoretical analysis of the stability of the method was conducted in Article III. On the empirical side, Article III also demonstrates that combining a convex optimizer and an adaptative gradient-based non-convex optimizer can dramatically speedup convergence in practice (in the sense that less iterations are needed before the heuristic stabilizes).

Developing more scalable methods for Bayesian network structure learning has become a recent focus in the field [Aragam et al., 2017; Dong and Sebag, 2022; Scanagatta et al., 2015; Yu et al., 2021; Zhu et al., 2021], and both BNSL2MAS and ProxiMAS contribute to that endeavor: the former (BNSL2MAS) exploits the *compactness* of ILPs for MAS compared to ILPs for BNSL and BNSL2MAS can be applied to learning significantly larger and denser DAGs compared to state-of-the-art exact BNSL solvers; the latter (ProxiMAS) uses a *trick* (alternating proximal gradient descent steps and approximate MAS projections) such that one no longer needs to rely on the computation and differentiation of a costly smooth acyclicity function [Zheng et al., 2018] in order to explore acyclic structures. Besides, the ability to partially vectorize greedy MAS heuristics (see Section 2.6) means one can run ProxiMAS on a GPU and achieve significant speed-up (we note that the proximal gradient descent step itself is embarrassingly parallel, owing to the fully vectorizable closed-form solution of the proximal operator derived in Lemma 37). The low memory footprint of ProxiMAS means even very large DAGs can be learned using a GPU with this heuristic, and that medium-to-large networks can be learned even using a laptop with a low-end GPU, significantly streamlining the process of learning DAGs at large scale.

The methods presented in this thesis are not without shortcoming: on the one hand, in its current state BNSL2MAS relies on the computation of an exact solution for a weighted MAS problem and therefore the scalability of this heuristic is at the moment tied to the scalability of solving the *lazy cycle set cover* ILP implementation of MAS, i.e. this scalability is highly dependent on the input weighted digraph due to the fact that solving optimally integer programs is NP-hard. Additionally, the upper-bound returned by BNSL2MAS, while theoretically valid, is rarely useful in practice based on the experiments conducted in Article I. Worse: our attempts at tightening this upper bound led to largely reduced scalability, defying the purpose of the approach; on the other hand, while ProxiMAS provides an *optimistic* scalability scenario when learning DAGs assuming linear dependencies between (continuous) random variables (linear SEMs setting),

real world data typically involve a degree of non-linearity which ProxiMAS would fail to capture properly in its current design. Besides, although we could formulate theoretical understanding regarding the stability of ProxiMAS (the optimization point of view), currently there is no theoretical understanding on which acyclic structures are being explored by the heuristic and ultimately on the quality of found DAGs (the machine learning point of view), i.e. one could argue ProxiMAS is merely a useful *black box* for approximately learning large and sparse DAGs.

Based on the research conducted in the context of this thesis, it appears unlikely that feedback arc set-based heuristics for learning DAGs will become the dominant strategy for general use, although they constitute an elegant approach to large scale learning. This approach was left - to the best of our knowledge - surprisingly unexplored (with the exception of the recent work from Park and Klabjan [2017]) and shows potential in terms of practical scalability, although we found difficulties in achieving theoretical understanding. We believe key aspects of the proposed methods can still be improved: BNSL2MAS currently solves exactly a single weighted MAS instance, such that one may want instead to sequentially solve approximately several weighted MAS instances, although the process by which these MAS instances should evolve in the sequence is not yet understood. Generalizing ProxiMAS to handle non-linearity is straightforward assuming non-linearities take the form of linear combinations of smooth non-linear functions, yet it is unclear how this would perform in practice (especially at large scale). A third interesting route would be to replace the L1 norm by the L0 norm in order to enforce sparsity in ProxiMAS, motivated by the fact that L0 regularization gives the same score to Markov equivalent DAGs, which is not true of L1 regularization. Unlike the L1 norm, the L0 norm is non-convex such that clever adjustments of ProxiMAS would be needed. A recent contribution [Nhat et al., 2018] in the context of difference of convex functions optimization showed how accelerated optimizers could be designed for that class of problems, with an application to approximately minimizing the L0 norm. It would then be interesting to replace the accelerated proximal gradient descent optimizer embedded in ProxiMAS by an accelerated difference of convex functions optimizer [Nhat et al., 2018], assess the cost on scalability and potential gain in terms of the quality of found sparse solutions.

Chapter 5

Appendix

This chapter offers complementary knowledge that should prove useful in order to fully grasp the concepts behind the ProxiMAS heuristic (Section 3.2) developed in this thesis:

- Sections 5.1-5.4 provide the necessary notions from convex analysis for the theoretical analysis of the proximal gradient descent optimization scheme presented in Section 2.7.
- Sections 5.5, 5.6 use analysis and calculus in order to better understand the way proximal gradient descent is integrated into the ProxiMAS heuristic.
- Section 5.7 gives the full proof of a theorem we proposed in Article III regarding the stability of the ProxiMAS heuristic.

5.1 Basics on convexity

In this section, the reader is reminded of key knowledge surrounding the notion of *convexity*. This property exists both for sets and functions and both have a geometrical interpretation as illustrated in Figures 5.1, 5.3, respectively. A fundamental consequence of convexity is the so-called *Jensen property* (Lemma 16).

Additionally, topological properties of convex sets are established (Lemmas 13, 14). *Convex hulls* form an important family of convex sets characterized in Lemma 17; a topological result for convex hulls is given in Lemma 18. These topological notions will come in handy when working with more advanced tools from convex analysis later in the Appendix (particularly in Section 5.2).

Definition 16. *Let $\Omega \subset \mathbb{R}^n$. We say Ω is a* ***convex (sub)set (of $\mathbb{R}^n$)*** *if:*

$$\forall (x, y) \in \Omega^2, \forall t \in [0, 1]\,,\ tx + (1-t)y \in \Omega. \tag{5.1}$$

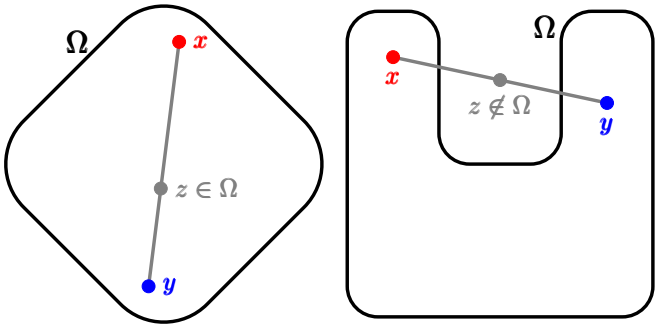

Figure 5.1: Representation of a convex set (left) and a non-convex set (right). Geometrically, a convex set contains all segments between two points in the set.

Lemma 13. *Let $\Omega \subset \mathbb{R}^n$ a convex set. If $\overset{\circ}{\Omega}$ is non-empty, then $\overset{\circ}{\Omega}$ is a convex set.*

Proof. Fix $(x, y) \in \left(\overset{\circ}{\Omega}\right)^2$ and $t \in [0, 1]$. We want to show that $tx + (1-t)y \in \overset{\circ}{\Omega}$. By definition of the interior, $\overset{\circ}{\Omega}$ is the union of all open subsets of Ω thus it suffices to identify an open subset of Ω containing $tx + (1-t)y$. Now, since $(x, y) \in \left(\overset{\circ}{\Omega}\right)^2$ we know:

$$\begin{cases} \exists r_x > 0 : & B(x, r_x) \subset \Omega \\ \exists r_y > 0 : & B(y, r_y) \subset \Omega. \end{cases}$$

We prove that the set $S \coloneqq \bigcup_{t' \in [0,1]} \left\{ t' B(x, r_x) + (1-t') B(y, r_y) \right\}$ works:

- S contains $tx + (1-t)y$. This follows trivially from the definition of S.
- S is a subset of Ω. Indeed, we have $B(x, r_x) \subset \Omega$ and $B(y, r_y) \subset \Omega$ where Ω is a convex set, hence $t'x' + (1-t')y' \in \Omega$ for all $(x', y') \in B(x, r_x) \times B(y, r_y)$ and $t' \in [0, 1]$.
- S is open. To see this fix $t' \in]0, 1[$, then notice $t' B(x, r_x) = B(t'x, t'r_x)$ and $(1-t')B(y, r_y) = B\big((1-t')y, (1-t')r_y\big)$; the sum of two open sets remains open, so the set $B(t'x, t'r_x) + B\big((1-t')y, (1-t')r_y\big)$ is open. We can therefore write S as a union of open sets, proving S is open:

$$S = B(x, r_x) \cup B(y, r_y) \cup \bigcup_{t' \in]0,1[} \left\{ B(t'x, t'r_x) + B\big((1-t')y, (1-t')r_y\big) \right\}.$$

□

Lemma 14. *Let $\Omega \subset \mathbb{R}^n$ a convex set. If $\overset{\circ}{\Omega}$ is non-empty, then $\overline{\Omega} = \overline{\overset{\circ}{\Omega}}$.*

Proof. The inclusion $\overline{\Omega} \supset \overline{\overset{\circ}{\Omega}}$ is clear since $\Omega \supset \overset{\circ}{\Omega}$, thus we simply need to show that $\overline{\Omega} \subset \overline{\overset{\circ}{\Omega}}$. Noting that the boundary of Ω is defined as $\partial\Omega \coloneqq \overline{\Omega} \backslash \overset{\circ}{\Omega}$, we then have $\overline{\Omega} = \overset{\circ}{\Omega} \cup \partial\Omega$ and the claim reduces to $\partial\Omega \subset \overline{\overset{\circ}{\Omega}}$. Now fix $x \in \partial\Omega$ and pick $y \in \overset{\circ}{\Omega}$ (y must exist by hypothesis). Our strategy will be to prove the set inclusion $\bigcup_{t\in]0,1[} \{tx + (1-t)y\} \subset \overset{\circ}{\Omega}$; $x \in \overline{\overset{\circ}{\Omega}}$ will follow. For that purpose, fix $z \coloneqq tx + (1-t)y$ where $t \in]0,1[$; in order to show that $z \in \overset{\circ}{\Omega}$, we will identify an open subset of Ω containing z. Let us first remember that $y \in \overset{\circ}{\Omega}$, i.e. there is $r_y > 0$ such that $B(y, r_y) \subset \Omega$. Now, consider the following mapping:

$$f_z : y' \mapsto \frac{z - (1-t)y'}{t}.$$

It is straightforward to show that f_z is a bijection from the open ball $B(y, r_y)$ to the open ball $B(x, r_x)$, where $r_x \coloneqq \frac{1-t}{t} r_y$. From a geometric standpoint, f_z maps every point y' in $B(y, r_y)$ to a unique corresponding point x' in $B(x, r_x)$ in such a way that the segment $[x', y']$ *passes through* z $\big($formally, one has $z = tx' + (1-t)y'\big)$. Then, since $x \in \partial\Omega$ we deduce that Ω and $B(x, r_x)$ necessarily intersect i.e. we can pick $w \in \Omega \cap B(x, r_x)$. We now can identify the desired set. We show $S \coloneqq \bigcup_{t'\in[0,1[} \big\{t'\{w\} + (1-t')B(y, r_y)\big\}$ works:

- S contains z. This follows from the construction of the mapping f_z: w is in the ball $B(x, r_x)$ and is therefore uniquely mapped to $f_z^{-1}(w) \in B(y, r_y)$ in such a way that $z = tw + (1-t)f_z^{-1}(w)$ holds.
- S is a subset of Ω. Indeed, we have $w \in \Omega$ and $B(y, r_y) \subset \Omega$ where Ω is a convex set, hence $t'w + (1-t')y' \in \Omega$ for all $y' \in B(y, r_y)$ and $t' \in [0, 1[$.
- S is open. To see this fix $t' \in [0, 1[$, then notice one has the following set equality: $t'\{w\} + (1-t')B(y, r_y) = B\big(t'w + (1-t')y, (1-t')r_y\big)$. We can therefore write S as a union of open sets, proving S is open:

$$S = \bigcup_{t'\in[0,1[} B\big(t'w + (1-t')y, (1-t')r_y\big).$$

□

Definition 17. *Let $f : \Omega \to \mathbb{R} \cup \{+\infty\}$ a function defined on a convex set $\Omega \subset \mathbb{R}^n$. We say f is* ***convex on Ω*** *if:*

$$\forall (x, y) \in \Omega^2, \forall t \in [0, 1]\,,\ f\big(tx + (1-t)y\big) \leq tf(x) + (1-t)f(y). \tag{5.2}$$

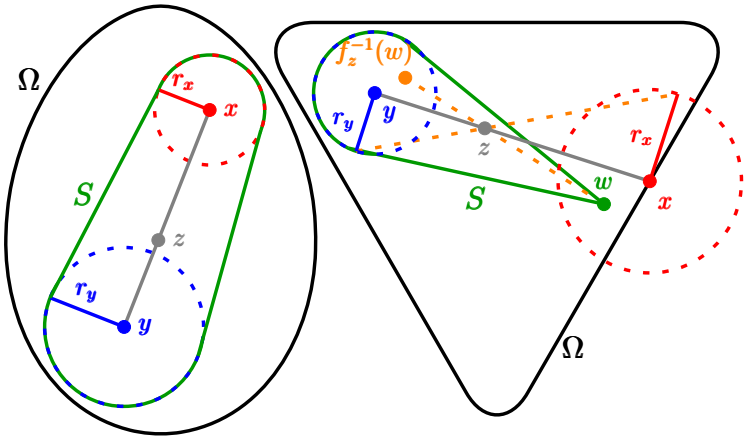

Figure 5.2: Illustration of Lemmas 13, 14: construction of an open subset S of the convex set $\Omega \subset \mathbb{R}^n$ such that S contains $z \in]x, y[$ (left: Lemma 13 where $(x, y) \in \left(\overset{\circ}{\Omega}\right)^2$; right: Lemma 14 where $(x, y) \in \partial\Omega \times \overset{\circ}{\Omega}$). On the left, the same open set S is valid for all $z \in]x, y[$ whereas on the right, the open set S is constructed for $z \in]x, y[$ fixed.

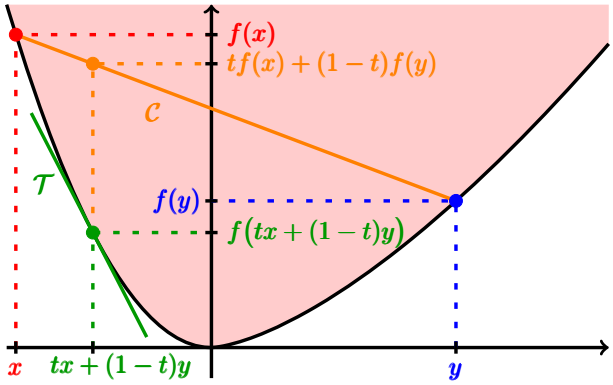

Figure 5.3: Representation of a convex function. The geometrical interpretation of convexity is that chords between two points in the graph of a convex function (represented by the orange line $\mathcal{C}$) always lie above the graph; equivalently, the epigraph of a convex function (represented by the pink surface) is a convex set (see Lemma 23). Another geometrical interpretation is that hyperplanes tangential to the graph of a convex function (represented by the green line $\mathcal{T}$) always lie below the graph (see Lemma 26).

Lemma 15. *Let $\Omega \subset \mathbb{R}^n$ a convex set, $f_1, \ldots, f_k : \Omega \to \mathbb{R} \cup \{+\infty\}$ k functions convex on Ω and $\alpha_1, \ldots, \alpha_k > 0$. Then the function $\sum_{j=1}^{k} \alpha_j f_j$ is convex on Ω.*

Proof. For convenience, let us write $f \coloneqq \sum_{j=1}^{k} \alpha_j f_j$. Fix $(x, y) \in \Omega^2$ and $t \in [0, 1]$. Due to the convexity of the f_j on Ω we have:

$$\forall j \in [\![1, k]\!],\ f_j\big(tx + (1-t)y\big) \leq t f_j(x) + (1-t) f_j(y).$$

The previous property combined with the positivity of the α_j then yields:

$$\begin{aligned} f\big(tx + (1-t)y\big) &= \sum_{j=1}^{k} \alpha_j f_j\big(tx + (1-t)y\big) \\ &\leq \sum_{j=1}^{k} \alpha_j \big(t f_j(x) + (1-t) f_j(y)\big) \\ &= t \cdot \sum_{j=1}^{k} \alpha_j f_j(x) + (1-t) \cdot \sum_{j=1}^{k} \alpha_j f_j(y) \\ &= t f(x) + (1-t) f(y), \end{aligned}$$

which proves that f is convex on Ω. □

Lemma 16. *Let $\Omega \subset \mathbb{R}^n$ a convex set and $f : \Omega \to \mathbb{R} \cup \{+\infty\}$ a function convex on Ω. Then for all $k \in \mathbb{N}^*$, for all $(x_j)_{j \in [\![1,k]\!]} \in \Omega^k$, for all $(t_j)_{j \in [\![1,k]\!]} \in [0, 1]^k$ satisfying $\sum_{j=1}^{k} t_j = 1$, the following holds:*

$$\sum_{j=1}^{k} t_j x_j \in \Omega \quad \textit{and} \quad f\left(\sum_{j=1}^{k} t_j x_j\right) \leq \sum_{j=1}^{k} t_j f(x_j). \tag{5.3}$$

Proof. We proceed by induction on k. The case $k = 1$ is trivial. Now assume the claim holds for $k-1$, we then prove that it holds for k as well. Let us fix $(x_j)_{j \in [\![1,k]\!]} \in \Omega^k$ and $(t_j)_{j \in [\![1,k]\!]} \in [0, 1]^k$ satisfying $\sum_{j=1}^{k} t_j = 1$. Note that the claim trivially follows if $t_k = 1$, since in that case $\sum_{j=1}^{k-1} t_j = 1 - t_k = 0$ such that $t_j = 0$ for all $j \in [\![1, k-1]\!]$. We can therefore assume that $t_k < 1$ and define $(t'_j)_{j \in [\![1,k-1]\!]}$ where $t'_j \coloneqq \frac{t_j}{1-t_k}$ for all $j \in [\![1, k-1]\!]$. Notice that $(t'_j)_{j \in [\![1,k-1]\!]} \in [0, 1]^{k-1}$:

$$\begin{aligned} \forall j \in [\![1, k-1]\!],\ 0 &\leq \frac{t_j}{1 - t_k} \\ &= \frac{t_j}{\sum_{i=1}^{k-1} t_i} \\ &\leq 1. \end{aligned}$$

Additionally, the t'_j sum to 1:

$$\begin{aligned}\sum_{j=1}^{k-1} t'_j &= \sum_{j=1}^{k-1} \frac{t_j}{1-t_k} \\ &= \frac{1}{1-t_k} \cdot \sum_{j=1}^{k-1} t_j \\ &= \frac{1}{1-t_k} \cdot (1-t_k) \\ &= 1.\end{aligned}$$

Now, define $x \coloneqq \sum_{j=1}^{k-1} t'_j x_j$. By induction hypothesis, $(x_j)_{j\in[\![1,k-1]\!]} \in \Omega^{k-1}$ implies that $x \in \Omega$ and $f(x) \leq \sum_{j=1}^{k-1} t'_j f(x_j)$. Next, let us derive:

$$\begin{aligned}\sum_{j=1}^{k} t_j x_j &= (1-t_k) \cdot \sum_{j=1}^{k-1} \frac{t_j}{1-t_k} x_j + t_k x_k \\ &= (1-t_k) \cdot \sum_{j=1}^{k-1} t'_j x_j + t_k x_k \\ &= (1-t_k)x + t_k x_k.\end{aligned}$$

On the one hand, since Ω is a convex set we deduce:

$$\sum_{j=1}^{k} t_j x_j = (1-t_k)x + t_k x_k \in \Omega.$$

On the other hand, since f is convex on Ω we deduce:

$$\begin{aligned}f\Big(\sum_{j=1}^{k} t_j x_j\Big) &= f\big((1-t_k)x + t_k x_k\big) \\ &\leq (1-t_k) f(x) + t_k f(x_k) \\ &\leq (1-t_k) \cdot \sum_{j=1}^{k-1} t'_j f(x_j) + t_k f(x_k) \\ &= (1-t_k) \cdot \sum_{j=1}^{k-1} \frac{t_j}{1-t_k} f(x_j) + t_k f(x_k) \\ &= \sum_{j=1}^{k} t_j f(x_j).\end{aligned}$$

The claim follows since we proved the induction. □

Definition 18. *Let $\Omega \subset \mathbb{R}^n$. We define the* ***convex hull of Ω*** *to be the set of all convex*

combinations of elements in Ω, that is:

$$\mathcal{CH}(\Omega) = \left\{ \sum_{j=1}^{k} t_j x_j : \ k \in \mathbb{N}^*, (x_j)_{j\in[\![1,k]\!]} \in \Omega^k, (t_j)_{j\in[\![1,k]\!]} \in [0,1]^k, \sum_{j=1}^{k} t_j = 1 \right\}. \quad (5.4)$$

Lemma 17. *Let $\Omega \subset \mathbb{R}^n$. Then $\mathcal{CH}(\Omega)$ is the smallest convex set containing Ω.*

Proof.

- $\Omega \subset \mathcal{CH}(\Omega)$. This follows trivially from the definition of the convex hull.
- $\mathcal{CH}(\Omega)$ is a convex set. To verify this, let us fix $(x, y) \in \left(\mathcal{CH}(\Omega)\right)^2$ and $r \in [0, 1]$. We want to show that $z \coloneqq rx + (1 - r)y \in \mathcal{CH}(\Omega)$, that is we want to prove that z is a convex combination of elements in Ω. Now, since $x \in \mathcal{CH}(\Omega)$ we can write:

  $$x = \sum_{j=1}^{k_1} s_j x_j : \ k_1 \in \mathbb{N}^*, (x_j)_{j\in[\![1,k_1]\!]} \in \Omega^{k_1}, (s_j)_{j\in[\![1,k_1]\!]} \in [0,1]^{k_1}, \sum_{j=1}^{k_1} s_j = 1.$$

  Similarly for $y \in \mathcal{CH}(\Omega)$:

  $$y = \sum_{j=1}^{k_2} t_j y_j : \ k_2 \in \mathbb{N}^*, (y_j)_{j\in[\![1,k_2]\!]} \in \Omega^{k_2}, (t_j)_{j\in[\![1,k_2]\!]} \in [0,1]^{k_2}, \sum_{j=1}^{k_2} t_j = 1.$$

  Define $k \coloneqq k_1 + k_2 \in \mathbb{N}^*$ and consider the sequence $(z_j)_{j\in[\![1,k]\!]}$:

  $$\forall j \in [\![1,k]\!], \ z_j \coloneqq \begin{cases} x_j \text{ if } & j \in [\![1,k_1]\!] \\ y_{j-k_1} \text{ if } & j \in [\![k_1+1,k]\!]. \end{cases}$$

  Consider as well the sequence $(u_j)_{j\in[\![1,k]\!]}$:

  $$\forall j \in [\![1,k]\!], \ u_j \coloneqq \begin{cases} rs_j \text{ if } & j \in [\![1,k_1]\!] \\ (1-r)t_{j-k_1} \text{ if } & j \in [\![k_1+1,k]\!]. \end{cases}$$

  Clearly we have $(z_j)_{j\in[\![1,k]\!]} \in \Omega^k$ and $(u_j)_{j\in[\![1,k]\!]} \in [0,1]^k$. Besides, the u_j sum to 1:

  $$\begin{aligned} \sum_{j=1}^{k} u_j &= \sum_{j=1}^{k_1} rs_j + \sum_{j=k_1+1}^{k} (1-r)t_{j-k_1} \\ &= r \cdot \sum_{j=1}^{k_1} s_j + (1-r) \cdot \sum_{j=1}^{k_2} t_j \\ &= r \cdot 1 + (1-r) \cdot 1 \\ &= 1. \end{aligned}$$

We conclude that z is indeed a convex combination of elements in Ω:

$$\begin{aligned} z &= rx + (1-r)y \\ &= r \cdot \sum_{j=1}^{k_1} s_j x_j + (1-r) \cdot \sum_{j=1}^{k_2} t_j y_j \\ &= \sum_{j=1}^{k_1} r s_j x_j + \sum_{j=k_1+1}^{k} (1-r) t_{j-k_1} y_{j-k_1} \\ &= \sum_{j=1}^{k} u_j z_j. \end{aligned}$$

- $\Omega \subset \mathcal{C}$ where $\mathcal{C}$ is a convex set $\implies \mathcal{CH}(\Omega) \subset \mathcal{C}$. To see this, fix $x \in \mathcal{CH}(\Omega)$:

$$x = \sum_{j=1}^{k} t_j x_j : \ k \in \mathbb{N}^*, (x_j)_{j\in[\![1,k]\!]} \in \Omega^k, (t_j)_{j\in[\![1,k]\!]} \in [0,1]^k, \sum_{j=1}^{k} t_j = 1.$$

By hypothesis, we have $(x_j)_{j\in[\![1,k]\!]} \in \Omega^k \subset \mathcal{C}^k$ where $\mathcal{C}$ is a convex set, therefore Lemma 16 ensures that $x = \sum_{j=1}^{k} t_j x_j \in \mathcal{C}$ as desired.

□

Lemma 18. *Let $(e_k)_{k\in[\![1,n]\!]}$ an orthonormal basis of $\mathbb{R}^n$, $x \in \mathbb{R}^n$ and $r > 0$. Define the set $S_r \coloneqq \{x \pm re_k : \ k \in [\![1,n]\!]\}$. Then the following holds:*

1. $r' \in [0,r] \implies S_{r'} \subset \mathcal{CH}(S_r)$.
2. $\overline{B\left(x, \frac{r}{n}\right)} \subset \mathcal{CH}(S_r)$.

Proof.

1. Fix $y \in S_{r'}$, i.e. $y = x \pm r'e_k$ where $k \in [\![1,n]\!]$. First we notice that $x \in \mathcal{CH}(S_r)$, due to the fact that one can write $x = \frac{1}{2}(x + re_1) + \frac{1}{2}(x - re_1)$ i.e. x is a convex combination of elements in S_r; besides, by definition of the convex hull, one has $S_r \subset \mathcal{CH}(S_r)$ such that $x \pm re_k \in \mathcal{CH}(S_r)$. Now, $r' \in [0,r]$ means there is $t \in [0,1]$ such that $r' = tr$. Let us then write:

$$\begin{aligned} y &= x \pm r'e_k \\ &= (1-t)x + tx \pm tre_k \\ &= (1-t)x + t(x \pm re_k). \end{aligned}$$

We know from Lemma 17 that $\mathcal{CH}(S_r)$ is a convex set, hence $y \in \mathcal{CH}(S_r)$.

2. Fix $y \in \overline{B\big(x, \frac{r}{n}\big)}$. Equivalently, this means that $y = x + \frac{r}{n}z$ where $\|z\| \leq 1$. Since $(e_k)_{k\in[\![1,n]\!]}$ is an orthonormal basis of $\mathbb{R}^n$, we can write:

$$z = \sum_{k=1}^{n} z_k e_k \quad \text{and} \quad \|z\| = \sqrt{\sum_{k=1}^{n} z_k^2}.$$

We can then decompose y as follows:

$$\begin{aligned} y &= x + \frac{r}{n}z \\ &= x + \frac{r}{n}\sum_{k=1}^{n} z_k e_k \\ &= \sum_{k=1}^{n} \frac{1}{n}(x + z_k r e_k). \end{aligned}$$

Now, the square-root function is non-decreasing and we have:

$$\begin{aligned} \forall k \in [\![1,n]\!],\ |z_k| &= \sqrt{z_k^2} \\ &\leq \sqrt{\sum_{j=1}^{n} z_j^2} \\ &= \|z\| \\ &\leq 1, \end{aligned}$$

implying that $|z_k|r \in [0, r]$ for all $k \in [\![1,n]\!]$; the first claim of the Lemma thus entails $x + z_k r e_k \in S_{|z_k|r} \subset \mathcal{CH}(S_r)$ for all $k \in [\![1,n]\!]$. This shows that y is a convex combination of elements in the convex set $\mathcal{CH}(S_r)$, hence using Lemma 16 finally yields $y \in \mathcal{CH}(S_r)$.

□

5.2 Extended-real-valued convex functions

In this section, convex functions taking value on the *extended real line* are analysed. We note that this framework is larger than that which truly matters in this thesis, namely convex functions taking value on the *real line* (addressed specifically in Section 5.3). Working in this more general framework yields a better understanding of the notion of function convexity via the introduction of two sets, namely the *effective domain* and the

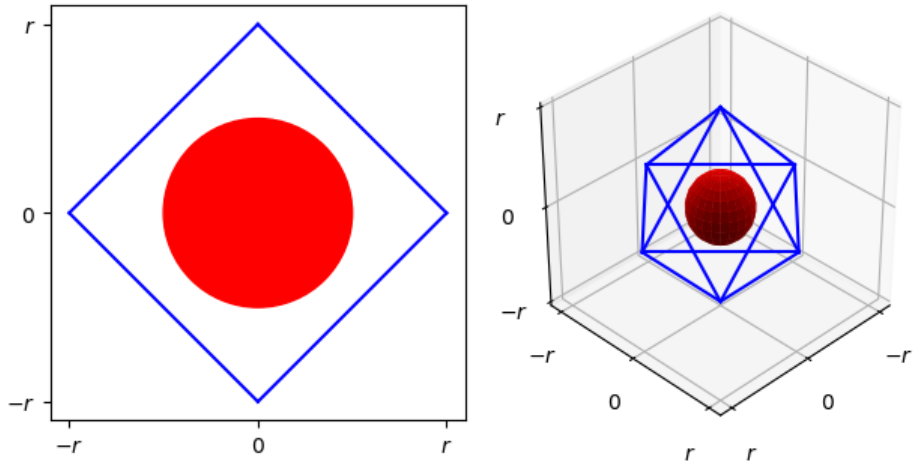

Figure 5.4: Illustration of the second clause of Lemma 18: $\overline{B\left(x, \frac{r}{n}\right)} \subset \mathcal{CH}(S_r)$ where $S_r \coloneqq \left\{x \pm re_k : k \in [\![1, n]\!]\right\}$ with $(e_k)_{k\in[\![1,n]\!]}$ an orthonormal basis of $\mathbb{R}^n$, $x \in \mathbb{R}^n$ and $r > 0$. In both figures, $x = 0$ and $(e_k)_{k\in[\![1,n]\!]}$ is the canonical orthonormal basis of $\mathbb{R}^n$ (left: $\mathbb{R}^2$; right: $\mathbb{R}^3$). The set $\overline{B\left(x, \frac{r}{n}\right)}$ is depicted in red and the edges of the boundary of the set $\mathcal{CH}(S_r)$ are depicted in blue.

epigraph of a convex function taking value on the extended real line. More precisely, useful characterizations of function convexity based on the effective domain and the epigraph are presented to the reader in Lemmas 19, 23, respectively.

In a second time, the crucial notion of *subgradient* is developed; subgradients are a generalization of the gradients of differentiable functions to convex functions taking value on the extended real line. Using our knowledge of convexity acquired in Section 5.1 as well as a series of three technical lemmas describing the topology of the effective domain and the epigraph (Lemmas 20-22), this section culminates with two fundamental results from convex analysis:

1. Lemma 24 gives conditions guaranteeing the existence of subgradients of convex functions taking value on the extended real line.

2. Lemma 25 gives conditions allowing to decompose subgradients of a sum of convex functions taking value on the extended real line into sums of subgradients of the functions taken separately.

Definition 19. *The* ***extended real-line*** *is the set* $\mathbb{R} \cup \{+\infty\}$ *equipped with the usual arithmetical operations on* $\mathbb{R}$*, with the addition of the following axioms:*

1. $\forall \alpha \in \mathbb{R} \cup \{+\infty\},\ \alpha + (+\infty) = (+\infty) + \alpha = +\infty.$

2. $\forall \alpha \in \mathbb{R}_+ \cup \{+\infty\},\ \alpha \cdot (+\infty) = (+\infty) \cdot \alpha = +\infty.$

Definition 20. *Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ a function. The **effective domain** of f is defined as:*

$$\mathcal{D}om(f) = \{x \in \mathbb{R}^n :\ f(x) < +\infty\}. \tag{5.5}$$

*When $\mathcal{D}om(f)$ is non-empty, f is said to be **proper**.*

Lemma 19. *Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ a proper function. Then f is convex on $\mathbb{R}^n$ if and only if $\mathcal{D}om(f)$ is a convex set and f is convex on $\mathcal{D}om(f)$.*

Proof.

$\Rightarrow$. It suffices to show that $\mathcal{D}om(f)$ is a convex set: indeed, one will then have that f is convex on $\mathcal{D}om(f)$ since it is convex on $\mathbb{R}^n$ and $\mathcal{D}om(f) \subset \mathbb{R}^n$. Fix $(x, y) \in \big(\mathcal{D}om(f)\big)^2$ and $t \in [0, 1]$. By definition of the effective domain we have $f(x) < +\infty$ and $f(y) < +\infty$. But since f is convex on $\mathbb{R}^n$, we have as well:

$$f\big(tx + (1-t)y\big) \leq tf(x) + (1-t)f(y) < +\infty,$$

hence $tx + (1-t)y \in \mathcal{D}om(f)$. This proves that $\mathcal{D}om(f)$ is a convex set.

$\Leftarrow$. By hypothesis, the convexity property satisfied by f is valid on $\mathcal{D}om(f)$ and we want to show it extends to the whole domain $\mathbb{R}^n$, that is it suffices to show:

$$\forall (x, y) \in \big(\mathbb{R}^n \setminus \mathcal{D}om(f)\big) \times \mathbb{R}^n,\ f\big(tx + (1-t)y\big) \leq tf(x) + (1-t)f(y).$$

Fix $(x, y) \in \big(\mathbb{R}^n \setminus \mathcal{D}om(f)\big) \times \mathbb{R}^n$ and $t \in [0, 1]$. With $t \geq 0$ we get that $tf(x) = +\infty$ (Definition 19: axiom 2). If $y \in \mathcal{D}om(f)$, then $(1-t)f(y) \in \mathbb{R}$; otherwise, with $1 - t \geq 0$ we get that $(1-t)f(y) = +\infty$ (Definition 19: axiom 2). In both cases we obtain $tf(x) + (1-t)f(y) = +\infty$ (Definition 19: axiom 1). It must then hold that $f\big(tx + (1-t)y\big) \leq tf(x) + (1-t)f(y)$, as desired.

□

Lemma 20. *Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ a proper function convex on $\mathbb{R}^n$ and $x \in \mathbb{R}^n$. If $x \in \overset{\circ}{\mathcal{D}om}(f)$, then f is upper-bounded on a closed ball centered at x:*

$$\exists r > 0, \exists \lambda \in \mathbb{R} :\ \forall y \in \overline{B(x, r)},\ f(y) \leq \lambda.^1 \tag{5.6}$$

[1] One could formulate a much stronger lemma: any function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ that is proper and convex on $\mathbb{R}^n$ is in fact continuous on the whole interior of its effective domain, $\overset{\circ}{\mathcal{D}om}(f)$ (assuming it is not empty). Rockafellar [1970]'s book discusses extensively of the continuity of convex functions.

Proof. Since $x \in \overset{\circ}{\mathcal{D}om}(f)$, we can find $r > 0$ such that $B(x,r) \subset \mathcal{D}om(f)$. Let $(e_k)_{k\in[\![1,n]\!]}$ denote an orthonormal basis of $\mathbb{R}^n$ and choose any r' in the open interval $]0,r[$, then define the following set:

$$S_{r'} \coloneqq \big\{x \pm r'e_k :\ k \in [\![1,n]\!]\big\}.$$

Notice $S_{r'}$ is a finite set and all its elements are in $B(x,r) \subset \mathcal{D}om(f)$. Indeed:

$$\forall k \in [\![1,n]\!],\ \big\|x \pm r'e_k - x\big\| = r'\|e_k\| = r' < r.$$

In particular, one can find $\lambda \in \mathbb{R}$ such that $f(y) \leq \lambda$ for all $y \in S_{r'}$. Now, due to Lemma 18 we know that $\overline{B\big(x, \frac{r'}{n}\big)} \subset \mathcal{CH}(S_{r'})$. Given a fixed $y \in \overline{B\big(x, \frac{r'}{n}\big)}$, the last statement entails that y is a convex combination of elements in $S_{r'}$:

$$y = \sum_{j=1}^{k} t_j y_j :\ k \in \mathbb{N}^*, (y_j)_{j\in[\![1,k]\!]} \in S_{r'}^k, (t_j)_{j\in[\![1,k]\!]} \in [0,1]^k, \sum_{j=1}^{k} t_j = 1.$$

Moreover, with $(y_j)_{j\in[\![1,k]\!]} \in S_{r'}^k$ we must have $f(y_j) \leq \lambda$ for all $j \in [\![1,k]\!]$. Since f is convex on $\mathbb{R}^n$, using Lemma 16 yields:

$$\begin{aligned} f(y) &= f\Big(\sum_{j=1}^{k} t_j y_j\Big) \\ &\leq \sum_{j=1}^{k} t_j f(y_j) \\ &\leq \sum_{j=1}^{k} t_j \lambda \\ &= \lambda \cdot \sum_{j=1}^{k} t_j \\ &= \lambda. \end{aligned}$$

We have just showed that f is upper-bounded by λ on the closed ball $\overline{B\big(x, \frac{r'}{n}\big)}$. □

Definition 21. *Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ a proper function. The **epigraph** of f is defined as:*

$$\mathcal{E}pi(f) = \{(x,\lambda) \in \mathbb{R}^n \times \mathbb{R} :\ f(x) \leq \lambda\}. \tag{5.7}$$

Lemma 21. *Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ a proper function and $x \in \mathbb{R}^n$. If $x \in \mathcal{D}om(f)$, then $\big(x, f(x)\big) \in \mathcal{E}pi(f) \setminus \overset{\circ}{\mathcal{E}pi}(f)$.*

Proof. Since $x \in \mathcal{D}om(f) \implies \big(x, f(x)\big) \in \mathcal{E}pi(f)$, clearly it only remains to show that

$\big(x, f(x)\big) \notin \overset{\circ}{\mathcal{E}pi}(f)$. By way of contradiction, suppose $\big(x, f(x)\big) \in \overset{\circ}{\mathcal{E}pi}(f)$, that is:

$$\exists r > 0 : \; B\Big(\big(x, f(x)\big), r\Big) \subset \mathcal{E}pi(f).$$

Consider the sequence $(\lambda_k)_k$ where $\lambda_k \coloneqq f(x) - \frac{1}{k}$. Clearly, one always has $\lambda_k < f(x)$. But at the same time, $(x, \lambda_k) \underset{k\to+\infty}{\longrightarrow} \big(x, f(x)\big)$, therefore one can find a large enough k such that $(x, \lambda_k) \in B\Big(\big(x, f(x)\big), r\Big) \subset \mathcal{E}pi(f)$. By definition of the epigraph, this would entail $f(x) \leq \lambda_k < f(x)$ which is a contradiction. □

Lemma 22. *Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ a proper function convex on $\mathbb{R}^n$. If $\overset{\circ}{\mathcal{D}om}(f)$ is non-empty, then $\overset{\circ}{\mathcal{E}pi}(f)$ is non-empty.*

Proof. Let $x \in \overset{\circ}{\mathcal{D}om}(f)$ (x must exist by hypothesis). Since f is convex on $\mathbb{R}^n$, by Lemma 20 we know that there is $r > 0$ such that f is upper-bounded on the closed ball $\overline{B(x, r)}$, that is we can find $\lambda < +\infty$ such that $f(y) \leq \lambda$ for all $y \in \overline{B(x, r)}$. We will now show that $\big(x, 2|\lambda|\big) \in \overset{\circ}{\mathcal{E}pi}(f)$: we will find $r' > 0$ such that $B\Big(\big(x, 2|\lambda|\big), r'\Big) \subset \mathcal{E}pi(f)$, or equivalently, such that for all $(y, \mu) \in \mathbb{R}^n \times \mathbb{R}$:

$$\big\|(y, \mu) - \big(x, 2|\lambda|\big)\big\| < r' \implies (y, \mu) \in \mathcal{E}pi(f).$$

Fix $(y, \mu) \in \mathbb{R}^n \times \mathbb{R}$ and define $r' \coloneqq \min\big(r, |\lambda|\big)$. This choice of r' gives us the desired result:

$$\begin{aligned}
\big\|(y, \mu) - \big(x, 2|\lambda|\big)\big\| < r' &\iff \big\|\big(y - x, \mu - 2|\lambda|\big)\big\|^2 < r'^2 \\
&\iff \|y - x\|^2 + \big(\mu - 2|\lambda|\big)^2 < r'^2 \\
&\implies \begin{cases} \|y - x\|^2 < r^2 \\ \big(\mu - 2|\lambda|\big)^2 < |\lambda|^2 \end{cases} \\
&\iff \begin{cases} \|y - x\| < r \\ \big|\mu - 2|\lambda|\big| < |\lambda| \end{cases} \\
&\implies \begin{cases} y \in \overline{B(x, r)} \\ -|\lambda| < \mu - 2|\lambda| \end{cases} \\
&\implies \begin{cases} f(y) \leq \lambda \\ |\lambda| < \mu \end{cases} \\
&\implies f(y) < \mu \\
&\implies (y, \mu) \in \mathcal{E}pi(f).
\end{aligned}$$

□

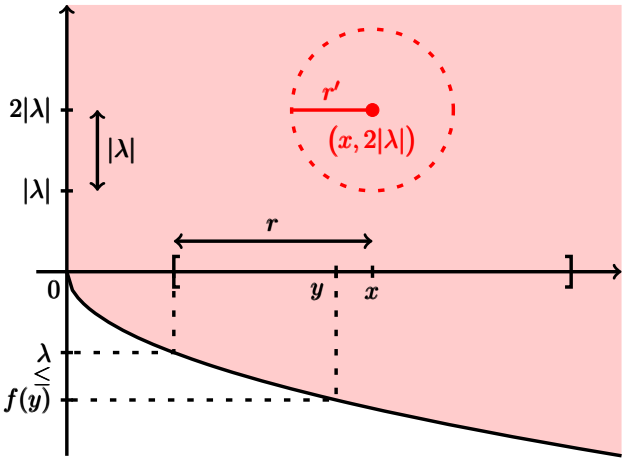

Figure 5.5: Illustration of Lemma 22: construction of an open ball included in $\mathcal{E}pi(f)$, where f is a (proper) function convex on $\mathbb{R}^n$ and $x \in \overset{\circ}{\mathcal{D}om}(f)$. The figure depicts the function $f : t \mapsto -\sqrt{t}$ if $t \geq 0$, $+\infty$ otherwise: $\mathcal{D}om(f) = [0, +\infty[$ and $\overset{\circ}{\mathcal{D}om}(f) =]0, +\infty[$. The existence of $r > 0$ and $\lambda < +\infty$ such that $f(y) \leq \lambda$ for all $y \in [x - r, x + r]$ is a consequence of Lemma 20; the constructed ball has radius $r' := \min\big(r, |\lambda|\big)$. The pink surface corresponds to $\mathcal{E}pi(f)$.

Lemma 23. *Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ a proper function. Then f is convex on $\mathbb{R}^n$ if and only if $\mathcal{E}pi(f)$ is a convex set.*

Proof.

$\Rightarrow$. Fix $\big((x, \lambda), (y, \mu)\big) \in \big(\mathcal{E}pi(f)\big)^2$ and $t \in [0, 1]$. We want to prove that:

$$t(x, \lambda) + (1 - t)(y, \mu) = \big(tx + (1 - t)y, t\lambda + (1 - t)\mu\big) \in \mathcal{E}pi(f),$$

that is we must show that $f\big(tx + (1 - t)y\big) \leq t\lambda + (1 - t)\mu < +\infty$. But since f is convex on $\mathbb{R}^n$ and $\big((x, \lambda), (y, \mu)\big) \in \big(\mathcal{E}pi(f)\big)^2$, we get as desired:

$$\begin{aligned} f\big(tx + (1 - t)y\big) &\leq tf(x) + (1 - t)f(y) \\ &\leq \underbrace{t\lambda}_{<+\infty} + \underbrace{(1 - t)\mu}_{<+\infty} < +\infty. \end{aligned}$$

$\Leftarrow$. Due to Lemma 19, it suffices to show that $\mathcal{D}om(f)$ is a convex set and f is convex on $\mathcal{D}om(f)$. Fix $(x, y) \in \big(\mathcal{D}om(f)\big)^2$ and $t \in [0, 1]$. Notice that $x \in \mathcal{D}om(f)$ clearly implies $\big(x, f(x)\big) \in \mathcal{E}pi(f)$; likewise, $\big(y, f(y)\big) \in \mathcal{E}pi(f)$. Now, since $\mathcal{E}pi(f)$

is a convex set we have:

$$t\big(x, f(x)\big) + (1-t)\big(y, f(y)\big) = \big(tx + (1-t)y, tf(x) + (1-t)f(y)\big) \in \mathcal{E}pi(f),$$

that is by definition of the epigraph:

$$f\big(tx + (1-t)y\big) \leq tf(x) + (1-t)f(y) < +\infty.$$

This proves both $\mathcal{D}om(f)$ is a convex set and f is convex on $\mathcal{D}om(f)$.

□

Definition 22. *Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ a proper function and $x \in \mathcal{D}om(f)$. A vector $v \in \mathbb{R}^n$ is called a **subgradient at point x** of f if:*

$$\forall y \in \mathcal{D}om(f),\ f(y) - f(x) \geq \langle v, y - x \rangle. \tag{5.8}$$

*The set of all subgradients at point x of f is called the **subdifferential at point x** of f and is denoted $\partial f(x)$.*

Lemma 24. *Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ a proper function convex on $\mathbb{R}^n$ and $x \in \mathbb{R}^n$. If $x \in \overset{\circ}{\mathcal{D}om}(f)$, then $\partial f(x)$ is non-empty.*

Proof. Our strategy will be to use the so-called *supporting hyperplane theorem* (see for instance Boyd and Vandenberghe [2014]'s book) on the set $\mathcal{E}pi(f)$. Remember that this fundamental theorem ensures that given a non-empty convex set and a point lying on its boundary, there exists an hyperplane containing that point such that the set is entirely contained in one of the two closed half-spaces bounded by the hyperplane: this hyperplane is said to *support* the set at that point.

We proceed to show the requirements for the supporting hyperplane theorem are satisfied:

- We first show that $\mathcal{E}pi(f)$ is non-empty and $\big(x, f(x)\big)$ lies on the boundary of $\mathcal{E}pi(f)$. This follows clearly from Lemma 21: one has $x \in \overset{\circ}{\mathcal{D}om}(f) \subset \mathcal{D}om(f)$, such that $\big(x, f(x)\big) \in \mathcal{E}pi(f) \setminus \overset{\circ}{\mathcal{E}pi}(f)$.
- We then show that $\mathcal{E}pi(f)$ is convex. This is immediate due to Lemma 23, since by hypothesis f is convex on $\mathbb{R}^n$.

We can therefore use the supporting hyperplane theorem which guarantees the existence of an hyperplane (in $\mathbb{R}^{n+1}$) supporting the set $\mathcal{E}pi(f)$ at point $\big(x, f(x)\big)$. Formally, this

translates into the following hyperplane inequality:

$$\exists (v,\alpha) \in \mathbb{R}^n \times \mathbb{R} :\ (v,\alpha) \neq 0 \quad \text{and} \quad \forall (y,\lambda) \in \mathcal{E}pi(f),\ \langle v, x\rangle + \alpha f(x) \leq \langle v, y\rangle + \alpha\lambda.$$

Notice that $\alpha \geq 0$ must hold. Indeed, suppose by way of contradiction that $\alpha < 0$. Given any $\lambda > f(x)$ we have $(x,\lambda) \in \mathcal{E}pi(f)$, hence the hyperplane inequality yields $\langle v, x\rangle + \alpha f(x) \leq \langle v, x\rangle + \alpha\lambda$, i.e. we get $\alpha f(x) \leq \alpha\lambda$. Dividing by (negative) α we would obtain $f(x) \geq \lambda > f(x)$ which is a contradiction. We have just showed that the hyperplane inequality reduces to:

$$\exists v \in \mathbb{R}^n, \exists \alpha \geq 0 :\ (v,\alpha) \neq 0 \quad \text{and} \quad \forall (y,\lambda) \in \mathcal{E}pi(f),\ \langle v, x\rangle + \alpha f(x) \leq \langle v, y\rangle + \alpha\lambda.$$

Since $y \in \mathcal{D}om(f) \Longrightarrow \big(y, f(y)\big) \in \mathcal{E}pi(f)$, let us now focus on the following restriction of the hyperplane inequality:

$$\exists v \in \mathbb{R}^n, \exists \alpha \geq 0 :\ (v,\alpha) \neq 0 \quad \text{and} \quad \forall y \in \mathcal{D}om(f),\ \langle v, x\rangle + \alpha f(x) \leq \langle v, y\rangle + \alpha f(y).$$

It turns out that $\alpha \neq 0$. To see this, suppose by way of contradiction that $\alpha = 0$. The restricted hyperplane inequality simplifies as such:

$$\exists v \in \mathbb{R}^n :\ v \neq 0 \quad \text{and} \quad \forall y \in \mathcal{D}om(f),\ \langle v, x\rangle \leq \langle v, y\rangle.$$

Since by hypothesis $x \in \overset{\circ}{\mathcal{D}om}(f)$, we have:

$$\exists r > 0 :\ B(x,r) \subset \mathcal{D}om(f).$$

Consider the sequence $(x_k)_k$ where $x_k \coloneqq x - \frac{v}{k}$. Due to $v \neq 0$, one can easily check that $\langle v, x_k\rangle < \langle v, x\rangle$ holds for all k. But at the same time x_k converges to x, therefore one can find a large enough k such that $x_k \in B(x,r) \subset \mathcal{D}om(f)$. The restricted hyperplane inequality would then yield $\langle v, x\rangle \leq \langle v, x_k\rangle < \langle v, x\rangle$ which is a contradiction. We have just showed that the restricted hyperplane inequality reduces to:

$$\exists v \in \mathbb{R}^n, \exists \alpha > 0 :\ \forall y \in \mathcal{D}om(f),\ \langle v, x\rangle + \alpha f(x) \leq \langle v, y\rangle + \alpha f(y).$$

Rearranging terms and dividing by (positive) α, this can be equivalently rewritten:

$$\begin{aligned} \exists v \in \mathbb{R}^n, \exists \alpha > 0 :\ \forall y \in \mathcal{D}om(f),\ f(y) - f(x) &\geq \Big\langle \frac{v}{\alpha}, x - y \Big\rangle \\ &= \Big\langle -\frac{v}{\alpha}, y - x \Big\rangle, \end{aligned}$$

which by definition of subgradient entails $-\frac{v}{\alpha} \in \partial f(x)$. □

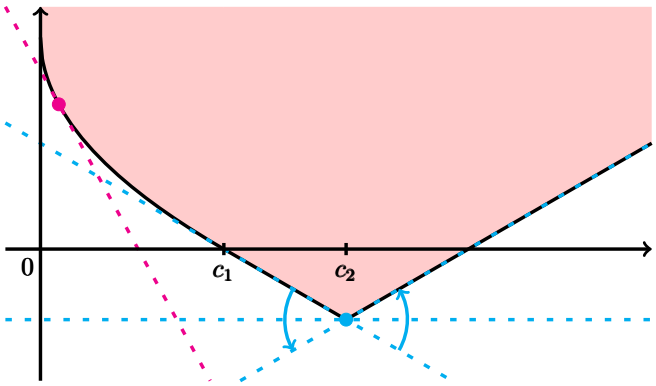

Figure 5.6: Illustration of Lemma 24: representation of hyperplanes supporting $\mathcal{E}pi(f)$, where f is a (proper) function convex on $\mathbb{R}^n$ (supporting hyperplanes are represented by dashed lines and $\mathcal{E}pi(f)$ is represented by the pink surface). Given $0 < c_1 < c_2$, the figure depicts the function $f : t \mapsto \frac{1}{\sqrt{c_1}}\big(|t - c_2| + c_1 - c_2\big)$ if $t \geq c_1$, $2\big(\sqrt{c_1} - \sqrt{t}\big)$ if $t \in [0, c_1]$, $+\infty$ otherwise. Supporting hyperplanes are carried by subgradients and always exist at points in $\overset{\circ}{\mathcal{D}om}(f)$, but may not exist at points on $\partial\big(\mathcal{D}om(f)\big) := \overline{\mathcal{D}om(f)} \backslash \overset{\circ}{\mathcal{D}om}(f)$. Here, $\mathcal{D}om(f) = [0, +\infty[$ and $\overset{\circ}{\mathcal{D}om}(f) =]0, +\infty[$, so $0 \in \partial\big(\mathcal{D}om(f)\big)$ and one can easily check that $\partial f(0) = \emptyset$ (intuitively, the slope of the supporting hyperplane at 0 is infinitely steep and cannot correspond to a subgradient). At differentiable points, a unique supporting hyperplane/subgradient exists (represented in magenta, see Lemma 27: clause 2); at non-differentiable points, several supporting hyperplanes/subgradients may exist (represented in cyan). Also notice that the horizontal line passing at $\big(c_2, f(c_2)\big)$ lies below the curve of f, i.e. $0 \in \partial f(c_2)$ and c_2 is the global minimizer of f (see Lemma 27: clause 1).

Lemma 25. *Let $f, g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ two proper functions convex on $\mathbb{R}^n$ such that $\overset{\circ}{\mathcal{D}om}(f) \cap \overset{\circ}{\mathcal{D}om}(g)$ is non-empty and let $x \in \mathbb{R}^n$. If $x \in \mathcal{D}om(f) \cap \mathcal{D}om(g)$, then $\partial(f+g)(x) \subset \partial f(x) + \partial g(x)$.*

Proof. We begin the proof with a preliminary part. Note that if $\partial(f+g)(x)$ is empty, the claim trivially follows. Let us then fix $v \in \partial(f+g)(x)$. By definition of subgradient, we have:

$$\forall y \in \mathcal{D}om(f+g) = \mathcal{D}om(f) \cap \mathcal{D}om(g),\ (f+g)(y) - (f+g)(x) \geq \langle v, y - x \rangle.$$

Let us define the function $h : y \mapsto -g(y) + \langle v, y - x \rangle + (f+g)(x)$. Clearly by construction, $f(x) = h(x)$. Moreover, the previous property entails:

$$\forall y \in \mathcal{D}om(g),\ f(y) \geq h(y).$$

Let us then define the so-called *hypograph* of the function h:

$$\mathcal{H}ypo(h) = \{(y, \lambda) \in \mathbb{R}^n \times \mathbb{R} :\ h(y) \geq \lambda\}.$$

By considering the symmetry operator $\mathcal{S} : (y, \lambda) \in \mathbb{R}^n \times \mathbb{R} \mapsto (y, -\lambda)$, we can easily establish the relationship between the sets $\mathcal{H}ypo(h)$ and $\mathcal{E}pi(-h)$:

$$\begin{aligned} \mathcal{H}ypo(h) &= \{(y, \lambda) \in \mathbb{R}^n \times \mathbb{R} :\ h(y) \geq \lambda\} \\ &= \mathcal{S}\big(\{(y, \lambda) \in \mathbb{R}^n \times \mathbb{R} :\ -h(y) \leq \lambda\}\big) \\ &= \mathcal{S}\big(\mathcal{E}pi(-h)\big). \end{aligned}$$

The main part of the proof begins. Our strategy will be to use the so-called *separating hyperplane theorem* (see for instance Boyd and Vandenberghe [2014]'s book) on the sets $\overset{\circ}{\mathcal{E}pi}(f)$ and $\mathcal{H}ypo(h)$. Remember that this fundamental theorem ensures that given two non-empty convex sets that are disjoint, there exists an hyperplane such that the first set is entirely contained in one of the two half-spaces bounded by the hyperplane, while the second set is entirely contained in the other half-space: this hyperplane is said to *separate* the sets. Also, if one of the sets is open, its corresponding half-space is open.

We proceed to show the requirements for the separating hyperplane theorem are satisfied:

- We first show that $\overset{\circ}{\mathcal{E}pi}(f)$ and $\mathcal{H}ypo(h)$ are non-empty. By hypothesis we have that $x \in \mathcal{D}om(f) \cap \mathcal{D}om(g) \subset \mathcal{D}om(g) = \mathcal{D}om(-h)$, therefore we trivially get that $\big(x, -h(x)\big) \in \mathcal{E}pi(-h)$. Knowing that $\mathcal{H}ypo(h) = \mathcal{S}\big(\mathcal{E}pi(-h)\big)$, clearly $\mathcal{H}ypo(h)$ is non-empty. Again by hypothesis, we know $\overset{\circ}{\mathcal{D}om}(f) \cap \overset{\circ}{\mathcal{D}om}(g)$ is non-empty and in

particular so is $\overset{\circ}{\mathcal{D}om}(f)$. With f being convex on $\mathbb{R}^n$, we get from Lemma 22 that $\overset{\circ}{\mathcal{E}pi}(f)$ is non-empty.

- We then show that $\overset{\circ}{\mathcal{E}pi}(f)$ and $\mathcal{H}ypo(h)$ are convex. By hypothesis f is convex on $\mathbb{R}^n$ thus by Lemma 23 we get that $\mathcal{E}pi(f)$ is a convex set. Combined with Lemma 13, this entails that $\overset{\circ}{\mathcal{E}pi}(f)$ is a convex set as well. Now, we have from Lemma 15 that the function $-h$ is convex on $\mathbb{R}^n$, since it is the sum of the affine function $y \mapsto -\langle v, y-x\rangle - (f+g)(x)$ and the function g, both convex on $\mathbb{R}^n$. Using Lemma 23 once more yields that $\mathcal{E}pi(-h)$ is a convex set; so is $\mathcal{H}ypo(h)$ since the symmetry operator $\mathcal{S}$ preserves convexity.

- We finally show that $\overset{\circ}{\mathcal{E}pi}(f)$ and $\mathcal{H}ypo(h)$ are disjoint. By way of contradiction, suppose there exists $(y,\lambda) \in \overset{\circ}{\mathcal{E}pi}(f) \cap \mathcal{H}ypo(h)$. Due to Lemma 21, one clearly has $(y,\lambda) \in \overset{\circ}{\mathcal{E}pi}(f) \implies \lambda > f(y)$; $(y,\lambda) \in \mathcal{H}ypo(h)$ translates $h(y) \geq \lambda$ and necessarily one must have $y \in \mathcal{D}om(g)$, which in turn lets us deduce $f(y) \geq h(y)$ must hold as was showed in the preliminary part of the proof. Combining results, we would obtain $\lambda > f(y) \geq h(y) \geq \lambda$ which is a contradiction.

We can therefore use the separating hyperplane theorem which guarantees the existence of an hyperplane (in $\mathbb{R}^{n+1}$) separating the open set $\overset{\circ}{\mathcal{E}pi}(f)$ and the set $\mathcal{H}ypo(h)$. Formally, this translates into the following hyperplane inequality:

$$\exists \beta \in \mathbb{R}, \exists (w,\alpha) \in \mathbb{R}^n \times \mathbb{R} : \ (w,\alpha) \neq 0 \quad \text{and} \quad \begin{cases} \forall (y,\lambda) \in \overset{\circ}{\mathcal{E}pi}(f), & \langle w, y\rangle + \alpha\lambda > \beta \\ \forall (z,\mu) \in \mathcal{H}ypo(h), & \langle w, z\rangle + \alpha\mu \leq \beta. \end{cases}$$

Note that since $\mathcal{E}pi(f)$ is a convex set and $\overset{\circ}{\mathcal{E}pi}(f)$ is non-empty, Lemma 14 ensures that every element of $\mathcal{E}pi(f)$ can be approximated by a sequence of elements in $\overset{\circ}{\mathcal{E}pi}(f)$. We can therefore rewrite the hyperplane inequalities as follows:

$$\exists \beta \in \mathbb{R}, \exists (w,\alpha) \in \mathbb{R}^n \times \mathbb{R} : \ (w,\alpha) \neq 0 \quad \text{and} \quad \begin{cases} \forall (y,\lambda) \in \mathcal{E}pi(f), & \langle w, y\rangle + \alpha\lambda \geq \beta \\ \forall (z,\mu) \in \mathcal{H}ypo(h), & \langle w, z\rangle + \alpha\mu \leq \beta. \end{cases}$$

Now, notice that $\alpha \geq 0$ must hold. Indeed, suppose by way of contradiction that $\alpha < 0$. By hypothesis we know that $x \in \mathcal{D}om(f) \cap \mathcal{D}om(g)$, so we can find $(\lambda,\mu) \in \mathbb{R}^2$ such that $(x,\lambda) \in \mathcal{E}pi(f)$, $(x,\mu) \in \mathcal{H}ypo(h)$ and $\lambda \neq \mu$. Using the hyperplane inequalities we deduce that $\langle w, x\rangle + \alpha\lambda \geq \beta \geq \langle w, x\rangle + \alpha\mu$, hence dividing by (negative) alpha yields $\lambda \leq \mu$. But at the same time we have by construction $\lambda \geq f(x) = h(x) \geq \mu$, i.e. $\lambda \geq \mu$. This would entail that $\lambda = \mu$ which is a contradiction. We have just showed that the

hyperplane inequalities reduce to:

$$\exists \beta \in \mathbb{R}, \exists w \in \mathbb{R}^n, \exists \alpha \geq 0 : \ (w, \alpha) \neq 0 \quad \text{and} \quad \begin{cases} \forall (y, \lambda) \in \mathcal{E}pi(f), & \langle w, y \rangle + \alpha\lambda \geq \beta \\ \forall (z, \mu) \in \mathcal{H}ypo(h), & \langle w, z \rangle + \alpha\mu \leq \beta. \end{cases}$$

Since $y \in \mathcal{D}om(f) \implies \big(y, f(y)\big) \in \mathcal{E}pi(f)$ and similarly $z \in \mathcal{D}om(g) \implies \big(z, h(z)\big) \in \mathcal{H}ypo(h)$, let us now focus on the following restriction of the hyperplane inequalities:

$$\exists \beta \in \mathbb{R}, \exists w \in \mathbb{R}^n, \exists \alpha \geq 0 : \ (w, \alpha) \neq 0 \quad \text{and} \quad \begin{cases} \forall y \in \mathcal{D}om(f), & \langle w, y \rangle + \alpha f(y) \geq \beta \\ \forall z \in \mathcal{D}om(g), & \langle w, z \rangle + \alpha h(z) \leq \beta. \end{cases}$$

We next prove that $\alpha \neq 0$. Again by way of contradiction, suppose $\alpha = 0$. The restricted hyperplane inequalities then simplify as follows:

$$\exists \beta \in \mathbb{R}, \exists w \in \mathbb{R}^n : \ w \neq 0 \quad \text{and} \quad \begin{cases} \forall y \in \mathcal{D}om(f), & \langle w, y \rangle \geq \beta \\ \forall z \in \mathcal{D}om(g), & \langle w, z \rangle \leq \beta. \end{cases}$$

Let us pick $z \in \overset{\circ}{\mathcal{D}om}(f) \cap \overset{\circ}{\mathcal{D}om}(g)$ (z must exist by hypothesis). This means:

$$\begin{cases} \exists r_1 > 0 : & B(z, r_1) \subset \mathcal{D}om(f) \\ \exists r_2 > 0 : & B(z, r_2) \subset \mathcal{D}om(g). \end{cases}$$

Consider the sequences $(z_k)_k$ and $(z'_k)_k$ where $z_k \coloneqq z - \frac{w}{k}$ and $z'_k \coloneqq z + \frac{w}{k}$. Owing to the fact that $w \neq 0$, one can easily check that $\langle w, z'_k \rangle > \langle w, z \rangle > \langle w, z_k \rangle$ holds for all k. At the same time, the sequences $(z_k)_k$ and $(z'_k)_k$ converge to z so we can find a large enough k such that both $z_k \in B(z, r_1) \subset \mathcal{D}om(f)$ and $z'_k \in B(z, r_2) \subset \mathcal{D}om(g)$ hold. Using the restricted hyperplane inequalities we deduce $\langle w, z_k \rangle \geq \beta \geq \langle w, z'_k \rangle$, but combining results we would obtain that $\langle w, z \rangle > \langle w, z_k \rangle \geq \langle w, z'_k \rangle > \langle w, z \rangle$ which is a contradiction. We have just showed that the restricted hyperplane inequalities reduce to:

$$\exists \beta \in \mathbb{R}, \exists w \in \mathbb{R}^n, \exists \alpha > 0 : \ \begin{cases} \forall y \in \mathcal{D}om(f), & \langle w, y \rangle + \alpha f(y) \geq \beta \\ \forall z \in \mathcal{D}om(g), & \langle w, z \rangle + \alpha h(z) \leq \beta. \end{cases}$$

We now have the necessary tools to finish the demonstration:

- On the one hand, using the fact that $\alpha > 0$ and $f(x) = h(x)$, the restricted hyperplane inequalities applied to $x \in \mathcal{D}om(f) \cap \mathcal{D}om(g) \subset \mathcal{D}om(g)$ yield:

$$\begin{aligned} y \in \mathcal{D}om(f) \implies & \langle w, y \rangle + \alpha f(y) \geq \langle w, x \rangle + \alpha h(x) \\ \iff & f(y) - h(x) \geq \Big\langle \frac{w}{\alpha}, x - y \Big\rangle \\ \iff & f(y) - f(x) \geq \Big\langle -\frac{w}{\alpha}, y - x \Big\rangle, \end{aligned}$$

that is $-\frac{w}{\alpha} \in \partial f(x)$.

- On the other hand, using again $\alpha > 0$ and the definition of h, the restricted hyperplane inequalities applied to $x \in \mathcal{D}om(f) \cap \mathcal{D}om(g) \subset \mathcal{D}om(f)$ yield:

$$\begin{aligned} y \in \mathcal{D}om(g) &\implies \langle w, x\rangle + \alpha f(x) \geq \langle w, y\rangle + \alpha h(y) \\ &\iff \left\langle \frac{w}{\alpha}, x - y \right\rangle \geq h(y) - f(x) \\ &\iff \left\langle \frac{w}{\alpha}, x - y \right\rangle \geq -g(y) + \langle v, y - x\rangle + (f+g)(x) - f(x) \\ &\iff \left\langle \frac{w}{\alpha}, x - y \right\rangle \geq \langle v, y - x\rangle + g(x) - g(y) \\ &\iff g(y) - g(x) \geq \left\langle \frac{w}{\alpha} + v, y - x \right\rangle, \end{aligned}$$

that is $\frac{w}{\alpha} + v \in \partial g(x)$. We finally conclude:

$$v = \overbrace{-\frac{w}{\alpha}}^{\in \partial f(x)} + \overbrace{\frac{w}{\alpha} + v}^{\in \partial g(x)} \in \partial f(x) + \partial g(x).$$

□

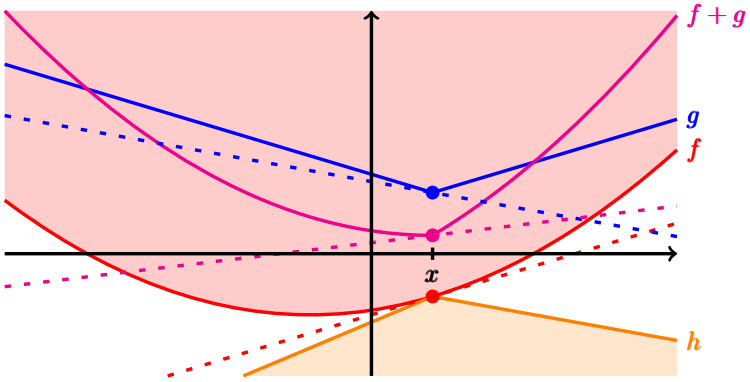

Figure 5.7: Illustration of Lemma 25: given two (proper) functions f and g convex on $\mathbb{R}^n$ such that $\mathcal{D}om(f) \cap \mathcal{D}om(g) \neq \emptyset$, $x \in \mathcal{D}om(f) \cap \mathcal{D}om(g)$ and $v_{f+g} \in \partial(f+g)(x)$ (identified with the slope of the magenta dashed line), one can construct a concave function h such that: 1. $\mathcal{H}ypo(h)$ is a rotated and shifted image of $\mathcal{E}pi(g)$; 2. $\mathcal{E}pi(f)$ and $\mathcal{H}ypo(h)$ (represented by the pink and orange surfaces, respectively) can be separated by an hyperplane corresponding to $v_f \in \partial(f)(x)$ (identified with the slope of the red dashed line). Then, one can construct $v_g := v_{f+g} - v_f \in \partial(g)(x)$ (identified with the slope of the blue dashed line): geometrically, the slope of the magenta dashed line equals the sum of the slopes of the red and blue dashed lines. Note that the constructed function h is only unique when both f and g are differentiable at x.

5.3 Real-valued convex functions

This section focuses on characterizing convexity of real-valued convex functions, which form a special case of the setting of extended-real-valued convex functions studied in Section 5.2. As such, the fundamental results obtained for the latter are inherited which helps us:

1. Characterize function convexity using subgradients in Lemma 26 with the help of Lemma 24 from Section 5.2.
2. Establish subgradient properties in the case of real-valued convex functions in Lemma 27, including: relation to the *argmin*; relation to the gradient in case of differentiability; calculus rules with the help of Lemma 25 from Section 5.2.

These subgradient properties are essential in order to understand the proximal gradient descent analysis in Section 2.7 (particularly in Lemma 8).

The section finishes with two additional characterizations of function convexity: the characterization from Lemma 26 is used to obtain a second characterization that holds for differentiable functions (Lemma 28); the characterization from Lemma 28 is in turn used to obtain a third characterization that holds for continuously differentiable functions (Lemma 29). These two new characterizations will come in handy later in the Appendix (Section 5.4).

Convention 3. ***In all that follows, we will write $f : \mathbb{R}^n \to \mathbb{R}$ to denote a (proper) function whose effective domain is $\mathcal{D}om(f) = \mathbb{R}^n$. If f is convex on the entire domain $\mathbb{R}^n$ we will simply write that f is convex. We do the same with other properties (e.g. strong convexity, differentiability, etc. . .).***

Lemma 26. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a function. Then f is convex if and only if $\partial f(x)$ is non-empty for all $x \in \mathbb{R}^n$.*

Proof.

$\Rightarrow$. This follows immediately from Lemma 24, noting that $\mathcal{D}om(f) = \mathbb{R}^n$ (see Convention 3), such that $\overset{\circ}{\mathcal{D}om}(f) = \mathbb{R}^n$.

$\Leftarrow$. Fix $(x, y) \in (\mathbb{R}^n)^2$ and $t \in [0, 1]$; write $z \coloneqq tx + (1 - t)y$. We want to show that $f(z) \leq tf(x) + (1 - t)f(y)$. By hypothesis $\partial f(z)$ is non-empty, that is:

$$\exists v \in \mathbb{R}^n : \ \forall z' \in \mathbb{R}^n, \ f(z') - f(z) \geq \langle v, z' - z \rangle.$$

With $z' = x$, we get:

$$\begin{aligned} f(z) &\leq f(x) + \big\langle v, tx + (1-t)y - x \big\rangle \\ &= f(x) + \big\langle v, (1-t)(y-x) \big\rangle \\ &= f(x) + (1-t)\langle v, y - x \rangle. \end{aligned}$$

Similarly with $z' = y$:

$$\begin{aligned} f(z) &\leq f(y) + \big\langle v, tx + (1-t)y - y \big\rangle \\ &= f(y) + \big\langle v, t(x-y) \big\rangle \\ &= f(y) - t\langle v, y - x \rangle. \end{aligned}$$

Combining these results, we obtain as desired:

$$\begin{aligned} f(z) &= tf(z) + (1-t)f(z) \\ &\leq t\big(f(x) + (1-t)\langle v, y-x\rangle\big) + (1-t)\big(f(y) - t\langle v, y-x\rangle\big) \\ &= tf(x) + (1-t)f(y) + t(1-t)\langle v, y-x\rangle - t(1-t)\langle v, y-x\rangle \\ &= tf(x) + (1-t)f(y). \end{aligned}$$

□

Lemma 27. *Let $f, g : \mathbb{R}^n \to \mathbb{R}$ two convex functions, $x \in \mathbb{R}^n$ and $\alpha > 0$. Then the following holds:*

1. $x = \underset{y \in \mathbb{R}^n}{\operatorname{argmin}}\ f(y) \iff 0 \in \partial f(x)$.
2. *f differentiable at x* $\implies \partial f(x) = \{\nabla f(x)\}$.
3. $\partial(f+g)(x) = \partial f(x) + \partial g(x)$.
4. $\partial(\alpha f)(x) = \alpha \partial f(x)$.

Proof.

1. We have the following:

$$\begin{aligned} x = \underset{y \in \mathbb{R}^n}{\operatorname{argmin}}\ f(y) &\iff \forall y \in \mathbb{R}^n,\ f(y) - f(x) \geq 0 \\ &\iff \forall y \in \mathbb{R}^n,\ f(y) - f(x) \geq \langle 0, y - x \rangle \\ &\iff 0 \in \partial f(x). \end{aligned}$$

2. We first verify that $\nabla f(x)$ is a subgradient of f at x, i.e. we want to show that $\forall y \in \mathbb{R}^n$, $f(y) - f(x) \geq \langle \nabla f(x), y - x \rangle$. By definition of the gradient of f at x, we know that:

$$\lim_{\|h\| \to 0} \frac{f(x+h) - f(x) - \langle \nabla f(x), h \rangle}{\|h\|} = 0.$$

Let us write $h = t(y - x)$ such that $\|h\| \underset{t \to 0}{\longrightarrow} 0$. Clearly one has:

$$\lim_{t \to 0} \frac{f\big(x + t(y-x)\big) - f(x) - \langle \nabla f(x), t(y-x) \rangle}{\big\|t(y-x)\big\|} = 0.$$

But owing to the convexity of f, one has as well that for all $t \in]0, 1]$:

$$\begin{aligned}
&\frac{f\big(x + t(y-x)\big) - f(x) - \langle \nabla f(x), t(y-x) \rangle}{\big\|t(y-x)\big\|} \\
&\qquad = \frac{f\big((1-t)x + ty\big) - f(x) - \langle \nabla f(x), t(y-x) \rangle}{\big\|t(y-x)\big\|} \\
&\qquad \leq \frac{(1-t)f(x) + tf(y) - f(x) - \langle \nabla f(x), t(y-x) \rangle}{\big\|t(y-x)\big\|} \\
&\qquad = \frac{t\big(f(y) - f(x)\big) - t\langle \nabla f(x), y - x \rangle}{t\|y-x\|} \\
&\qquad = \frac{f(y) - f(x) - \langle \nabla f(x), y - x \rangle}{\|y-x\|}.
\end{aligned}$$

Thus, taking the limit $t \longrightarrow 0$ on both sides yields:

$$0 \leq \frac{f(y) - f(x) - \langle \nabla f(x), y - x \rangle}{\|y-x\|}$$

and we obtain $f(y) - f(x) \geq \langle \nabla f(x), y - x \rangle$ as expected. The second part of the proof is to show that $\nabla f(x)$ is in fact the only element of $\partial f(x)$. For that purpose, let us fix $v \in \partial f(x)$, we then proceed to show that $v = \nabla f(x)$ must hold. Again, by definition of the gradient of f at x written with quantification we have:

$$\forall \epsilon > 0, \exists \delta > 0 : \ \big[\|h\| < \delta \implies \big|f(x+h) - f(x) - \langle \nabla f(x), h \rangle\big| < \epsilon \|h\|\big].$$

But since $v \in \partial f(x)$, we have as well:

$$\begin{aligned}
\big|f(x+h) - f(x) - \langle \nabla f(x), h \rangle\big| &\geq f(x+h) - f(x) - \langle \nabla f(x), h \rangle \\
&\geq \langle v, x + h - x \rangle - \langle \nabla f(x), h \rangle \\
&= \langle v, h \rangle - \langle \nabla f(x), h \rangle \\
&= \langle v - \nabla f(x), h \rangle,
\end{aligned}$$

such that:

$$\forall \epsilon > 0, \exists \delta > 0 : \ \big[\|h\| < \delta \implies \big\langle v - \nabla f(x), h\big\rangle < \epsilon \|h\|\big].$$

Now, it is straightforward to check that for fixed $\epsilon > 0$ and $\delta > 0$, the condition $\big[\|h\| < \delta \implies \langle w, h\rangle < \epsilon\|h\|\big]$ entails $\|w\| < \epsilon$ $\big($to verify this, choose $h = \frac{\delta}{2}\frac{w}{\|w\|}\big)$. Therefore, substituting $w \coloneqq v - \nabla f(x)$ we obtain $\|v - \nabla f(x)\| < \epsilon$ for all $\epsilon > 0$, implying $v = \nabla f(x)$.

3. We prove inclusions separately:

- $\subset$. This follows immediately from Lemma 25, noting that $\mathcal{D}om(f) = \mathcal{D}om(g) = \mathbb{R}^n$ (see Convention 3), such that $\mathcal{D}om(f) \cap \mathcal{D}om(g) = \overset{\circ}{\mathcal{D}om}(f) \cap \overset{\circ}{\mathcal{D}om}(g) = \mathbb{R}^n$.
- $\supset$. Let $v \in \partial f(x) + \partial g(x)$, i.e. $v = v_1 + v_2$ where $(v_1, v_2) \in \partial f(x) \times \partial g(x)$. We have by definition:

$$\forall y \in \mathbb{R}^n, \ f(y) - f(x) \geq \langle v_1, y - x\rangle \text{ and } g(y) - g(x) \geq \langle v_2, y - x\rangle,$$

hence we get by adding these two inequalities:

$$\begin{aligned} \forall y \in \mathbb{R}^n, \ (f+g)(y) - (f+g)(x) &\geq \langle v_1, y - x\rangle + \langle v_2, y - x\rangle \\ &= \langle v_1 + v_2, y - x\rangle \\ &= \langle v, y - x\rangle, \end{aligned}$$

that is $v \in \partial(f + g)(x)$.

4. Using the fact that $\alpha > 0$, we can write:

$$\begin{aligned} v \in \partial(\alpha f)(x) &\iff \forall y \in \mathbb{R}^n, \ \alpha f(y) - \alpha f(x) \geq \langle v, y - x\rangle \\ &\iff \forall y \in \mathbb{R}^n, \ f(y) - f(x) \geq \Big\langle \frac{v}{\alpha}, y - x\Big\rangle \\ &\iff \frac{v}{\alpha} \in \partial f(x) \\ &\iff v \in \alpha \partial f(x). \end{aligned}$$

□

Lemma 28. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a differentiable function. Then f is convex if and only if:*

$$\forall (x, y) \in (\mathbb{R}^n)^2, \ f(y) - f(x) \geq \big\langle \nabla f(x), y - x\big\rangle. \tag{5.9}$$

Proof. By definition of subgradients we have:

$$\big[\forall x \in \mathbb{R}^n,\ \nabla f(x) \in \partial f(x)\big] \iff \big[\forall (x,y) \in (\mathbb{R}^n)^2,\ f(y) - f(x) \geq \big\langle \nabla f(x), y - x \big\rangle\big].$$

$\Rightarrow$. The second clause of Lemma 27 guarantees $\nabla f(x) \in \partial f(x)$ for all $x \in \mathbb{R}^n$.

$\Leftarrow$. We have that $\nabla f(x) \in \partial f(x)$ for all $x \in \mathbb{R}^n$ and in particular $\partial f(x)$ is non-empty for all $x \in \mathbb{R}^n$. Due to Lemma 26, this entails that f is convex.

□

Lemma 29. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a continuously differentiable function. Then f is convex if and only if:*

$$\forall (x,y) \in (\mathbb{R}^n)^2,\ \big\langle \nabla f(x) - \nabla f(y), x - y \big\rangle \geq 0. \tag{5.10}$$

Proof.

$\Rightarrow$. Due to Lemma 28, we have that for all $(x,y) \in (\mathbb{R}^n)^2$:

$$f(y) - f(x) \geq \big\langle \nabla f(x), y - x \big\rangle \quad \text{and} \quad f(x) - f(y) \geq \big\langle \nabla f(y), x - y \big\rangle,$$

therefore we get by combining these inequalities:

$$\begin{aligned} 0 &= f(y) - f(x) + f(x) - f(y) \\ &\geq \big\langle \nabla f(x), y - x \big\rangle + \big\langle -\nabla f(y), y - x \big\rangle \\ &= \big\langle \nabla f(x) - \nabla f(y), y - x \big\rangle. \end{aligned}$$

Multiplying both sides by -1 yields the desired result.

$\Leftarrow$. Since f is continuously differentiable, we have access to its (multivariate) Taylor expansion with remainder of first order:

$$\forall (x,y) \in (\mathbb{R}^n)^2,\ f(y) - f(x) = \int_0^1 \big\langle \nabla f\big(x + t(y-x)\big), y - x \big\rangle dt.$$

Now, writing $z \coloneqq x + t(y - x)$ we have when $t > 0$ that $y - x = \frac{1}{t}(z - x)$, hence:

$$\begin{aligned} \forall t > 0,\ \big\langle \nabla f\big(x + t(y-x)\big), y - x \big\rangle &= \big\langle \nabla f(z), y - x \big\rangle \\ &= \big\langle \nabla f(z) - \nabla f(x) + \nabla f(x), y - x \big\rangle \\ &= \frac{1}{t}\big\langle \nabla f(z) - \nabla f(x), z - x \big\rangle + \big\langle \nabla f(x), y - x \big\rangle. \end{aligned}$$

From the latter and since by hypothesis we have $\langle \nabla f(z) - \nabla f(x), z - x \rangle \geq 0$ for all $(x, z) \in (\mathbb{R}^n)^2$, we deduce by positivity of the integral:

$$\begin{aligned}
\forall (x,y) \in (\mathbb{R}^n)^2,\ f(y) - f(x) &= \int_0^1 \langle \nabla f\big(x + t(y-x)\big), y - x \rangle dt \\
&= \int_0^1 \frac{1}{t} \langle \nabla f(z) - \nabla f(x), z - x \rangle dt \\
&\quad + \int_0^1 \langle \nabla f(x), y - x \rangle dt \\
&= \int_0^1 \underbrace{\frac{1}{t} \langle \nabla f(z) - \nabla f(x), z - x \rangle}_{\geq 0} dt + \langle \nabla f(x), y - x \rangle \\
&\geq 0 + \langle \nabla f(x), y - x \rangle \\
&= \langle \nabla f(x), y - x \rangle.
\end{aligned}$$

Applying Lemma 28, we conclude f is convex.

□

5.4 Gradient's Lipschitz continuity

Lipschitz continuity of the gradient is a crucial *smoothness* property intervening in the context of convex optimization. It is intimately tied to the theoretical analysis of the proximal gradient descent optimization scheme (see Section 2.7, notably Lemma 9).

More precisely, we show in this section that differentiable functions with a Lipschitz continuous gradient satisfy an important inequality (Lemma 33). In order to reach that goal, a technical lemma is needed (Lemma 30), as well as some additional knowledge about the *strong convexity* property (see in particular Lemma 31).

Definition 23. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a differentiable function and $L > 0$. We say that f has an **L-Lipschitz continuous gradient** if:*

$$\forall (x,y) \in (\mathbb{R}^n)^2,\ \big\|\nabla f(x) - \nabla f(y)\big\| \leq L\|x - y\|. \tag{5.11}$$

Lemma 30. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a differentiable function with an L-Lipschitz continuous gradient. Then the function $g_\rho : x \mapsto \frac{\rho}{2}\|x\|^2 - f(x)$ is convex for all $\rho \geq L$.*

Proof. Fix $\rho \geq L$. By hypothesis, f has an L-Lipschitz continuous gradient; in particular it is continuously differentiable and the same is true of g_ρ. By Lemma 29, we know that

to prove the convexity of g_ρ it suffices to show:

$$\forall (x,y) \in (\mathbb{R}^n)^2,\ \big\langle \nabla g_\rho(x) - \nabla g_\rho(y), x-y \big\rangle \geq 0.$$

The gradient of g_ρ at x is $\nabla g_\rho(x) = \rho x - \nabla f(x)$ so we have:

$$\begin{aligned}\forall (x,y) \in (\mathbb{R}^n)^2,\ \big\langle \nabla g_\rho(x) - \nabla g_\rho(y), x-y \big\rangle &= \big\langle \rho x - \rho y + \nabla f(y) - \nabla f(x), x-y \big\rangle \\ &= \rho \|x-y\|^2 - \big\langle \nabla f(x) - \nabla f(y), x-y \big\rangle \\ &\geq L\|x-y\|^2 - \big\langle \nabla f(x) - \nabla f(y), x-y \big\rangle.\end{aligned}$$

Now, notice that one can apply the Cauchy-Schwarz inequality and use the L-Lipschitz continuous property of the gradient of f to get:

$$\begin{aligned}\forall (x,y) \in (\mathbb{R}^n)^2,\ \big\langle \nabla f(x) - \nabla f(y), x-y \big\rangle &\leq \big\| \nabla f(x) - \nabla f(y) \big\| \|x-y\| \\ &\leq L\|x-y\|^2,\end{aligned}$$

thus we conclude:

$$\begin{aligned}\forall (x,y) \in (\mathbb{R}^n)^2, \big\langle \nabla g_\rho(x) - \nabla g_\rho(y), x-y \big\rangle &\geq L\|x-y\|^2 - \big\langle \nabla f(x) - \nabla f(y), x-y \big\rangle \\ &\geq L\|x-y\|^2 - L\|x-y\|^2 \\ &= 0,\end{aligned}$$

proving g_ρ is convex. □

Definition 24. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a function and $\rho \geq 0$. We say that f is **ρ-strongly convex** if:*

$$\forall x \in \mathbb{R}^n,\ \partial f(x) \text{ is non-empty and } \forall (v,y) \in \partial f(x) \times \mathbb{R}^n,$$
$$f(y) - f(x) \geq \big\langle v, y-x \big\rangle + \frac{\rho}{2}\|x-y\|^2. \quad (5.12)$$

Lemma 31. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a function and $\rho \geq 0$. Then f is ρ-strongly convex if and only if the function $g : x \mapsto f(x) - \frac{\rho}{2}\|x\|^2$ is convex.*

Proof. We know from the third clause of Lemma 27 that the subdifferential is additive, such that $\partial f(x) = \partial g(x) + \partial\big(\frac{\rho}{2}\|\cdot\|^2\big)(x)$. Moreover, the second clause of the same Lemma tells us that $\partial\big(\frac{\rho}{2}\|\cdot\|^2\big)(x) = \big\{\nabla\big(\frac{\rho}{2}\|\cdot\|^2\big)(x)\big\} = \{\rho x\}$, hence $\partial f(x) = \partial g(x) + \{\rho x\}$. By Lemma 26 we know that g is convex if and only if $\partial g(x) = \partial f(x) - \{\rho x\}$ is non-empty for all $x \in \mathbb{R}^n$, which can be equivalently rewritten as:

$$\forall x \in \mathbb{R}^n,\ \partial f(x) \text{ is non-empty and } \forall (v,y) \in \partial f(x) \times \mathbb{R}^n,\ g(y) - g(x) \geq \langle v - \rho x, y - x \rangle.$$

Now, $g(y) - g(x) \geq \langle v - \rho x, y - x \rangle$ if and only if:

$$\begin{aligned} f(y) - f(x) &\geq \frac{\rho}{2}\|y\|^2 - \frac{\rho}{2}\|x\|^2 + \langle v - \rho x, y - x \rangle \\ &= \frac{\rho}{2}\big(\|y\|^2 - \|x\|^2\big) + \langle v, y - x \rangle - \rho\langle x, y \rangle + \rho\|x\|^2 \\ &= \langle v, y - x \rangle + \frac{\rho}{2}\big(\|x\|^2 - 2\langle x, y \rangle + \|y\|^2\big) \\ &= \langle v, y - x \rangle + \frac{\rho}{2}\|x - y\|^2. \end{aligned}$$

We therefore conclude that g is convex if and only if:

$$\forall x \in \mathbb{R}^n,\ \partial f(x) \text{ is non-empty and } \forall (v, y) \in \partial f(x) \times \mathbb{R}^n,$$
$$f(y) - f(x) \geq \big\langle v, y - x \big\rangle + \frac{\rho}{2}\|x - y\|^2.$$

The latter is exactly the definition of f being ρ-strongly convex. □

Lemma 32. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a function and $\rho \geq 0$. Then f is ρ-strongly convex if and only if there is $a \in \mathbb{R}^n$ such that the function $g_a : x \mapsto f(x) - \frac{\rho}{2}\|x - a\|^2$ is convex.*

Proof. Due to Lemma 31 it is equivalent to show that the function $g : x \mapsto f(x) - \frac{\rho}{2}\|x\|^2$ is convex if and only if there is $a \in \mathbb{R}^n$ such that the function $g_a : x \mapsto f(x) - \frac{\rho}{2}\|x - a\|^2$ is convex.

⇒. Trivial with $a = 0$.

⇐. Assume there is $a \in \mathbb{R}^n$ such that the function $g_a : x \mapsto f(x) - \frac{\rho}{2}\|x - a\|^2$ is convex. Clearly one has that:

$$\begin{aligned} \forall x \in \mathbb{R}^n,\ g(x) &= f(x) - \frac{\rho}{2}\|x\|^2 \\ &= f(x) - \frac{\rho}{2}\|x - a + a\|^2 \\ &= f(x) - \frac{\rho}{2}\|x - a\|^2 - \rho\langle x - a, a \rangle - \frac{\rho}{2}\|a\|^2 \\ &= g_a(x) + \frac{\rho}{2}\|a\|^2 - \rho\langle x, a \rangle, \end{aligned}$$

that is $g = g_a + h_a$ where $h_a : x \mapsto \frac{\rho}{2}\|a\|^2 - \rho\langle x, a \rangle$ is an affine function and is therefore convex. Since g_a is convex as well by hypothesis, Lemma 15 lets us conclude that g is convex as the sum of two convex functions.

□

Lemma 33. *Let $f : \mathbb{R}^n \to \mathbb{R}$ a differentiable function with an L-Lipschitz continuous*

gradient. Then the following holds:

$$\forall (x,y) \in (\mathbb{R}^n)^2,\ f(y) \leq f(x) + \big\langle \nabla f(x), y - x \big\rangle + \frac{L}{2}\|x-y\|^2. \tag{5.13}$$

Proof. First notice that due to Lemma 30, the hypothesis that f is differentiable with an L-Lipschitz continuous gradient entails the function $g : x \mapsto \frac{L}{2}\|x\|^2 - f(x)$ is convex. Moreover, since g is itself differentiable we know from Lemma 28 that the convexity of g can be restated as:

$$\forall (x,y) \in (\mathbb{R}^n)^2,\ g(y) - g(x) \geq \big\langle \nabla g(x), y - x \big\rangle.$$

With $g(x) = \frac{L}{2}\|x\|^2 - f(x)$ and $\nabla g(x) = Lx - \nabla f(x)$, this means:

$$\forall (x,y) \in (\mathbb{R}^n)^2,\ \frac{L}{2}\|y\|^2 - \frac{L}{2}\|x\|^2 + f(x) - f(y) \geq \big\langle Lx - \nabla f(x), y - x \big\rangle.$$

Rearranging terms, this becomes:

$$\forall (x,y) \in (\mathbb{R}^n)^2,\ f(y) \leq f(x) + \big\langle \nabla f(x) - Lx, y - x \big\rangle + \frac{L}{2}\|y\|^2 - \frac{L}{2}\|x\|^2.$$

Now clearly, by Lemma 31 the function $x \mapsto \frac{L}{2}\|x\|^2$ is L-strongly convex, its gradient at y is equal to Ly, so we have by definition of strong convexity:

$$\forall (x,y) \in (\mathbb{R}^n)^2,\ \frac{L}{2}\|x\|^2 - \frac{L}{2}\|y\|^2 \geq \langle Ly, x - y \rangle + \frac{L}{2}\|x-y\|^2.$$

We thus deduce:

$$\begin{aligned}
\forall (x,y) \in (\mathbb{R}^n)^2,\ \frac{L}{2}\|y\|^2 - \frac{L}{2}\|x\|^2 &\leq \langle Ly, y - x \rangle - \frac{L}{2}\|x-y\|^2 \\
&= \langle Ly - Lx + Lx, y - x \rangle - \frac{L}{2}\|x-y\|^2 \\
&= L\langle y - x, y - x \rangle + \langle Lx, y - x \rangle - \frac{L}{2}\|x-y\|^2 \\
&= L\|x-y\|^2 - \frac{L}{2}\|x-y\|^2 + \langle Lx, y - x \rangle \\
&= \frac{L}{2}\|x-y\|^2 + \langle Lx, y - x \rangle.
\end{aligned}$$

Combining results, we finally get:

$$\begin{aligned}\forall (x,y) \in (\mathbb{R}^n)^2,\ f(y) &\leq f(x) + \big\langle \nabla f(x) - Lx, y - x \big\rangle + \frac{L}{2}\|y\|^2 - \frac{L}{2}\|x\|^2 \\ &\leq f(x) + \big\langle \nabla f(x) - Lx, y - x \big\rangle + \frac{L}{2}\|x-y\|^2 + \langle Lx, y - x \rangle \\ &= f(x) + \big\langle \nabla f(x) + Lx - Lx, y - x \big\rangle + \frac{L}{2}\|x-y\|^2 \\ &= f(x) + \big\langle \nabla f(x), y - x \big\rangle + \frac{L}{2}\|x-y\|^2.\end{aligned}$$

□

5.5 Proximal operator: closed-form examples

In this section we want to derive closed-form solutions to *proximal operators* (see Section 2.7) of certain convex functions. The key ingredients of the section are the *subdifferentials* of the Euclidean norm (see Lemma 34). These subdifferentials enable the computation of a closed-form solution to the proximal operator of the (scaled) Euclidean norm in Lemma 35. A closed-form solution to a more general proximal operator is computed in Lemma 37, with the help of a technical lemma (Lemma 36).

The proximal operator from Lemma 37 is of interest, since it generalizes as well the proximal operator whose closed-form solution is computed in the ProxiMAS algorithm (Algorithm 3.5: line 5.ii).

Lemma 34. *The Euclidean norm on $\mathbb{R}^n$ has the following subdifferentials:*

$$\forall x \in \mathbb{R}^n,\ \partial\big(\|\cdot\|\big)(x) = \begin{cases} \left\{\frac{x}{\|x\|}\right\} & \text{if } x \neq 0 \\ \\ \overline{B(0,1)} & \text{if } x = 0. \end{cases} \tag{5.14}$$

Proof. We first remark that the Euclidean norm satisfies the triangular inequality and in particular it is convex, thus from Lemma 26 we know that $\partial\big(\|\cdot\|\big)(x)$ is non-empty for all $x \in \mathbb{R}^n$. Notice as well that the Euclidean norm can be written as the following composition of functions: $\|\cdot\| = \sqrt{\cdot} \circ \|\cdot\|^2$. We know that the squared Euclidean norm $\|\cdot\|^2$ is differentiable on $\mathbb{R}^n$ and $\|x\|^2 > 0$ for all $x \neq 0$. Since $\sqrt{\cdot}$ is differentiable on $\mathbb{R}_+^*$, we deduce that 0 is the only point of non-differentiability of $\|\cdot\|$ and separate our analysis accordingly. Fix $x \in \mathbb{R}^n$; two cases should be discussed:

- $x \neq 0$: $\|\cdot\|$ is convex and differentiable at x, thus by the second clause of Lemma 27 we know that $\partial\big(\|\cdot\|\big)(x) = \big\{\nabla\big(\|\cdot\|\big)(x)\big\}$. Now, consider the *differential operator*

d, defined as follows:

$$\forall y \in \mathbb{R}^n, \forall f : \mathbb{R}^n \to \mathbb{R} \text{ differentiable at } y,\ d(f)(y) : h \in \mathbb{R}^n \mapsto \big\langle \nabla f(y), h \big\rangle.$$

We have:

$$\begin{cases} \forall h \in \mathbb{R}^n,\ d\big(\|\cdot\|^2\big)(x)(h) = \big\langle \nabla\big(\|\cdot\|^2\big)(x), h \big\rangle = \langle 2x, h \rangle \\ \forall h \in \mathbb{R},\ d\big(\sqrt{\cdot}\big)\big(\|x\|^2\big)(h) = \big\langle \nabla\big(\sqrt{\cdot}\big)\big(\|x\|^2\big), h \big\rangle = \frac{h}{2\sqrt{\|x\|^2}} = \frac{h}{2\|x\|}. \end{cases}$$

Applying the *differential chain rule* then yields:

$$\begin{aligned} \forall h \in \mathbb{R}^n,\ d\big(\|\cdot\|\big)(x)(h) &= d\big(\sqrt{\cdot} \circ \|\cdot\|^2\big)(x)(h) \\ &= d\big(\sqrt{\cdot}\big)\big(\|x\|^2\big)\Big(d\big(\|\cdot\|^2\big)(x)(h)\Big) \\ &= d\big(\sqrt{\cdot}\big)\big(\|x\|^2\big)\big(\langle 2x, h \rangle\big) \\ &= \frac{\langle 2x, h \rangle}{2\|x\|} \\ &= \Big\langle \frac{x}{\|x\|}, h \Big\rangle, \end{aligned}$$

that is we have just showed $\nabla\big(\|\cdot\|\big)(x) = \frac{x}{\|x\|}$ and $\partial\big(\|\cdot\|\big)(x) = \big\{\frac{x}{\|x\|}\big\}$.

- $x = 0$: We prove that $\partial\big(\|\cdot\|\big)(0) = \overline{B(0,1)}$ by double inclusion:

  $\subset$. Let $v \in \partial\big(\|\cdot\|\big)(0)$, i.e. $\|y\| - \|0\| \geq \langle v, y - 0 \rangle$ for all $y \in \mathbb{R}^n$. Setting $y = v$, we get in particular $\|v\| \geq \langle v, v \rangle = \|v\|^2$, implying $\|v\| \leq 1$.

  $\supset$. Let $v \in \overline{B(0,1)}$, i.e. $\|v\| \leq 1$. Applying the Cauchy-Schwarz inequality immediately yields:

$$\begin{aligned} \forall y \in \mathbb{R}^n,\ \langle v, y - 0 \rangle &= \langle v, y \rangle \\ &\leq \|v\|\|y\| \\ &\leq \|y\| \\ &= \|y\| - \|0\|, \end{aligned}$$

  proving that $v \in \partial\big(\|\cdot\|\big)(0)$.

□

Lemma 35. *Let $\|\cdot\|$ denote the Euclidean norm on $\mathbb{R}^n$ and let $\alpha > 0$. The proximal operator of $\alpha\|\cdot\|$ at point $a \in \mathbb{R}^n$ has the following closed-form solution:*

$$\mathcal{P}rox\big(\alpha\|\cdot\|\big)(a) = \begin{cases} \big(\|a\| - \alpha\big) \cdot \frac{a}{\|a\|} & \text{if } \|a\| > \alpha \\ 0 & \text{if } \|a\| \leq \alpha. \end{cases} \tag{5.15}$$

Proof. By definition of the proximal operator and using all clauses from Lemma 27, we can write:

$$\begin{aligned} x_* = \mathcal{P}rox\big(\alpha\|\cdot\|\big)(a) &\iff x_* = \operatorname*{argmin}_{x\in\mathbb{R}^n} \Big(\alpha\|x\| + \frac{1}{2}\|x-a\|^2\Big) \\ &\iff 0 \in \partial\Big(\alpha\|\cdot\| + \frac{1}{2}\|\cdot - a\|^2\Big)(x_*) \\ &\iff 0 \in \alpha\partial\big(\|\cdot\|\big)(x_*) + \partial\Big(\frac{1}{2}\|\cdot - a\|^2\Big)(x_*) \\ &\iff 0 \in \alpha\partial\big(\|\cdot\|\big)(x_*) + \Big\{\nabla\Big(\frac{1}{2}\|\cdot - a\|^2\Big)(x_*)\Big\} \\ &\iff 0 \in \alpha\partial\big(\|\cdot\|\big)(x_*) + \big\{x_* - a\big\}. \end{aligned}$$

In fact, due to Lemma 34 we have the following set equality:

$$\alpha\partial\big(\|\cdot\|\big)(x_*) + \big\{x_* - a\big\} = \begin{cases} \alpha\big\{\frac{x_*}{\|x_*\|}\big\} + \big\{x_* - a\big\} = \big\{\alpha\frac{x_*}{\|x_*\|} + x_* - a\big\} & \text{if } x_* \neq 0 \\ \alpha\overline{B(0,1)} + \big\{x_* - a\big\} = \overline{B(-a,\alpha)} & \text{if } x_* = 0. \end{cases}$$

In particular, we have:

$$x_* = \mathcal{P}rox\big(\alpha\|\cdot\|\big)(a) \implies 0 \in \begin{cases} \big\{\alpha\frac{x_*}{\|x_*\|} + x_* - a\big\} & \text{if } x_* \neq 0 \\ \overline{B(-a,\alpha)} & \text{if } x_* = 0. \end{cases}$$

The proof will thus be complete if we can establish:

$$0 \in \begin{cases} \big\{\alpha\frac{x_*}{\|x_*\|} + x_* - a\big\} & \text{if } x_* \neq 0 \\ \overline{B(-a,\alpha)} & \text{if } x_* = 0 \end{cases} \implies x_* = \begin{cases} \big(\|a\| - \alpha\big)\cdot\frac{a}{\|a\|} & \text{if } \|a\| > \alpha \\ 0 & \text{if } \|a\| \leq \alpha. \end{cases}$$

Assume the left-hand side holds. Now, notice that a simple derivation yields:

$$\begin{aligned} x_* \neq 0 \implies \alpha + \|x_*\| &= \Big(\frac{\alpha}{\|x_*\|} + 1\Big)\|x_*\| \\ &= \Big\|\Big(\frac{\alpha}{\|x_*\|} + 1\Big)x_*\Big\|, \end{aligned}$$

hence we deduce:

$$\begin{aligned} x_* \neq 0 &\implies 0 \in \Big\{\alpha\frac{x_*}{\|x_*\|} + x_* - a\Big\} \text{ and } \alpha + \|x_*\| = \Big\|\Big(\frac{\alpha}{\|x_*\|} + 1\Big)x_*\Big\| \\ &\implies a = \Big(\frac{\alpha}{\|x_*\|} + 1\Big)x_* \text{ and } \|a\| = \alpha + \|x_*\|. \end{aligned}$$

We can then easily show that $x_* = 0 \iff \|a\| \leq \alpha$:

$\Rightarrow$. This is immediate:

$$\begin{aligned} x_* = 0 &\implies 0 \in \overline{B(-a,\alpha)} \\ &\iff \big\|0-(-a)\big\| = \|a\| \leq \alpha. \end{aligned}$$

$\Leftarrow$. To see this, we proceed by way of contradiction. Suppose $\|a\| \leq \alpha$ and $x_* \neq 0$. We noticed previously that $x_* \neq 0$ entails $\|a\| = \alpha + \|x_*\|$, but then $\|x_*\| > 0$ causes $\|a\| > \alpha$ which is a contradiction.

Finally, the previous analysis lets us investigate the case $\|a\| > \alpha$:

$$\begin{aligned} \|a\| > \alpha &\iff x_* \neq 0 \\ &\implies a = \Big(\frac{\alpha}{\|x_*\|} + 1\Big) x_* \text{ and } \|a\| = \alpha + \|x_*\| \\ &\implies x_* = \Big(\frac{\alpha}{\|a\| - \alpha} + 1\Big)^{-1} a \\ &\iff x_* = \Big(\frac{\|a\|}{\|a\| - \alpha}\Big)^{-1} a \\ &\iff x_* = \big(\|a\| - \alpha\big) \cdot \frac{a}{\|a\|}. \end{aligned}$$

□

Lemma 36. *Consider a decomposition of $\mathbb{R}^n$, in the following sense:*

$$y \in \mathbb{R}^n \iff \begin{cases} y = (y_1, \dots, y_k) \in \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_k} \\ \sum_{j=1}^k n_j = n. \end{cases} \tag{5.16}$$

For all $j \in [\![1,k]\!]$, let $f_j : \mathbb{R}^{n_j} \to \mathbb{R}$ a function. Construct $f : x \in \mathbb{R}^n \mapsto \sum_{j=1}^k f_j(x_j)$. Then the following holds:

$$\forall (v,x) \in (\mathbb{R}^n)^2,\ \big[v \in \partial f(x) \iff \forall j \in [\![1,k]\!],\ v_j \in \partial f_j(x_j)\big]. \tag{5.17}$$

Proof. Fix $(v,x) \in (\mathbb{R}^n)^2$ and write $(v_1, \dots, v_k)$ $\big($respectively $(x_1, \dots, x_k)\big)$ the decomposition of v (respectively x) in $\mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_k}$.

$\Rightarrow$. Fix $j \in [\![1,k]\!]$. For all $y_j \in \mathbb{R}^{n_j}$, define $\widehat{y_j} \coloneqq (x_1, \dots, x_{j-1}, y_j, x_{j+1}, \dots, x_k) \in \mathbb{R}^n$. We have:

$$\begin{aligned} v \in \partial f(x) &\iff \forall y \in \mathbb{R}^n,\ f(y) - f(x) \geq \langle v, y - x \rangle \\ &\implies \forall y_j \in \mathbb{R}^{n_j},\ f(\widehat{y_j}) - f(x) \geq \langle v, \widehat{y_j} - x \rangle, \end{aligned}$$

where by definition of f and by property of $\langle\cdot,\cdot\rangle$:

$$\begin{cases} f(\widehat{y}_j) - f(x) = f_j(y_j) + \sum_{\substack{i=1\\i\neq j}}^{k} f_i(x_i) - \sum_{i=1}^{k} f_i(x_i) = f_j(y_j) - f_j(x_j) \\ \langle v, \widehat{y}_j - x\rangle = \langle v_j, y_j - x_j\rangle + \sum_{\substack{i=1\\i\neq j}}^{k} \langle v_i, x_i - x_i\rangle = \langle v_j, y_j - x_j\rangle. \end{cases}$$

We thus obtain:

$$\begin{aligned} v \in \partial f(x) &\implies \forall y_j \in \mathbb{R}^{n_j},\ f(\widehat{y}_j) - f(x) \geq \langle v, \widehat{y}_j - x\rangle \\ &\iff \forall y_j \in \mathbb{R}^{n_j},\ f_j(y_j) - f_j(x_j) \geq \langle v_j, y_j - x_j\rangle \\ &\iff v_j \in \partial f_j(x_j). \end{aligned}$$

$\Leftarrow$. For all $j \in [\![1,k]\!]$, define $\overline{f}_j : x \in \mathbb{R}^n \mapsto f_j(x_j)$ and $\overline{v}_j \coloneqq (0,\dots,0,v_j,0,\dots,0) \in \mathbb{R}^n$. Clearly, $f = \sum_{j=1}^{k} \overline{f}_j$ and $v = \sum_{j=1}^{k} \overline{v}_j$. Again by property of $\langle\cdot,\cdot\rangle$, we can write that for all $j \in [\![1,k]\!]$:

$$\forall z \in \mathbb{R}^n,\ \langle \overline{v}_j, z\rangle = \langle v_j, z_j\rangle + \sum_{\substack{i=1\\i\neq j}}^{k} \langle 0, z_i\rangle = \langle v_j, z_j\rangle,$$

and one can easily check that for all $j \in [\![1,k]\!]$:

$$\left[\forall y \in \mathbb{R}^n,\ \overline{f}_j(y) - \overline{f}_j(x) \geq \langle \overline{v}_j, y - x\rangle\right] \iff \left[\forall y_j \in \mathbb{R}^{n_j},\ f_j(y_j) - f_j(x_j) \geq \langle v_j, y_j - x_j\rangle\right].$$

We finally obtain:

$$\begin{aligned} v \in \partial f(x) &\iff \forall y \in \mathbb{R}^n,\ f(y) - f(x) \geq \langle v, y - x\rangle \\ &\iff \forall y \in \mathbb{R}^n,\ \sum_{j=1}^{k} \overline{f}_j(y) - \sum_{j=1}^{k} \overline{f}_j(x) \geq \left\langle \sum_{j=1}^{k} \overline{v}_j, y - x \right\rangle \\ &\iff \forall y \in \mathbb{R}^n,\ \sum_{j=1}^{k} \left(\overline{f}_j(y) - \overline{f}_j(x)\right) \geq \sum_{j=1}^{k} \langle \overline{v}_j, y - x\rangle \\ &\impliedby \forall j \in [\![1,k]\!], \forall y \in \mathbb{R}^n,\ \overline{f}_j(y) - \overline{f}_j(x) \geq \langle \overline{v}_j, y - x\rangle \\ &\iff \forall j \in [\![1,k]\!], \forall y_j \in \mathbb{R}^{n_j},\ f_j(y_j) - f_j(x_j) \geq \langle v_j, y_j - x_j\rangle \\ &\iff \forall j \in [\![1,k]\!],\ v_j \in \partial f_j(x_j). \end{aligned}$$

□

Lemma 37. *Given $l, m, n \in \mathbb{N}^*$, let $\|\cdot\|$ (respectively $\|\cdot\|'$) denote the Euclidean norm on $\mathbb{R}^n$ (respectively $\mathbb{R}^{lmn}$); also let $\alpha > 0$ and define the function $g : W \in \mathbb{R}^{l\times m\times n} \mapsto \sum_{i=0}^{l-1}\sum_{j=0}^{m-1} \left\|W[i,j,:]\right\|$. Then, the proximal operator of αg at point $A \in \mathbb{R}^{l\times m\times n}$ has the*

following closed-form solution:

$$\mathcal{P}rox(\alpha g)(A) = \Big(\mathcal{P}rox\big(\alpha\|\cdot\|\big)\big(A[i,j,:]\big)\Big)_{(i,j)\in[\![0,l-1]\!]\times[\![0,m-1]\!]}. \tag{5.18}$$

More precisely, one has for all $(i,j) \in [\![0,l-1]\!] \times [\![0,m-1]\!]$*:*

$$\mathcal{P}rox(\alpha g)(A)[i,j,:] = \begin{cases} \Big(\big\|A[i,j,:]\big\| - \alpha\Big) \cdot \frac{A[i,j,:]}{\big\|A[i,j,:]\big\|} & \text{if } \big\|A[i,j,:]\big\| > \alpha \\ 0 & \text{if } \big\|A[i,j,:]\big\| \leq \alpha. \end{cases} \tag{5.19}$$

Proof. In what follows, due to the fact that the metric spaces $\mathbb{R}^{lmn}$ and $\mathbb{R}^{l\times m\times n}$ (equipped with their respective Euclidean norm) are isometric, we will use the same notation $\|\cdot\|'$ to denote their respective Euclidean norm. Now, consider the function:

$$h : W \in \mathbb{R}^{l\times m\times n} \mapsto \alpha g(W) + \frac{1}{2}\|W - A\|'^2.$$

Notice that the function h can be written as a sum of functions over a decomposition of $\mathbb{R}^{lmn}$ (up to isometry):

$$\begin{aligned} \forall W \in \mathbb{R}^{l\times m\times n},\ h(W) &= \alpha \sum_{i=0}^{l-1}\sum_{j=0}^{m-1} \big\|W[i,j,:]\big\| + \frac{1}{2}\sum_{i=0}^{l-1}\sum_{j=0}^{m-1} \big\|W[i,j,:] - A[i,j,:]\big\|^2 \\ &= \sum_{i=0}^{l-1}\sum_{j=0}^{m-1} \Big(\alpha\big\|W[i,j,:]\big\| + \frac{1}{2}\big\|W[i,j,:] - A[i,j,:]\big\|^2\Big) \\ &= \sum_{i=0}^{l-1}\sum_{j=0}^{m-1} h_{i,j}\big(W[i,j,:]\big), \end{aligned}$$

where:

$$\forall (i,j) \in [\![0,l-1]\!] \times [\![0,m-1]\!],\ h_{i,j} : x \in \mathbb{R}^n \mapsto \alpha\|x\| + \frac{1}{2}\big\|x - A[i,j,:]\big\|^2.$$

We can therefore apply Lemma 36 to obtain:

$$\begin{aligned} &\forall (V,W) \in (\mathbb{R}^{l\times m\times n})^2, \\ &\qquad \big[V \in \partial h(W) \iff \forall (i,j) \in [\![0,l-1]\!] \times [\![0,m-1]\!],\ V[i,j,:] \in \partial h_{i,j}\big(W[i,j,:]\big)\big]. \end{aligned}$$

Setting $V = 0$ and $W = W_*$ in the previous statement, combined with the definition of

the proximal operator and the first clause of Lemma 27 thus gives:

$$\begin{aligned}
W_* = \mathcal{P}rox(\alpha g)(A) &\iff W_* = \operatorname*{argmin}_{W \in \mathbb{R}^{l \times m \times n}} \Big(\alpha g(W) + \frac{1}{2} \|W - A\|'^2 \Big) \\
&\iff 0 \in \partial \Big(\alpha g + \frac{1}{2} \| \cdot - A\|'^2 \Big)(W_*) \\
&\iff 0 \in \partial h(W_*), \\
&\iff \forall (i,j) \in [\![0, l-1]\!] \times [\![0, m-1]\!],\ 0 \in \partial h_{i,j}\big(W_*[i,j,:]\big) \\
&\iff \forall (i,j) \in [\![0, l-1]\!] \times [\![0, m-1]\!], \\
&\qquad\qquad 0 \in \partial \Big(\alpha \| \cdot \| + \frac{1}{2} \big\| \cdot - A[i,j,:] \big\|^2 \Big) \big(W_*[i,j,:]\big) \\
&\iff \forall (i,j) \in [\![0, l-1]\!] \times [\![0, m-1]\!], \\
&\qquad\qquad W_*[i,j,:] = \operatorname*{argmin}_{x \in \mathbb{R}^n} \Big(\alpha \|x\| + \frac{1}{2} \big\| x - A[i,j,:] \big\|^2 \Big) \\
&\iff \forall (i,j) \in [\![0, l-1]\!] \times [\![0, m-1]\!], \\
&\qquad\qquad W_*[i,j,:] = \mathcal{P}rox\big(\alpha \| \cdot \|\big)\big(A[i,j,:]\big).
\end{aligned}$$

We have just showed that:

$$\mathcal{P}rox(\alpha g)(A) = \Big(\mathcal{P}rox\big(\alpha \| \cdot \|\big)\big(A[i,j,:]\big) \Big)_{(i,j) \in [\![0,l-1]\!] \times [\![0,m-1]\!]}.$$

Finally, the precise closed-form solution is obtained directly from Lemma 35. □

5.6 Frobenius norm and spectral norm of matrices

In this short section, we reestablish the *sub-multiplicativity* of the Frobenius norm, that is, the Frobenius norm of a product of matrices is upper-bounded by the product of the Frobenius norms of these matrices. In fact, a finer analysis (conducted in Lemma 39) reveals that a tighter upper-bound is the product of the *spectral norm* of the first matrix with the Frobenius norm of the second matrix.

The spectral norm is a matrix norm that can be defined in two very different ways: it corresponds to the largest singular value of a matrix as well as the operator norm associated with the Euclidean norm (this dual form is proved in Lemma 38).

The sub-multiplicativity of the Frobenius norm is important in that it naturally arises while showing that the smooth component of a LASSO-like objective function has an L-Lipschitz continuous gradient (as in Lemma 41). A smaller Lipschitz constant for the gradient is equivalent to a larger learning rate using proximal gradient descent (see

Section 2.7) and is therefore of prime importance, hence why using the spectral norm to get a tighter sub-multiplicativity upper-bound matters.

Definition 25. *The **spectral norm** of a matrix $A \in \mathbb{R}^{m\times n}$, denoted $\|A\|_*$, is defined to be the largest singular value of A, that is the square root of the largest eigenvalue of the matrix A^tA:*

$$\|A\|_* = \sqrt{\lambda_{\max}(A^tA)}. \tag{5.20}$$

Lemma 38. *In finite dimensions, the spectral norm coincides with the operator norm associated with the Euclidean norm:*

$$\forall A \in \mathbb{R}^{m\times n},\ \|A\|_* = \sup_{x\in\mathbb{R}^n\setminus\{0\}} \frac{\|Ax\|}{\|x\|}. \tag{5.21}$$

Proof. In this proof, we will use the fact that $A^tA \in \mathbb{R}^{n\times n}$ is a positive semi-definite matrix such that its eigenvalues $(\lambda_k)_{k\in[\![1,n]\!]}$ are non-negative; for convenience we can assume the eigenvalues are sorted in increasing order, that is:

$$\lambda_{\max}(A^tA) = \lambda_n \geq \cdots \geq \lambda_1 \geq 0.$$

Besides, $A^tA \in \mathbb{R}^{n\times n}$ is a real and symmetric matrix, so the spectral theorem [Hawkins, 1975] ensures there is an orthonormal basis of $\mathbb{R}^n$ made of eigenvectors $(e_k)_{k\in[\![1,n]\!]}$:

$$\begin{cases} \forall k \in [\![1,n]\!], & A^tAe_k = \lambda_k e_k \quad \text{and} \quad \|e_k\| = 1 \\ \forall (j,k) \in [\![1,n]\!]^2 : j,k \text{ distinct}, & \langle e_j, e_k\rangle = 0. \end{cases}$$

$\geq$. Fix $x \in \mathbb{R}^n$. We can write x in the orthonormal basis $(e_k)_{k\in[\![1,n]\!]}$:

$$x = \sum_{k=1}^n \mu_k e_k \quad \text{and} \quad \|x\|^2 = \sum_{k=1}^n \mu_k^2.$$

Now, we have:

$$\begin{aligned} \|Ax\|^2 &= \langle Ax, Ax\rangle \\ &= \langle x, A^tAx\rangle \\ &= \left\langle \sum_{j=1}^n \mu_j e_j, A^tA \sum_{k=1}^n \mu_k e_k \right\rangle \\ &= \left\langle \sum_{j=1}^n \mu_j e_j, \sum_{k=1}^n \mu_k A^tAe_k \right\rangle \\ &= \left\langle \sum_{j=1}^n \mu_j e_j, \sum_{k=1}^n \mu_k \lambda_k e_k \right\rangle, \end{aligned}$$

hence by bilinearity of the scalar product:

$$\begin{aligned}\|Ax\|^2 &= \sum_{j=1}^{n}\sum_{k=1}^{n}\mu_j\mu_k\lambda_k\langle e_j, e_k\rangle \\ &= \sum_{k=1}^{n}\mu_k^2\lambda_k \\ &\leq \sum_{k=1}^{n}\mu_k^2\lambda_n \\ &= \lambda_n\cdot\sum_{k=1}^{n}\mu_k^2 \\ &= \|A\|_*^2\cdot\|x\|^2,\end{aligned}$$

therefore:

$$\forall x\in\mathbb{R}^n\setminus\{0\},\ \|A\|_*\geq\frac{\|Ax\|}{\|x\|},$$

which implies that:

$$\|A\|_*\geq\sup_{x\in\mathbb{R}^n\setminus\{0\}}\frac{\|Ax\|}{\|x\|}.$$

$\leq$. On the one hand, we have:

$$\begin{aligned}\|A\|_*^2 &= \lambda_n \\ &= \lambda_n\|e_n\| \\ &= \|\lambda_n e_n\| \\ &= \|A^tAe_n\|.\end{aligned}$$

On the other hand, since $(e_k)_{k\in[\![1,n]\!]}$ is an orthonormal basis we can use Parseval's identity, that is:

$$\forall x\in\mathbb{R}^n,\ \|x\|^2=\sum_{k=1}^{n}\langle x,e_k\rangle^2.$$

Substituting $x\coloneqq A^tAe_n$, we deduce:

$$\begin{aligned}\|A^tAe_n\|^2 &= \sum_{k=1}^{n}\langle A^tAe_n,e_k\rangle^2 \\ &= \sum_{k\neq n}\big(\lambda_n\langle e_n,e_k\rangle\big)^2+\langle A^tAe_n,e_n\rangle^2 \\ &= 0+\langle Ae_n,Ae_n\rangle^2 \\ &= \|Ae_n\|^4.\end{aligned}$$

Combining the two previous results yields $\|A\|_*=\sqrt{\|A^tAe_n\|}=\|Ae_n\|$. But since

$\|e_n\| = 1$, we clearly get:

$$\begin{aligned}\|A\|_* &= \frac{\|Ae_n\|}{\|e_n\|} \\ &\leq \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|Ax\|}{\|x\|}.\end{aligned}$$

□

Lemma 39. *The Frobenius norm is sub-multiplicative. More precisely, one has:*

$$\forall (A, B) \in \mathbb{R}^{m \times n} \times \mathbb{R}^{n \times p},\ \|AB\| \leq \|A\|_* \cdot \|B\| \leq \|A\| \cdot \|B\|. \tag{5.22}$$

Proof. From Lemma 38, we know that for a matrix $A \in \mathbb{R}^{m \times n}$, its spectral norm $\|A\|_*$ coincides with its operator norm associated with the Euclidean norm:

$$\|A\|_* = \sup_{x \in \mathbb{R}^n \setminus \{0\}} \frac{\|Ax\|}{\|x\|}.$$

One therefore has that $\|Ax\| \leq \|A\|_* \cdot \|x\|$ for all $x \in \mathbb{R}^n$. Now, given two matrices $(A, B) \in \mathbb{R}^{m \times n} \times \mathbb{R}^{n \times p}$, the j-th column of AB is Ab_j where $b_j \in \mathbb{R}^n$ is the j-th column of B. From this we deduce:

$$\begin{aligned}\|AB\|^2 &= \sum_{j=0}^{p-1} \big\|(AB)[:, j]\big\|^2 \\ &= \sum_{j=0}^{p-1} \|Ab_j\|^2 \\ &\leq \sum_{j=0}^{p-1} \|A\|_*^2 \cdot \|b_j\|^2 \\ &= \|A\|_*^2 \cdot \sum_{j=0}^{p-1} \big\|B[:, j]\big\|^2 \\ &= \|A\|_*^2 \cdot \|B\|^2,\end{aligned}$$

proving that $\|AB\| \leq \|A\|_* \cdot \|B\|$. Finally, by definition the spectral norm of $\|A\|_*$ is equal to the largest singular value of A, i.e. $\|A\|_* = \sqrt{\lambda_{\max}(A^t A)}$ where $\lambda_{\max}(A^t A)$ is the largest eigenvalue of $A^t A$. Let $(\lambda_k)_{k \in [\![1,n]\!]}$ denote the eigenvalues of $A^t A$ (again, these eigenvalues are non-negative since $A^t A$ is positive semi-definite). Since the trace of a

matrix is equal to the sum of the eigenvalues of that matrix, we can write:

$$\begin{aligned}\|A\|^2 &= \langle A, A\rangle \\ &= \mathcal{T}r(A^tA) \\ &= \sum_{k=1}^{n} \lambda_k \\ &\geq \lambda_{\max}(A^tA) \\ &= \|A\|_*^2,\end{aligned}$$

i.e. $\|A\|_* \leq \|A\|$. Combining results, we conclude $\|AB\| \leq \|A\|_* \cdot \|B\| \leq \|A\| \cdot \|B\|$. □

5.7 Stability analysis of ProxiMAS

In this section we write down for completeness the full theoretical convergence analysis of the ProxiMAS heuristic described in Algorithm 3.5, which had to be shortened in Article III due to space limitations.

More precisely: Lemma 40 gives a necessary condition for the sequence of permutations constructed by ProxiMAS to become constant after a finite number of iterations; Lemma 41 shows that the objective functions constructed by ProxiMAS have *stationary* properties that will be exploited in Lemma 42; the latter lemma gives a set of sufficient conditions for the sequence of cyclic solutions constructed by ProxiMAS to admit a converging subsequence. Finally, Theorem 1 gives a set of sufficient conditions ensuring certain subsequences of permutations constructed by ProxiMAS become constant after a finite number of iterations. The proof is slightly intricate and recursively applies an intermediary stability lemma (Lemma 43).

Lemma 40. *Let $\left(\widetilde{W}_k\right)_k$, $(W_k)_k$ and $(\pi_k)_k$ respectively denote the sequence of cyclic solutions, acyclic solutions and permutations in ProxiMAS. Assume permutations stabilize, i.e. $\exists k_1 : \forall k \geq k_1,\ \pi_k = \pi$. Then the following convergence condition necessarily holds:*

$$\exists k_0 : \forall k \geq k_0,\ \left\|\widetilde{W}_k - W_k\right\| \leq \left\|\widetilde{W}_k - W_{k-1}\right\|. \tag{5.23}$$

Proof. We start by writing:

$$\begin{aligned}\left\|\widetilde{W}_k - W_{k-1}\right\|^2 &= \left\|\widetilde{W}_k - W_k + W_k - W_{k-1}\right\|^2 \\ &= \left\|\widetilde{W}_k - W_k\right\|^2 + 2\left\langle\widetilde{W}_k - W_k, W_k - W_{k-1}\right\rangle + \left\|W_k - W_{k-1}\right\|^2,\end{aligned}$$

but since $\widetilde{W}_k - W_k$ and W_k are orthogonal by construction, this implies:

$$\begin{aligned}\left\|\widetilde{W}_k - W_k\right\|^2 \leq \left\|\widetilde{W}_k - W_{k-1}\right\|^2 &\iff 2\left\langle \widetilde{W}_k - W_k, W_k - W_{k-1}\right\rangle + \left\|W_k - W_{k-1}\right\|^2 \geq 0 \\ &\iff 2\left\langle \widetilde{W}_k - W_k, -W_{k-1}\right\rangle + \left\|W_k - W_{k-1}\right\|^2 \geq 0 \\ &\iff \left\|W_k - W_{k-1}\right\|^2 \geq 2\left\langle \widetilde{W}_k - W_k, W_{k-1}\right\rangle.\end{aligned}$$

Now due to our assumption, there exists k_1 such that for all $k \geq k_0 \coloneqq k_1 + 1$, $\pi_k = \pi_{k-1} = \pi$. Seeing π as a permutation operator, we get by construction of Algorithm 3.4 that for all $(i, j) \in [\![0, d-1]\!]^2$:

$$\forall k \geq k_0, \begin{cases} \widetilde{W}_k[i,j] - W_k[i,j] \neq 0 \implies \pi_k(i) \leq \pi_k(j) \iff \pi(i) \leq \pi(j) \\ W_{k-1}[i,j] \neq 0 \implies \pi_{k-1}(i) > \pi_{k-1}(j) \iff \pi(i) > \pi(j), \end{cases}$$

such that for all $(i, j) \in [\![0, d-1]\!]^2$:

$$\forall k \geq k_0, \ \left[\widetilde{W}_k[i,j] - W_k[i,j] \neq 0 \implies W_{k-1}[i,j] = 0\right].$$

The last result entails:

$$\forall k \geq k_0, \ \left\langle \widetilde{W}_k - W_k, W_{k-1}\right\rangle = 0,$$

therefore $\left\|W_k - W_{k-1}\right\|^2 \geq 2\left\langle \widetilde{W}_k - W_k, W_{k-1}\right\rangle$ trivially holds when $k \geq k_0$, and the claim follows. □

Lemma 41. *Let $(\phi_k)_k$ denote the sequence of objective functions in ProxiMAS. Then the functions ϕ_k are composite ρ-strongly convex functions of the form $\phi_k = f_k + g$ where both the f_k and g are convex and the f_k are differentiable with an L-Lipschitz continuous gradient where $L = \frac{1}{n}\|X^tX + n\rho I\|_*$ does not depend on k.*

Proof. The functions ϕ_k have the following form:

$$\phi_k : W \in \mathbb{R}^{d\times d} \mapsto \frac{1}{2n}\|XW - X\|^2 + \frac{\rho}{2}\|W - W_{k-1}\|^2 + \lambda\|W\|_1.$$

The fact that the ϕ_k are ρ-strongly convex follows immediately from Lemma 32, since the functions $W \in \mathbb{R}^{d\times d} \mapsto \phi_k(W) - \frac{\rho}{2}\|W - W_{k-1}\|^2$ are convex. Now, define the f_k and g as such:

$$f_k : W \in \mathbb{R}^{d\times d} \mapsto \frac{1}{2n}\|XW - X\|^2 + \frac{\rho}{2}\|W - W_{k-1}\|^2 \quad \text{and} \quad g : W \in \mathbb{R}^{d\times d} \mapsto \lambda\|W\|_1.$$

We clearly have that $\phi_k = f_k + g$ where both the f_k and g are convex and the f_k are

differentiable. Moreover, the gradient of the f_k at W takes the form:

$$\begin{aligned}\nabla f_k(W) &= \frac{1}{n}X^t(XW - X) + \rho(W - W_{k-1}) \\ &= \Big(\frac{1}{n}X^tX + \rho I\Big)W + C_k,\end{aligned}$$

where I is the identity matrix and $C_k \coloneqq -\big(\frac{1}{n}X^tX + \rho W_{k-1}\big)$ is constant with respect to the variable W. All is left is to investigate the Lipschitz continuity of the gradient of the smooth components f_k. Fix k and $(W, W') \in (\mathbb{R}^{d\times d})^2$; we have thanks to Lemma 39:

$$\begin{aligned}\big\|\nabla f_k(W) - \nabla f_k(W')\big\| &= \left\|\Big(\frac{1}{n}X^tX + \rho I\Big)W + C_k - \Big(\frac{1}{n}X^tX + \rho I\Big)W' - C_k\right\| \\ &= \frac{1}{n}\big\|(X^tX + n\rho I)(W - W')\big\| \\ &\leq \frac{1}{n}\|X^tX + n\rho I\|_* \cdot \|W - W'\|,\end{aligned}$$

which shows that the functions f_k have L-Lipschitz continuous gradient where $L \coloneqq \frac{1}{n}\|X^tX + n\rho I\|_*$ does not depend on k. □

Lemma 42. *Let $\big(\widetilde{W}_k\big)_k$, $(W_k)_k$ and $(\gamma_k)_k$ respectively denote the sequence of cyclic solutions, acyclic solutions and learning rates in ProxiMAS. Assume Lemma 40's condition holds: $\exists k_0 : \forall k \geq k_0,\ \big\|\widetilde{W}_k - W_k\big\| \leq \big\|\widetilde{W}_k - W_{k-1}\big\|$. Also assume one of the two following conditions on the learning rate:*

$$\begin{cases} \gamma_k = \mathcal{O}\big(\frac{1}{k^\alpha}\big) \text{ where } \alpha > 2 \\ \text{or} \\ \gamma_k = \mathcal{O}\big(\frac{1}{k^\alpha}\big) \text{ where } \alpha > 1 \quad \text{and} \quad k\gamma_k^{-1}\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\|^2 \underset{k\to+\infty}{\longrightarrow} l \in \mathbb{R}_+ \cup \{+\infty\}. \end{cases} \tag{5.24}$$

Then $\big(\widetilde{W}_k\big)_k$ admits a convergent subsequence.

Proof. We know from Lemma 41 that the ϕ_k have stationary properties, namely they are composite convex functions of the form $\phi_k = f_k + g$ with $g = \lambda\|\cdot\|_1$ and the differentiable components f_k have L-Lipschitz continuous gradient where L does not depend on k. Now, the $\widetilde{W}_k$ are generated according to Algorithm 3.5: line 5, that is:

$$\forall k \geq 1,\ \widetilde{W}_k = \mathcal{P}rox(\gamma_k g)\Big(\widetilde{W}_{k-1} - \gamma_k \nabla f_k\big(\widetilde{W}_{k-1}\big)\Big) \quad \text{where} \quad \gamma_k \in \left]0, L^{-1}\right],$$

hence applying Lemma 10 (the so-called *descent lemma*) to every ϕ_k at step k yields:

$$\forall k \geq 1,\ \phi_k\big(\widetilde{W}_k\big) \leq \phi_k\big(\widetilde{W}_{k-1}\big) - \frac{\gamma_k^{-1}}{2}\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\|^2.$$

In particular, $\phi_k\big(\widetilde{W}_k\big) \leq \phi_k\big(\widetilde{W}_{k-1}\big)$ holds for all $k \geq 1$. Furthermore, notice that by

definition of the ϕ_k one has:

$$\phi_k\big(\widetilde{W}_{k-1}\big) = \phi_{k-1}\big(\widetilde{W}_{k-1}\big) + \frac{\rho}{2}\Big(\big\|\widetilde{W}_{k-1} - W_{k-1}\big\|^2 - \big\|\widetilde{W}_{k-1} - W_{k-2}\big\|^2\Big).$$

Now due to Lemma 40's condition, we have:

$$\forall k \geq k_1 \coloneqq k_0 + 1,\ \big\|\widetilde{W}_{k-1} - W_{k-1}\big\|^2 - \big\|\widetilde{W}_{k-1} - W_{k-2}\big\|^2 \leq 0,$$

hence we get that $\phi_k\big(\widetilde{W}_k\big) \leq \phi_k\big(\widetilde{W}_{k-1}\big) \leq \phi_{k-1}\big(\widetilde{W}_{k-1}\big)$ for all $k \geq k_1$. The latter implies that the (non-negative) sequence $\big(\phi_k\big(\widetilde{W}_k\big)\big)_k$ converges to a limit $l \geq 0$. Additionally, this lets us write as well:

$$\begin{aligned}\forall k \geq k_1,\ \frac{\gamma_k^{-1}}{2}\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\|^2 &\leq \phi_k\big(\widetilde{W}_{k-1}\big) - \phi_k\big(\widetilde{W}_k\big)\\ &\leq \phi_{k-1}\big(\widetilde{W}_{k-1}\big) - \phi_k\big(\widetilde{W}_k\big).\end{aligned}$$

We then use the fact that the right-hand side in the last inequality is a telescopic term, along with $\phi_k\big(\widetilde{W}_k\big) \underset{k\to+\infty}{\longrightarrow} l$, to deduce that the infinite series $\sum_k \gamma_k^{-1}\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\|^2$ converges. In particular, $\gamma_k^{-1}\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\|^2 = o(1)$ holds, such that $\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\| = \mathcal{O}\big(\sqrt{\gamma_k}\big)$; with the additional assumption that $k\gamma_k^{-1}\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\|^2 \underset{k\to+\infty}{\longrightarrow} l \in \mathbb{R}_+ \cup \{+\infty\}$, then $\gamma_k^{-1}\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\|^2 = o\big(\frac{1}{k}\big)$ must hold instead (due to the fact that the harmonic series diverges), in which case $\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\| = \mathcal{O}\big(\sqrt{\frac{\gamma_k}{k}}\big)$. In the first case, with $\gamma_k = \mathcal{O}\big(\frac{1}{k^\alpha}\big)$ where $\alpha > 2$ we have $\sqrt{\gamma_k} = \mathcal{O}\big(\frac{1}{k^{\alpha/2}}\big)$, hence $\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\| = \mathcal{O}\big(\frac{1}{k^\beta}\big)$ where $\beta \coloneqq \frac{\alpha}{2} > 1$. In the second case (with the additional assumption) $\alpha > 1$ suffices to recover the same upper bound: we now have $\sqrt{\frac{\gamma_k}{k}} = \mathcal{O}\big(\frac{1}{k^{(\alpha+1)/2}}\big)$, hence $\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\| = \mathcal{O}\big(\frac{1}{k^\beta}\big)$ where $\beta \coloneqq \frac{\alpha+1}{2} > 1$. In both cases, we obtain that the infinite series $S \coloneqq \sum_k \big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\|$ converges. The triangular inequality finally yields:

$$\begin{aligned}\forall k \geq 1,\ \big\|\widetilde{W}_k - \widetilde{W}_0\big\| &= \Big\|\sum_{j=1}^{k} \widetilde{W}_j - \widetilde{W}_{j-1}\Big\|\\ &\leq \sum_{j=1}^{k} \big\|\widetilde{W}_j - \widetilde{W}_{j-1}\big\|\\ &\leq S < +\infty,\end{aligned}$$

thus $\sup_{k\geq 1} \big\|\widetilde{W}_k\big\| < +\infty$. The Bolzano-Weierstrass theorem concludes the proof. □

Lemma 43. *Let $\big(\widetilde{W}_k\big)_k$ and $(\pi_k)_k$ respectively denote the sequence of cyclic solutions and permutations in ProxiMAS. Assume $\big(\widetilde{W}_k\big)_k$ admits a converging subsequence: $\widetilde{W}_* \coloneqq \lim_{k\to+\infty} \widetilde{W}_{\psi(k)}$. Define π_* to be the permutation constructed by Algorithm 3.4 given $\widetilde{W}_*$ as input. Given $r \in [\![1, d]\!]$, let $V_{\psi(k),r}$ and $V_{*,r}$ denote the set of remaining nodes from*

which Algorithm 3.4 will select the r-th rightmost elements $\pi_{\psi(k)}[d-r]$ and $\pi_*[d-r]$, respectively. Assume the two following assumptions hold:

- *Algorithm 3.4 makes a strictly optimal decision when constructing $\pi_*[d-r]$ (i.e. argmin in Algorithm 3.4: line 3 is strict at step r given $\widetilde{W}_*$ as input).*
- $\exists k'_{r-1} : \forall k \geq k'_{r-1},\ V_{\psi(k),r} = V_{*,r}$.

Then, the r-th rightmost elements of the permutations constructed by ProxiMAS in the subsequence ψ stabilize after a finite number of iterations:

$$\exists k'_r \geq k'_{r-1} : \forall k \geq k'_r,\ \pi_{\psi(k)}[d-r] = \pi_*[d-r]. \tag{5.25}$$

Proof. For convenience, write $j_r \coloneqq \pi_*[d-r]$. By construction of Algorithm 3.4, it suffices to show:

$$\exists k'_r \geq k'_{r-1} : \forall k \geq k'_r, \forall j \in V_{\psi(k),r} \setminus \{j_r\}, \\ \big\|\widetilde{W}_{\psi(k)}\big[V_{\psi(k),r} \setminus \{j_r\}, j_r\big]\big\|^2 < \big\|\widetilde{W}_{\psi(k)}\big[V_{\psi(k),r} \setminus \{j\}, j\big]\big\|^2.$$

Since by assumption $V_{\psi(k),r} = V_{*,r}$ when $k \geq k'_{r-1}$, we can write that for all $k \geq k'_{r-1}$:

$$\begin{aligned}\big\|\widetilde{W}_{\psi(k)}\big[V_{\psi(k),r} \setminus \{j\}, j\big]\big\|^2 &= \big\|\widetilde{W}_{\psi(k)}\big[V_{*,r} \setminus \{j\}, j\big]\big\|^2 \\ &= \big\|\big(\widetilde{W}_* + \widetilde{W}_{\psi(k)} - \widetilde{W}_*\big)\big[V_{*,r} \setminus \{j\}, j\big]\big\|^2 \\ &= \big\|\widetilde{W}_*\big[V_{*,r} \setminus \{j\}, j\big]\big\|^2 + \big\|\big(\widetilde{W}_{\psi(k)} - \widetilde{W}_*\big)\big[V_{*,r} \setminus \{j\}, j\big]\big\|^2 \\ &\quad + 2\big\langle\widetilde{W}_*\big[V_{*,r} \setminus \{j\}, j\big], \big(\widetilde{W}_{\psi(k)} - \widetilde{W}_*\big)\big[V_{*,r} \setminus \{j\}, j\big]\big\rangle.\end{aligned}$$

Applying the Cauchy-Schwarz inequality thus yields for all $k \geq k'_{r-1}$:

$$\begin{cases} \big\|\widetilde{W}_{\psi(k)}\big[V_{\psi(k),r} \setminus \{j\}, j\big]\big\|^2 & \geq x_j^2 + y_{\psi(k),j}^2 - 2My_{\psi(k),j} \quad \forall j \in V_{\psi(k),r} \setminus \{j_r\} \\ \big\|\widetilde{W}_{\psi(k)}\big[V_{\psi(k),r} \setminus \{j_r\}, j_r\big]\big\|^2 & \leq x_{j_r}^2 + y_{\psi(k),j_r}^2 + 2My_{\psi(k),j_r}, \end{cases}$$

where $x_j \coloneqq \big\|\widetilde{W}_*\big[V_{*,r}\setminus\{j\}, j\big]\big\|$, $y_{\psi(k),j} \coloneqq \big\|\big(\widetilde{W}_{\psi(k)} - \widetilde{W}_*\big)\big[V_{*,r}\setminus\{j\}, j\big]\big\|$, $M \coloneqq \big\|\widetilde{W}_*\big\| < \infty$. It is therefore stronger to show instead:

$$\exists k'_r \geq k'_{r-1} : \forall k \geq k'_r, \forall j \in V_{\psi(k),r}\setminus\{j_r\},\ x_{j_r}^2 + y_{\psi(k),j_r}^2 + 2My_{\psi(k),j_r} < x_j^2 + y_{\psi(k),j}^2 - 2My_{\psi(k),j}.$$

Rearranging the terms and using again that $V_{\psi(k),r} = V_{*,r}$ for all $k \geq k'_{r-1}$, the last condition becomes:

$$\exists k'_r \geq k'_{r-1} : \forall k \geq k'_r, \forall j \in V_{*,r} \setminus \{j_r\},\ y_{\psi(k),j_r}^2 - y_{\psi(k),j}^2 + 2M(y_{\psi(k),j_r} + y_{\psi(k),j}) < x_j^2 - x_{j_r}^2.$$

Let us now formalize our assumption that Algorithm 3.4 makes a strictly optimal decision when constructing $\pi_*[d-r]$. This means:

$$\begin{aligned} x_{j_r}^2 &= \left\|\widetilde{W}_*\big[V_{*,r} \setminus \{j_r\}, j_r\big]\right\|^2 \\ &< \min_{j \in V_{*,r} \setminus \{j_r\}} \left\|\widetilde{W}_*\big[V_{*,r} \setminus \{j\}, j\big]\right\|^2 \\ &= \min_{j \in V_{*,r} \setminus \{j_r\}} x_j^2, \end{aligned}$$

which can be rewritten $\delta_r \coloneqq \min_{j \in V_{*,r} \setminus \{j_r\}} x_j^2 - x_{j_r}^2 > 0$. Now, it is clearly sufficient for us to satisfy the even stronger condition:

$$\exists k'_r \geq k'_{r-1} : \forall k \geq k'_r, \forall j \in V_{*,r} \setminus \{j_r\},\ y_{\psi(k),j_r}^2 - y_{\psi(k),j}^2 + 2M(y_{\psi(k),j_r} + y_{\psi(k),j}) < \delta_r.$$

But the last condition holds since $\delta_r > 0$, the set $V_{*,r}$ is finite and $y_{\psi(k),j} \underset{k \to +\infty}{\longrightarrow} 0$ for all j in $V_{*,r}$. □

Theorem 1. *Let $\big(\widetilde{W}_k\big)_k$ and $(\pi_k)_k$ respectively denote the sequence of cyclic solutions and permutations in ProxiMAS. Assume $\big(\widetilde{W}_k\big)_k$ admits a converging subsequence: $\widetilde{W}_* \coloneqq \lim_{k \to +\infty} \widetilde{W}_{\psi(k)}$. Define π_* to be the permutation constructed by Algorithm 3.4 given $\widetilde{W}_*$ as input. Assume the following assumption holds:*

- *For all $r \in [\![1, d]\!]$, Algorithm 3.4 makes a strictly optimal decision when constructing $\pi_*[d-r]$ (i.e. argmin in Algorithm 3.4: line 3 is strict at every step r given $\widetilde{W}_*$ as input).*

Then the permutations constructed by ProxiMAS in the subsequence ψ stabilize after a finite number of iterations:

$$\exists k' : \forall k \geq k',\ \pi_{\psi(k)} = \pi_*. \tag{5.26}$$

Proof. Given $r \in [\![1, d]\!]$, let $V_{\psi(k),r}$ and $V_{*,r}$ denote the set of remaining nodes from which Algorithm 3.4 will select the r-th rightmost elements $\pi_{\psi(k)}[d-r]$ and $\pi_*[d-r]$, respectively. The proof will be complete if we can establish the following:

$$\forall r \in [\![1, d]\!],\ \exists k'_r : \forall k \geq k'_r,\ \pi_{\psi(k)}[d-r{:}] = \pi_*[d-r{:}].$$

Indeed, one will then have that $\pi_{\psi(k)} = \pi_*$ for all $k \geq k'_d$. The proof uses a recursive argument. For the initialization (i.e. $r = 1$), notice one always has that $V_{\psi(k),1} = V_{*,1} = V$ for all $k \geq k'_0 \coloneqq 1$ where V is the full node set. By assumption, Algorithm 3.4 makes a

strictly optimal decision when constructing $\pi_*[d-1]$, thus Lemma 43 yields:

$$\exists k'_1 \geq k'_0 : \forall k \geq k'_1,\ \pi_{\psi(k)}[d-1] = \pi_*[d-1],$$

hence $\pi_{\psi(k)}[d-1:] = \pi_*[d-1:]$ for all $k \geq k'_1$. Now we proceed with the recursion itself: assume there is k'_{r-1} such that $\pi_{\psi(k)}[d-(r-1):] = \pi_*[d-(r-1):]$ for all $k \geq k'_{r-1}$. In particular, $V_{\psi(k),r} = V_{*,r}$ for all $k \geq k'_{r-1}$. Again by assumption, Algorithm 3.4 makes a strictly optimal decision when constructing $\pi_*[d-r]$, thus Lemma 43 yields:

$$\exists k'_r \geq k'_{r-1} : \forall k \geq k'_r,\ \pi_{\psi(k)}[d-r] = \pi_*[d-r],$$

hence $\pi_{\psi(k)}[d-r:] = \pi_*[d-r:]$ for all $k \geq k'_r$. The recursion is proved and the claim follows. □

Bibliography

N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5):23:1–23:27, 2008. doi: 10.1145/1411509.1411513. URL https://doi.org/10.1145/1411509.1411513.

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. doi: 10.1109/TAC.1974.1100705. URL https://doi.org/10.1109/TAC.1974.1100705.

B. Aragam, J. Gu, and Q. Zhou. Learning Large-Scale Bayesian Networks with the sparsebn Package. *CoRR*, abs/1703.04025, 2017. URL http://arxiv.org/abs/1703.04025.

G. Ausiello, P. Crescenzi, and M. Protasi. Approximate Solution of NP Optimization Problems. *Theor. Comput. Sci.*, 150(1):1–55, 1995. doi: 10.1016/0304-3975(94)00291-P. URL https://doi.org/10.1016/0304-3975(94)00291-P.

A. Baharev, H. Schichl, A. Neumaier, and T. Achterberg. An Exact Method for the Minimum Feedback Arc Set Problem. *ACM J. Exp. Algorithmics*, 26:1.4:1–1.4:28, 2021. doi: 10.1145/3446429. URL https://doi.org/10.1145/3446429.

G. A. Barnard. A New Test for 2 × 2 Tables. *Nature*, 156:177, 1945. doi: 10.1038/156177a0.

L. E. Baum and T. Petrie. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 37(6):1554 – 1563, 1966. doi: 10.1214/aoms/1177699147. URL https://doi.org/10.1214/aoms/1177699147.

A. Beck and M. Teboulle. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM J. Imaging Sci.*, 2(1):183–202, 2009. doi: 10.1137/080716542. URL https://doi.org/10.1137/080716542.

B. Berger and P. W. Shor. Approximation Algorithms for the Maximum Acyclic Subgraph Problem. In D. S. Johnson, editor, *Proceedings of the First Annual ACM-*

SIAM Symposium on Discrete Algorithms, 22-24 January 1990, San Francisco, California, USA, pages 236–243. SIAM, 1990. URL http://dl.acm.org/citation.cfm?id=320176.320203.

K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz, J. van Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, L. Gottwald, C. Graczyk, K. Halbig, A. Hoen, C. Hojny, R. van der Hulst, T. Koch, M. Lübbecke, S. J. Maher, F. Matter, E. Mühmer, B. Müller, M. E. Pfetsch, D. Rehfeldt, S. Schlein, F. Schlösser, F. Serrano, Y. Shinano, B. Sofranac, M. Turner, S. Vigerske, F. Wegscheider, P. Wellner, D. Weninger, and J. Witzig. The SCIP Optimization Suite 8.0. ZIB-Report 21-41, Zuse Institute Berlin, 2021. URL http://nbn-resolving.de/urn:nbn:de:0297-zib-85309.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003. URL http://jmlr.org/papers/v3/blei03a.html.

H. L. Bodlaender, F. V. Fomin, A. M. C. A. Koster, D. Kratsch, and D. M. Thilikos. A Note on Exact Algorithms for Vertex Ordering Problems on Graphs. *Theory Comput. Syst.*, 50(3):420–432, 2012. doi: 10.1007/s00224-011-9312-0. URL https://doi.org/10.1007/s00224-011-9312-0.

M. Bonamy, L. Kowalik, J. Nederlof, M. Pilipczuk, A. Socala, and M. Wrochna. On Directed Feedback Vertex Set Parameterized by Treewidth. In A. Brandstädt, E. Köhler, and K. Meer, editors, *Graph-Theoretic Concepts in Computer Science - 44th International Workshop, WG 2018, Cottbus, Germany, June 27-29, 2018, Proceedings*, volume 11159 of *Lecture Notes in Computer Science*, pages 65–78. Springer, 2018. doi: 10.1007/978-3-030-00256-5_6. URL https://doi.org/10.1007/978-3-030-00256-5_6.

R. D. Boschloo. Raised conditional level of significance for the 2 × 2-table when testing the equality of two probabilities. *Statistica Neerlandica*, 24(1):1–9, 1970. doi: https://doi.org/10.1111/j.1467-9574.1970.tb00104.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9574.1970.tb00104.x.

S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2014. ISBN 978-0-521-83378-3. doi: 10.1017/CBO9780511804441. URL https://web.stanford.edu/%7Eboyd/cvxbook/.

F.-J. Brandenburg and K. Hanauer. Sorting Heuristics for the Feedback Arc Set Problem. Technical report, University of Passau, 2011. URL https://www.fim.uni-passau.de/fileadmin/dokumente/fakultaeten/fim/forschung/mip-berichte/mip1104.pdf.

A. Chambolle and C. Dossal. On the Convergence of the Iterates of the "Fast Iterative Shrinkage/Thresholding Algorithm". *J. Optim. Theory Appl.*, 166(3):968–982, 2015. doi: 10.1007/s10957-015-0746-4. URL https://doi.org/10.1007/s10957-015-0746-4.

M. Charikar, K. Makarychev, and Y. Makarychev. On the Advantage over Random for Maximum Acyclic Subgraph. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 625–633. IEEE Computer Society, 2007. doi: 10.1109/FOCS.2007.47. URL https://doi.org/10.1109/FOCS.2007.47.

J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5):21:1–21:19, 2008. doi: 10.1145/1411509.1411511. URL https://doi.org/10.1145/1411509.1411511.

D. M. Chickering. Learning Bayesian Networks is NP-Complete. In D. Fisher and H. Lenz, editors, *Learning from Data - Fifth International Workshop on Artificial Intelligence and Statistics, AISTATS 1995, Key West, Florida, USA, January, 1995. Proceedings*, pages 121–130. Springer, 1995. doi: 10.1007/978-1-4612-2404-4_12. URL https://doi.org/10.1007/978-1-4612-2404-4_12.

D. M. Chickering. Optimal Structure Identification With Greedy Search. *J. Mach. Learn. Res.*, 3:507–554, 2002. URL http://jmlr.org/papers/v3/chickering02b.html.

D. Colombo and M. H. Maathuis. Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.*, 15(1):3741–3782, 2014. doi: 10.5555/2627435.2750365. URL https://dl.acm.org/doi/10.5555/2627435.2750365.

G. F. Cooper. The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks. *Artif. Intell.*, 42(2-3):393–405, 1990. doi: 10.1016/0004-3702(90)90060-D. URL https://doi.org/10.1016/0004-3702(90)90060-D.

G. F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Mach. Learn.*, 9:309–347, 1992. doi: 10.1007/BF00994110. URL https://doi.org/10.1007/BF00994110.

J. Cussens. Bayesian network learning with cutting planes. In F. G. Cozman and A. Pfeffer, editors, *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011*, pages 153–160. AUAI Press, 2011. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2197&proceeding_id=27.

P. Dagum and M. Luby. Approximating Probabilistic Inference in Bayesian Belief Networks is NP-Hard. *Artif. Intell.*, 60(1):141–153, 1993. doi: 10.1016/0004-3702(93)90036-B. URL https://doi.org/10.1016/0004-3702(93)90036-B.

G. B. Dantzig. *Origins of the Simplex Method*, page 141–151. Association for Computing Machinery, 1990. ISBN 0201508141. URL https://doi.org/10.1145/87252.88081.

S. Dasgupta. Learning Polytrees. In K. B. Laskey and H. Prade, editors, *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pages 134–141. Morgan Kaufmann, 1999. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=162&proceeding_id=15.

L. M. de Campos. A Scoring Function for Learning Bayesian Networks based on Mutual Information and Conditional Independence Tests. *J. Mach. Learn. Res.*, 7:2149–2187, 2006. URL http://jmlr.org/papers/v7/decampos06a.html.

L. M. de Campos, J. M. Fernández-Luna, and J. F. Huete. Bayesian networks and information retrieval: an introduction to the special issue. *Inf. Process. Manag.*, 40 (5):727–733, 2004. doi: 10.1016/j.ipm.2004.03.001. URL https://doi.org/10.1016/j.ipm.2004.03.001.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. doi: https://doi.org/10.1111/j.2517-6161.1977.tb01600.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1977.tb01600.x.

L. Denoyer and P. Gallinari. Bayesian network model for semi-structured document classification. *Inf. Process. Manag.*, 40(5):807–827, 2004. doi: 10.1016/j.ipm.2004.04.009. URL https://doi.org/10.1016/j.ipm.2004.04.009.

S. Dong and M. Sebag. From Graphs to DAGs: A Low-Complexity Model and a Scalable Algorithm. In M. Amini, S. Canu, A. Fischer, T. Guns, P. K. Novak, and G. Tsoumakas, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2022, Grenoble, France, September 19-23, 2022, Proceedings, Part V*, volume 13717 of *Lecture Notes in Computer Science*, pages 107–122. Springer, 2022. doi: 10.1007/978-3-031-26419-1_7. URL https://doi.org/10.1007/978-3-031-26419-1_7.

R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. ISBN 978-1-4471-5558-4. doi: 10.1007/978-1-4471-5559-1. URL https://doi.org/10.1007/978-1-4471-5559-1.

P. Eades, X. Lin, and W. F. Smyth. A Fast and Effective Heuristic for the Feedback Arc Set Problem. *Inf. Process. Lett.*, 47(6):319–323, 1993. doi: 10.1016/0020-0190(93)90079-O. URL https://doi.org/10.1016/0020-0190(93)90079-O.

S. Even and G. Even. *Graph Algorithms, Second Edition*. Cambridge University Press, 2012. ISBN 978-0-521-73653-4. URL http://www.cambridge.org/us/academic/subjects/computer-science/algorithmics-complexity-computer-algebra-and-computational-g/graph-algorithms-2nd-edition.

H. Fernau and D. Raible. Exact Algorithms for Maximum Acyclic Subgraph on a Superclass of Cubic Graphs. In S. Nakano and M. S. Rahman, editors, *WALCOM: Algorithms and Computation, Second International Workshop, WALCOM 2008, Dhaka, Bangladesh, February 7-8, 2008*, volume 4921 of *Lecture Notes in Computer Science*, pages 144–156. Springer, 2008. doi: 10.1007/978-3-540-77891-2_14. URL https://doi.org/10.1007/978-3-540-77891-2_14.

P. Festa, P. M. Pardalos, and M. G. C. Resende. Feedback Set Problems. In D. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, pages 209–258. Springer, 1999. doi: 10.1007/978-1-4757-3023-4_4. URL https://doi.org/10.1007/978-1-4757-3023-4_4.

R. A. Fisher. *Statistical Methods for Research Workers*, pages 66–70. Springer New York, 1992. ISBN 978-1-4612-4380-9. doi: 10.1007/978-1-4612-4380-9_6. URL https://doi.org/10.1007/978-1-4612-4380-9_6.

E. Giudice, J. Kuipers, and G. Moffa. The Dual PC Algorithm for Structure Learning. In A. Salmerón and R. Rumí, editors, *International Conference on Probabilistic Graphical Models, PGM 2022, 5-7 October 2022, Almería, Spain*, volume 186 of *Proceedings of Machine Learning Research*, pages 301–312. PMLR, 2022. URL https://proceedings.mlr.press/v186/giudice22a.html.

J. Gondzio. Interior point methods 25 years later. *Eur. J. Oper. Res.*, 218(3):587–601, 2012. doi: 10.1016/j.ejor.2011.09.017. URL https://doi.org/10.1016/j.ejor.2011.09.017.

M. Grötschel, M. Jünger, and G. Reinelt. On the acyclic subgraph polytope. *Math. Program.*, 33(1):28–42, 1985. doi: 10.1007/BF01582009. URL https://doi.org/10.1007/BF01582009.

M. Grötschel, M. Jünger, and G. Reinelt. Comments on "An Exact Method for the Minimum Feedback Arc Set Problem". *ACM J. Exp. Algorithmics*, 27:1.3:1–1.3:4, 2022. doi: 10.1145/3545001. URL https://doi.org/10.1145/3545001.

Y. Gurevich and S. Shelah. Expected Computation Time for Hamiltonian Path Problem. *SIAM J. Comput.*, 16(3):486–502, 1987. doi: 10.1137/0216034. URL https://doi.org/10.1137/0216034.

Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*, 2023. URL https://www.gurobi.com.

T. Hawkins. Cauchy and the spectral theory of matrices. *Historia Mathematica*, 2(1):1–29, 1975. ISSN 0315-0860. doi: https://doi.org/10.1016/0315-0860(75)90032-4. URL https://www.sciencedirect.com/science/article/pii/0315086075900324.

M. Hecht. Exact Localisations of Feedback Sets. *Theory Comput. Syst.*, 62(5):1048–1084, 2018. doi: 10.1007/s00224-017-9777-6. URL https://doi.org/10.1007/s00224-017-9777-6.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Mach. Learn.*, 20(3):197–243, 1995. doi: 10.1007/BF00994016. URL https://doi.org/10.1007/BF00994016.

M. Held and R. M. Karp. A dynamic programming approach to sequencing problems. In T. C. Rowan, editor, *Proceedings of the 16th ACM national meeting, ACM 1961, USA*, page 71. ACM, 1961. doi: 10.1145/800029.808532. URL https://doi.org/10.1145/800029.808532.

R. Hemmecke, S. Lindner, and M. Studený. Characteristic imsets for learning Bayesian network structure. *Int. J. Approx. Reason.*, 53(9):1336–1349, 2012. doi: 10.1016/j.ijar.2012.04.001. URL https://doi.org/10.1016/j.ijar.2012.04.001.

ILOG CPLEX Optimization Studio. *The CPLEX Python API Reference Manual. Version 22.1.1*, 2022. URL https://www.ibm.com/docs/en/icos/22.1.1?topic=optimizers-cplex-python-api-reference-manual.

T. S. Jaakkola, D. A. Sontag, A. Globerson, and M. Meila. Learning Bayesian Network Structure using LP Relaxations. In Y. W. Teh and D. M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 358–365. JMLR.org, 2010. URL http://proceedings.mlr.press/v9/jaakkola10a.html.

D. B. Johnson. Finding All the Elementary Circuits of a Directed Graph. *SIAM J. Comput.*, 4(1):77–84, 1975. doi: 10.1137/0204007. URL https://doi.org/10.1137/0204007.

N. Karmarkar. A new polynomial-time algorithm for linear programming. *Comb.*, 4(4):373–396, 1984. doi: 10.1007/BF02579150. URL https://doi.org/10.1007/BF02579150.

R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J. W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi: 10.1007/978-1-4684-2001-2_9. URL https://doi.org/10.1007/978-1-4684-2001-2_9.

C. Kenyon-Mathieu and W. Schudy. How to rank with few errors: A PTAS for Weighted Feedback Arc Set on Tournaments. *Electron. Colloquium Comput. Complex.*, TR06-144, 2006. URL https://eccc.weizmann.ac.il/eccc-reports/2006/TR06-144/index.html.

L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979. ISSN 0002-3264. URL https://zbmath.org/?q=an:0414.90086.

R. Kindermann. *Markov random fields and their applications*. Contemporary mathematics v. 1. American Mathematical Society, 1980. ISBN 0-8218-5001-6.

M. Koivisto and P. Parviainen. A Space-Time Tradeoff for Permutation Problems. In M. Charikar, editor, *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 484–492. SIAM, 2010. doi: 10.1137/1.9781611973075.41. URL https://doi.org/10.1137/1.9781611973075.41.

M. Koivisto and K. Sood. Exact Bayesian Structure Discovery in Bayesian Networks. *J. Mach. Learn. Res.*, 5:549–573, 2004. URL http://jmlr.org/papers/volume5/koivisto04a/koivisto04a.pdf.

D. Koller and N. Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009. ISBN 978-0-262-01319-2. URL http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=11886.

C. L. Koumenides. *A Bayesian network model for entity-oriented semantic web search*. PhD thesis, University of Southampton, UK, 2013. URL http://eprints.soton.ac.uk/362651/.

R. Kudelic. Monte-Carlo randomized algorithm for minimal feedback arc set problem. *Appl. Soft Comput.*, 41:235–246, 2016. doi: 10.1016/j.asoc.2015.12.018. URL https://doi.org/10.1016/j.asoc.2015.12.018.

R. Kudelic and N. Ivkovic. Ant inspired Monte Carlo algorithm for minimum feedback arc set. *Expert Syst. Appl.*, 122:108–117, 2019. doi: 10.1016/j.eswa.2018.12.021. URL https://doi.org/10.1016/j.eswa.2018.12.021.

A. H. Land and A. G. Doig. An Automatic Method for Solving Discrete Programming Problems. In M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, editors, *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 105–132. Springer, 2010. doi: 10.1007/978-3-540-68279-0_5. URL https://doi.org/10.1007/978-3-540-68279-0_5.

R. Lazimy. Mixed-integer quadratic programming. *Math. Program.*, 22(1):332–349, 1982. doi: 10.1007/BF01581047. URL https://doi.org/10.1007/BF01581047.

J. Liang, T. Luo, and C. Schönlieb. Improving "Fast Iterative Shrinkage-Thresholding Algorithm": Faster, Smarter, and Greedier. *SIAM J. Sci. Comput.*, 44(3):1069, 2022. doi: 10.1137/21m1395685. URL https://doi.org/10.1137/21m1395685.

B. G. Lindsay. *Mixture models : theory, geometry, and applications*. NSF-CBMS regional conference series in probability and statistics ; v. 5. Institute of Mathematical Statistics, 1995. ISBN 0940600323.

P. Loh and P. Bühlmann. High-dimensional learning of linear causal networks via inverse covariance estimation. *J. Mach. Learn. Res.*, 15(1):3065–3105, 2014. doi: 10.5555/2627435.2697063. URL https://dl.acm.org/doi/10.5555/2627435.2697063.

C. L. Lucchesi and D. H. Younger. A Minimax Theorem for Directed Graphs. *Journal of the London Mathematical Society*, s2-17(3):369–374, 1978. doi: https://doi.org/10.1112/jlms/s2-17.3.369. URL https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/jlms/s2-17.3.369.

H. Manzour, S. Küçükyavuz, H.-H. Wu, and A. Shojaie. Integer Programming for Learning Directed Acyclic Graphs from Continuous Data. *INFORMS Journal on Optimization*, 3(1):46–73, 2021. doi: 10.1287/ijoo.2019.0040. URL https://doi.org/10.1287/ijoo.2019.0040.

D. Margaritis. *Learning Bayesian Network Model Structure from Data*. PhD thesis, Carnegie-Mellon University, 2003.

R. J. McEliece, D. J. C. MacKay, and J. Cheng. Turbo Decoding as an Instance of Pearl's "Belief Propagation" Algorithm. *IEEE J. Sel. Areas Commun.*, 16(2):140–152, 1998. doi: 10.1109/49.661103. URL https://doi.org/10.1109/49.661103.

C. Meek. *Graphical Models: Selecting causal and statistical models*. PhD thesis, Carnegie-Mellon University, 1997.

T. M. Mitchell. *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill, 1997. ISBN 978-0-07-042807-2. URL https://www.worldcat.org/oclc/61321007.

MOSEK ApS. *The MOSEK Optimizer API for Python. Version 10.0.*, 2022. URL https://docs.mosek.com/10.0/pythonapi/index.html.

R. E. Neapolitan. *Probabilistic reasoning in expert systems - theory and algorithms*. Wiley, 1990. ISBN 978-0-471-61840-9.

Y. Nesterov. A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. *Doklady Akademii Nauk SSSR*, 269:543–547, 1983. URL https://zbmath.org/?q=an:0535.90071.

Y. E. Nesterov. *Introductory Lectures on Convex Optimization - A Basic Course*, volume 87 of *Applied Optimization*. Springer, 2004. ISBN 978-1-4613-4691-3. doi: 10.1007/978-1-4419-8853-9. URL https://doi.org/10.1007/978-1-4419-8853-9.

I. Ng, A. Ghassami, and K. Zhang. On the Role of Sparsity and DAG Constraints for Learning Linear DAGs. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/d04d42cdf14579cd294e5079e0745411-Abstract.html.

P. D. Nhat, H. M. Le, and H. A. L. Thi. Accelerated Difference of Convex functions Algorithm and its Application to Sparse Binary Logistic Regression. In J. Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 1369–1375. ijcai.org, 2018. doi: 10.24963/ijcai.2018/190. URL https://doi.org/10.24963/ijcai.2018/190.

B. O'Donoghue and E. J. Candès. Adaptive Restart for Accelerated Gradient Schemes. *Found. Comput. Math.*, 15(3):715–732, 2015. doi: 10.1007/s10208-013-9150-3. URL https://doi.org/10.1007/s10208-013-9150-3.

S. Ott, S. Imoto, and S. Miyano. Finding Optimal Models for Small Gene Networks. In R. B. Altman, A. K. Dunker, L. Hunter, T. A. Jung, and T. E. Klein, editors, *Biocomputing 2004, Proceedings of the Pacific Symposium, Hawaii, USA, 6-10 January 2004*, pages 557–567. World Scientific, 2004. URL http://psb.stanford.edu/psb-online/proceedings/psb04/ott.pdf.

C. H. Papadimitriou and M. Yannakakis. Optimization, Approximation, and Complexity Classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991. doi: 10.1016/0022-0000(91)90023-X. URL https://doi.org/10.1016/0022-0000(91)90023-X.

Y. W. Park and D. Klabjan. Bayesian Network Learning via Topological Order. *J. Mach. Learn. Res.*, 18:99:1–99:32, 2017. URL http://jmlr.org/papers/v18/17-033.html.

P. Parviainen and M. Koivisto. Bayesian structure discovery in Bayesian networks with less space. In Y. W. Teh and D. M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 589–596. JMLR.org, 2010. URL http://proceedings.mlr.press/v9/parviainen10a.html.

J. Pearl. Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach. In D. L. Waltz, editor, *Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, PA, USA, August 18-20, 1982*, pages 133–136. AAAI Press, 1982. URL http://www.aaai.org/Library/AAAI/1982/aaai82-032.php.

J. Pearl. *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, 1989.

K. Pearson. 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. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900. doi: 10.1080/14786440009463897. URL https://doi.org/10.1080/14786440009463897.

J. Pellet and A. Elisseeff. Using Markov Blankets for Causal Structure Learning. *J. Mach. Learn. Res.*, 9:1295–1342, 2008. doi: 10.5555/1390681.1442776. URL https://dl.acm.org/doi/10.5555/1390681.1442776.

J. Peters and P. Bühlmann. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika*, 101(1):219–228, 2013. ISSN 0006-3444. doi: 10.1093/biomet/ast043. URL https://doi.org/10.1093/biomet/ast043.

J. Peters, J. M. Mooij, D. Janzing, and B. Schölkopf. Identifiability of Causal Graphs using Functional Models. In F. G. Cozman and A. Pfeffer, editors, *UAI 2011, Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, July 14-17, 2011*, pages 589–598. AUAI Press, 2011. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2216&proceeding_id=27.

V. Ramachandran. Finding a Minimum Feedback Arc Set in Reducible Flow Graphs. *J. Algorithms*, 9(3):299–313, 1988. doi: 10.1016/0196-6774(88)90022-3. URL https://doi.org/10.1016/0196-6774(88)90022-3.

V. Raman and S. Saurabh. Improved fixed parameter tractable algorithms for two "edge" problems: MAXCUT and MAXDAG. *Inf. Process. Lett.*, 104(2):65–72, 2007. doi: 10.1016/j.ipl.2007.05.014. URL https://doi.org/10.1016/j.ipl.2007.05.014.

I. Razgon. Computing Minimum Directed Feedback Vertex Set in $\mathcal{O}^*(1.9977^n)$. In G. F. Italiano, E. Moggi, and L. Laura, editors, *Theoretical Computer Science, 10th Italian Conference, ICTCS 2007, Rome, Italy, October 3-5, 2007, Proceedings*, pages 70–81. World Scientific, 2007.

R. T. Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathematics and Physics. Princeton University Press, 1970. ISBN 978-1-4008-7317-3.

T. Roos, T. Silander, P. Kontkanen, and P. Myllymaki. Bayesian network structure learning using factorized NML universal models. In *2008 Information Theory and Applications Workshop*, pages 272–276, 2008. doi: 10.1109/ITA.2008.4601061.

M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian Approach to Filtering Junk E-Mail. *AAAI'98 Workshop on Learning for Text Categorization*, 62, 1998.

M. Scanagatta, C. P. de Campos, G. Corani, and M. Zaffalon. Learning Bayesian Networks with Thousands of Variables. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1864–1872, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/2b38c2df6a49b97f706ec9148ce48d86-Abstract.html.

G. Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461 – 464, 1978. doi: 10.1214/aos/1176344136. URL https://doi.org/10.1214/aos/1176344136.

M. Scutari. Learning Bayesian Networks with the bnlearn R Package. *Journal of Statistical Software*, 35(3):1–22, 2010. doi: 10.18637/jss.v035.i03. URL https://www.jstatsoft.org/index.php/jss/article/view/v035i03.

M. Scutari. An Empirical-Bayes Score for Discrete Bayesian Networks. In A. Antonucci, G. Corani, and C. P. de Campos, editors, *Probabilistic Graphical Models - Eighth International Conference, PGM 2016, Lugano, Switzerland, September 6-9, 2016. Proceedings*, volume 52 of *JMLR Workshop and Conference Proceedings*, pages 438–448. JMLR.org, 2016. URL http://proceedings.mlr.press/v52/scutari16.html.

M. Scutari. Bayesian Network Constraint-Based Structure Learning Algorithms: Parallel and Optimized Implementations in the bnlearn R Package. *Journal of Statistical Software*, 77(2):1–20, 2017. doi: 10.18637/jss.v077.i02. URL https://www.jstatsoft.org/index.php/jss/article/view/v077i02.

P. D. Seymour. Packing Directed Circuits Fractionally. *Comb.*, 15(2):281–288, 1995. doi: 10.1007/BF01200760. URL https://doi.org/10.1007/BF01200760.

S. Shimizu, P. O. Hoyer, A. Hyvärinen, and A. J. Kerminen. A Linear Non-Gaussian Acyclic Model for Causal Discovery. *J. Mach. Learn. Res.*, 7:2003–2030, 2006. URL http://jmlr.org/papers/v7/shimizu06a.html.

T. Silander and P. Myllymäki. A Simple Approach for Finding the Globally Optimal Bayesian Network Structure. In *UAI '06, Proceedings of the 22nd Conference in Uncertainty in Artificial Intelligence, Cambridge, MA, USA, July 13-16, 2006*. AUAI Press, 2006. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=1256&proceeding_id=22.

M. Simpson, V. Srinivasan, and A. Thomo. Efficient Computation of Feedback Arc Set at Web-Scale. *Proc. VLDB Endow.*, 10(3):133–144, 2016. doi: 10.14778/3021924.3021930. URL http://www.vldb.org/pvldb/vol10/p133-simpson.pdf.

A. P. Singh and A. W. Moore. Finding optimal Bayesian networks by dynamic programming. , 2004. doi: 10.1184/R1/6605669.v1. URL https://kilthub.cmu.edu/articles/journal_contribution/Finding_optimal_Bayesian_networks_by_dynamic_programming/6605669.

P. Spirtes, C. Meek, and T. S. Richardson. Causal Inference in the Presence of Latent Variables and Selection Bias. In P. Besnard and S. Hanks, editors, *UAI '95: Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec, Canada, August 18-20, 1995*, pages 499–506. Morgan Kaufmann, 1995. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=469&proceeding_id=11.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search, Second Edition*. Adaptive computation and machine learning. MIT Press, 2000. ISBN 978-0-262-19440-2.

C. Su, A. S. Andrew, M. R. Karagas, and M. E. Borsuk. Using Bayesian networks to discover relations between genes, environment, and disease. *BioData Min.*, 6:6, 2013. doi: 10.1186/1756-0381-6-6. URL https://doi.org/10.1186/1756-0381-6-6.

R. E. Tarjan. Enumeration of the Elementary Circuits of a Directed Graph. *SIAM J. Comput.*, 2(3):211–216, 1973. doi: 10.1137/0202017. URL https://doi.org/10.1137/0202017.

R. Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. doi: https://doi.org/10.1111/j.2517-6161.1996.tb02080.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1996.tb02080.x.

I. Tsamardinos and C. F. Aliferis. Towards Principled Feature Selection: Relevancy, Filters and Wrappers. In C. M. Bishop and B. J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, AISTATS 2003, Key West, Florida, USA, January 3-6, 2003*. Society for Artificial Intelligence and Statistics, 2003. URL http://research.microsoft.com/en-us/um/cambridge/events/aistats2003/proceedings/133.pdf.

I. Tsamardinos, C. F. Aliferis, and A. R. Statnikov. Time and sample efficient discovery of Markov blankets and direct causal relations. In L. Getoor, T. E. Senator, P. M. Domingos, and C. Faloutsos, editors, *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 673–678. ACM, 2003. doi: 10.1145/956750.956838. URL https://doi.org/10.1145/956750.956838.

L. G. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM J. Comput.*, 8(3):410–421, 1979. doi: 10.1137/0208032. URL https://doi.org/10.1137/0208032.

Y. Yu, T. Gao, N. Yin, and Q. Ji. DAGs with No Curl: An Efficient DAG Structure Learning Approach. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 12156–12166. PMLR, 2021. URL http://proceedings.mlr.press/v139/yu21a.html.

H. Zhang. The Optimality of Naive Bayes. In V. Barr and Z. Markov, editors, *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA*, pages 562–567. AAAI Press, 2004. URL http://www.aaai.org/Library/FLAIRS/2004/flairs04-097.php.

X. Zheng, B. Aragam, P. Ravikumar, and E. P. Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*,

pages 9492–9503, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/e347c51419ffb23ca3fd5050202f9c3d-Abstract.html.

X. Zheng, C. Dan, B. Aragam, P. Ravikumar, and E. P. Xing. Learning Sparse Nonparametric DAGs. In S. Chiappa and R. Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 3414–3425. PMLR, 2020. URL http://proceedings.mlr.press/v108/zheng20a.html.

R. Zhu, A. Pfadler, Z. Wu, Y. Han, X. Yang, F. Ye, Z. Qian, J. Zhou, and B. Cui. Efficient and Scalable Structure Learning for Bayesian Networks: Algorithms and Applications. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*, pages 2613–2624. IEEE, 2021. doi: 10.1109/ICDE51399.2021.00292. URL https://doi.org/10.1109/ICDE51399.2021.00292.

Article I

Scalable Bayesian Network Structure Learning via Maximum Acyclic Subgraph

Pierre Gillot & Pekka Parviainen

Proceedings of the 10th International Conference on Probabilistic Graphical Models, PMLR 138:209-220, 2020

Scalable Bayesian Network Structure Learning via Maximum Acyclic Subgraph

Pierre Gillot PIERRE.GILLOT@UIB.NO
Department of Informatics, University of Bergen, Norway

Pekka Parviainen PEKKA.PARVIAINEN@UIB.NO
Department of Informatics, University of Bergen, Norway

Abstract

Learning the structure of a Bayesian network is an NP-hard problem and exact learning algorithms that are guaranteed to find an optimal structure are not feasible with large number of variables. Thus, large-scale learning is usually done using heuristics that do not provide any quality guarantees. We present a heuristic method that scales up to networks with hundreds of variables and provides quality guarantees in terms of an upper bound for the score of the optimal network. The proposed method consists of two parts. First, we simplify the problem by approximating local scores using so-called edge scores. With the edge scores learning an optimal Bayesian network structure is equivalent to finding the maximum acyclic subgraph. Second, we solve the maximum acyclic subgraph problem fast using integer linear programming. Additionally, we choose the approximation in a specific way so that an upper bound for the score of an optimal network can be obtained.

Keywords: Bayesian networks; Structure learning; Integer Linear Programming.

1. Introduction

Bayesian networks are a type of probabilistic graphical model widely used in machine learning. They were originally introduced by Pearl (1988). Bayesian networks consist of two parts: a structure and parameters. The structure, represented by a directed acyclic graph (DAG), expresses conditional independencies between variables and the parameters specify local conditional distributions.

Often, the Bayesian network is not given but we learn it from data. The process of learning a Bayesian network is twofold: first, we learn the structure and then we learn the parameters of the local distributions. In this paper, we will study learning the structure of a Bayesian network. Specifically, we solve the Bayesian network structure learning (BNSL) problem using the so-called *score-based* approach where each structure is assigned a score based on how well it fits to the data and the goal is to find a structure that maximizes the score.

Exact structure learning algorithms are guaranteed to find an optimal structure, that is, a structure that maximizes the score. Exact structure learning is NP-hard (Chickering, 1996) so it is unlikely that exact algorithms scale-up to large networks in worst case scenarios. Even though sometimes even networks with few dozens of variables are too large for state-of-the-art exact algorithms (Cussens, 2011), they can solve easy instances with up to few hundred variables. When the exact algorithms cannot handle a dataset, one typically resorts to various scalable heuristics (Scanagatta et al., 2015; Tsamardinos et al., 2006). A downside of using heuristics is that they do not provide any guarantees for the quality of the result.

We present a new structure learning algorithm that attempts to bridge the gap between these two extremes. Specifically, we speed up the algorithm by relaxing the requirement of guaranteeing the optimal DAG. At the same time, we are still able to give an estimate of the quality of the found solution.

Our method builds upon GOBNILP (Cussens, 2011), the state-of-the-art exact algorithm, which is based on integer linear programming (ILP). In score-based structure learning, it is common to use so-called decomposable scores which means that the score of a structure is a sum of local scores for node-parent set

pairs. Many algorithms, including GOBNILP, require that these local scores are computed as a preprocessing step and given as an input to an optimization algorithm. One of the challenges using GOBNILP is that a large number of local scores makes solving the integer linear program slower. We tackle this problem by approximating local scores with approximate scores for each potential edge and thereby restricting the size of the input to a quadratic number of scores. Given edge scores, finding the optimal structure reduces to the maximum acyclic subgraph (MAS) problem; hence, we call our method *BNSL2MAS*. Also the MAS problem is NP-hard (Karp, 1972) but in practice it is much faster to solve.

A key challenge in our approach is the quality of the approximation. To enable us to assess the quality of the found network, we learn the approximate scores under the constraint that the approximate score (sum of the edge scores) for each node-parent set pair upper bounds the local score. This guarantees that the approximate score of the optimal solution of the MAS problem gives an upper bound for the score of the optimal solution of the BNSL problem.

The above-described version of BNSL2MAS is fast but it has a considerable weakness: It returns networks that are dense and upper bounds are too high to have any practical value. To get tighter upper bounds, we can add various additional constraints (described in Section 3.3). However, tightening the upper bound does not come for free. It leads to a tradeoff between speed and tightness of the bound. Our empirical results are mixed: We observed that the variants of BNSL2MAS that were fast did not give usable upper bounds. On the other hand, the variants that gave non-trivial upper bounds were slow and sometimes returned graphs which were of poorer quality.

Related work. There is limited amount of "approximate" algorithms for Bayesian network structure learning. Most notably, while integer linear programming (ILP) -based algorithms (Jaakkola et al., 2010; Cussens, 2011) are usually used as exact algorithms, they can be used as anytime algorithms. That is, it is possible to interrupt the algorithm at any time and output the best DAG found so far. Furthermore, ILP solves relaxations and optimal solutions for the relaxations give upper bounds for the score of the optimal DAG.

Other works include an approximation algorithm developed by Ziegler (2008). The algorithm gives k-approximation where k is the maximum size of the parent set. Also greedy equivalence search (GES) (Chickering, 2002) can be seen as a heuristic with quality guarantees. However, the guarantees hold only asymptotically.

Recently, there has also been work on developing anytime algorithms for Bayesian network structure learning (see, e.g., Lee and van Beek (2017); Scanagatta et al. (2017)). These algorithms typically consist of sampling node orders and then finding a best DAG that is compatible with the order.

2. Preliminaries

2.1 Score-based Structure Learning

Let $G = (N, A)$ be a DAG where N is the node set and A is the arc set. We denote the parent set of node v in G by A_v. Let n denote the cardinality of N.

The score of a DAG measures how well the DAG fits to the data. A score is decomposable, if the score of a DAG G can be written as

$$S(G) = \sum_{v \in N} S_v(A_v),$$

where $S_v(A_v)$ is the local score of node v given the parent set A_v. Commonly used scores such as BDe, BDeu, and BIC are decomposable. Our approach works for any decomposable score.

The Bayesian network structure learning (BNSL) problem with a decomposable score can be defined as

$$\arg\max_G \sum_v S_v(A_v)$$

s.t. A_v representing a DAG.

Note that A_v refers to the parent set of v in the graph G.

Without any restrictions, the input of the BNSL problem would be 2^{n-1} local scores for every v. Generally, computing such a number of local scores is not feasible. Thus, one typically restricts the number of local scores. To this end, let $\mathcal{F}_v$ denote the set of potential parent sets of the node v. We treat $S_v(A_v) = -\infty$ for all $A_v \notin \mathcal{F}_v$. One common way to restrict the size of $\mathcal{F}_v$ is pruning, that is, removing or not computing local scores for parent sets that cannot be a part of an optimal DAG. A commonly used pruning rule is based on the observation that if $S_v(W) > S_v(W')$ and $W \subset W'$ then W' cannot be the parent set v in the optimal graph. Another way to restrict the number of local scores is to assume that the maximal size of the parent sets is bounded by a small integer k.

We note that for large data sets, the number of local scores to compute becomes quickly restrictive even for a small k. For example, with 500 nodes and $k = 3$ one would need to compute approximately $500^4 \approx 6 \times 10^{10}$ local scores. Thus, heuristics that scale up to thousands of nodes, such as Scanagatta et al. (2015), compute local scores greedily online. We concentrate on slightly smaller networks and assume that the local scores are given (though we do assume that they have been pre-pruned).

2.2 Integer Linear Programming Formulation

The state of the art in exact score-based structure learning in Bayesian networks is based on integer linear programming (ILP) (Jaakkola et al., 2010; Cussens, 2011). In an integer linear programming formulation, one introduces a set of binary variables. The current state of the art involves the so-called "family variables"; a family variable $I_v(W)$ takes value 1 if W is the parent set of v in the DAG, and 0 otherwise. Now the optimization problem can be written as

$$\arg\max_{I_v(W)} \sum_{v,W} S_v(W) I_v(W)$$

s.t. the variables $I_v(W)$ represent a DAG.

We note that the constraints guaranteeing that the family variables $I_v(W)$ represent a DAG can be formulated in several different ways. GOBNILP uses so-called cluster constraints to guarantee acyclicity[1].

We note that the ILP formulation has one variable for each potential parent set in $\mathcal{F}_v$. The number of ILP variables can affect the optimization speed drastically and be the performance bottleneck in practice when the number of potential parent sets is large. This is a serious challenge because with a large number of nodes we tend to have lots of potential parent sets even when indegrees are small. Next, we will present edge scores that are designed to alleviate this problem.

3. Proposed Method

In this section, we will introduce the BNSL2MAS method for the BNSL problem. As mentioned in Section 2.1, we assume that we are given local scores $S_v(A_v)$ for $v \in N$ and $A_v \in \mathcal{F}_v$ and our goal is to find a DAG G that maximizes the sum of local scores. We do not directly solve the BNSL problem but we first approximate the local scores using edge scores and thereby convert the problem to the maximum acyclic subgraph problem which is easier to solve in practice.

1. These constraints are based on the observation that in a DAG, every subset of nodes has at least one node that does not have any parents in that subset.

3.1 Edge Scores

The core idea of our work is to approximate local scores $S_v(W)$ using a sum of edge scores $e_v(w)$. The rationale behind this approximation is that the resulting ILP formulation has only a quadratic number of variables instead of potentially an exponential number of variables. Therefore, it is expected that finding the DAG that maximizes the approximate score is faster than finding the DAG that maximizes the original score.

Formally, we define an approximate local score for a node v and a parent set W by $\widetilde{S}_v(W) = b_v + \sum_{w \in W} e_v(w)$ where b_v is the bias of v and $e_v(w)$ are the edge scores. This approximate score is additive, that is, each approximate local score is a sum of individual edge score contributions from the parent nodes w with respect to the node v, plus a bias that represents the score estimation of v having no parent at all.

Given local scores $S_v(W)$, we want to estimate the values taken by the variables b_v and $e_v(w)$. To this end, we define a loss function $L\big(S_v(W), \widetilde{S}_v(W)\big)$. In our experiments, we have used the absolute loss: $L\big(S_v(W), \widetilde{S}_v(W)\big) = |S_v(W) - \widetilde{S}_v(W)|$ and the squared loss: $L\big(S_v(W), \widetilde{S}_v(W)\big) = (S_v(W) - \widetilde{S}_v(W))^2$. However, other loss functions are also possible. Furthermore, in order to get approximation guarantees, we constrain the approximate local scores to always upper bound the true local scores.

Now, for each node v its bias and edge scores can be estimated solving

$$\operatorname*{argmin}_{b_v, e_v(w)} \sum_{W \in \mathcal{F}_v} L\big(S_v(W), \widetilde{S}_v(W)\big)$$

$$\text{s.t. } S_v(W) \leq \widetilde{S}_v(W) \quad \forall W \in \mathcal{F}_v \tag{1}$$

where $\widetilde{S}_v(W) = b_v + \sum_{w \in W} e_v(w)$.

The approximation guarantees hold for any loss function. The tightness of the bounds and the time requirements of solving the optimization problem, however, may be affected by the choice of the loss function.

The quadratic loss function results in a quadratic optimization problem which can be solved efficiently with solvers like GUROBI. In practice, even for large and dense datasets, the estimation of approximate scores is very fast compared to solving the structure learning problem itself. This is mainly due to the bias and edge scores being continuous variables, as well as the models for each child node v being independent meaning we can easily parallelize them.

There are at most $n(n-1)$ edge scores. We note that the above formulation yields a non-zero edge score $e_v(w)$ only when w is a member of at least one potential parent set, that is, $e_v(w)$ is non-zero only if $w \in W$ for some $W \in \mathcal{F}_v$. This can further simplify the following structure learning problem, especially when the underlying DAG is sparse.

3.2 Maximum Acyclic subgraph

Given the edge scores, solving BNSL reduces to solving an instance of the maximum acyclic subgraph (MAS) problem which is still NP-hard, as mentioned before. In practice, however, MAS can be solved significantly faster than BNSL.

Formally, the *maximum acyclic subgraph (MAS)* problem is defined as follows. We are given a directed graph $G' = (N, A')$ with a weight $s(a)$ assigned for each arc $a \in A'$. The goal is to find a directed acyclic graph $G = (N, A)$ such that $A \subseteq A'$ with maximum total weight $\sum_{a \in A} s(a)$. We note that solving MAS is equivalent to solving the feedback arc set (FAS) problem where one is given a directed weighted graph and the goal is to remove the lightest set of arcs to make the graph acyclic[2].

We solve MAS using integer linear programming. Our ILP formulation of MAS is as follows: consider the weighted graph whose edges are weighted by the edge scores $e_v(w)$. Now, for each directed edge (w, v)

2. MAS and FAS are complementary: the feedback arc set equals to $A' \setminus A$.

define a binary variable $J_v(w)$ which takes value 1 if w is a parent of v in the DAG, 0 otherwise. Then the optimization problem can be written as

$$\underset{J_v(w)}{\operatorname{argmax}} \sum_{v,w} e_v(w) J_v(w)$$

$$\text{s.t. the } J_v(w) \text{ representing a DAG.}$$

There are several alternative ways to guarantee that the resulting graph is a DAG. We solve MAS using the lazy set cover ILP formulation as in Baharev et al. (2015)[3].

The constraints to guarantee acyclicity can be formulated as follows. Let c be a directed cycle with length $|\mathsf{c}|$. Let $\mathcal{C}$ be the set of cycles in G'. If G is a DAG then at least one arc of each cycle has to be absent from G. Therefore, to ensure that G is a DAG, we can add the following constraints: for each cycle $\mathsf{c} \in \mathcal{C}$ it holds that

$$\sum_{(w,v)\in \mathsf{c}} J_v(w) \leq |\mathsf{c}| - 1.$$

One challenge with this formulation is that the ILP contains one constraint for each cycle in G'. In large or dense graphs there are potentially lots of cycles and it is usually not feasible to explicitly include all of them. In this case, we use the same approach as Cussens (2011) and add constraints lazily as cutting planes. That is, we start with a relaxation and whenever we find a feasible solution for the relaxation, it is checked whether the solution is acyclic. If not, we use the cutting plane approach and add at least one cycle constraint that make the found solution infeasible. In this approach, finding good cutting planes is essential for solving the problem quickly. We use a simple heuristical approach: for each edge in the found graph, we find the shortest cycle that goes through the edge (if the edge is a part of a cycle).

In Section 3.1, we constrained the approximate local scores to upper bound the local scores. Theorem 1 implies that the approximate score of the DAG found by solving the MAS problem upper bounds the (true) score of the DAG found by solving the BNSL problem.

Theorem 1 *Let G_{opt} be the optimal DAG of the BNSL problem and $\widetilde{G}_{opt}$ be the optimal DAG of the MAS problem. Furthermore, let $S_v(W) \leq \widetilde{S}_v(W) \quad \forall v \in N \quad \forall W \in \mathcal{F}_v$. Then,*

$$S(G_{opt}) \leq C + \widetilde{S}(\widetilde{G}_{opt}),$$

where $C = \sum_{v\in N} b_v$ is the sum of bias terms.

Proof For any graph $G = (N, A)$, we have that

$$\begin{aligned} S(G) &= \sum_{v\in N} S_v(A_v) \\ &\leq \sum_{v\in N} \widetilde{S}_v(A_v) \\ &= \sum_{v\in N} \left(b_v + \textstyle\sum_{w\in A_v} e_v(w) \right) \\ &= C + \sum_{v\in N} \textstyle\sum_{w\in A_v} e_v(w) \\ &= C + \widetilde{S}(G). \end{aligned}$$

3. Baharev et al. (2015) give their formulation for FAS problem but we adapt it for the MAS problem.

Because $\widetilde{G}_{opt}$ maximizes the approximate score $\widetilde{S}$, it follows that $\widetilde{S}(G_{opt}) \leq \widetilde{S}(\widetilde{G}_{opt})$. Finally, because the approximate score upper bounds the local score, we get the upper bound guarantee

$$S(G_{opt}) \leq C + \widetilde{S}(\widetilde{G}_{opt}).$$

■

3.3 Tightening the Upper Bound

One challenge with the MAS formulation above is that the obtained networks tend to be dense. This happens because the edge scores do not penalize complexity and thus MAS keeps adding edges with positive weights as long as they do not induce cycles.

This denseness has two negative consequences. First, the found network $\widetilde{G}_{opt}$, which is optimal with respect to approximate scores $\widetilde{S}$, may not have a high score with respect to the true scores S. Second, the upper bound depends on the weights of the included edges, adding an extra edge makes the bound looser. As a result, the upper bounds provided by the naive formulation above are usually not very useful.

Fortunately, we can alleviate these problems. First, we note that if the parent set of v in $\widetilde{G}_{opt}$ is not among the candidate parent sets, that is, $A_v \notin \mathcal{F}_v$, we can improve the score by replacing A_v by its highest-scoring subset with respect to true local scores, that is, $\arg\max_{A'_v \subset A_v} S_v(A'_v)$. In other words, given $\widetilde{G}_{opt} = (N, A)$ we can construct a graph $\widetilde{G}'_{opt} = (N, A')$ such that $\widetilde{G}'_{opt}$ maximizes the score $S(\widetilde{G}'_{opt})$ subject to $A' \subseteq A$. Because removing edges cannot make an acyclic graph cyclic, we can select the parent set of each node independently and therefore the running time is proportional to the size of $\mathcal{F}_v$. We refer to this variant of the BNSL2MAS method as *BNSL2MAS (Base)*.

Second, we can tighten the upper bound by adding additional constraints. A simple way is to bound the maximum indegree of $\widetilde{G}_{opt}$. If the size of the largest parent set in $\mathcal{F}_v$ is k then we can add constraints

$$\sum_{w \in N \setminus \{v\}} J_v(w) \leq k$$

for all v. A weakness of this approach is that it is still possible that the parent set of v in $\widetilde{G}_{opt}$ is not among the candidate parent sets: $A_v \notin \mathcal{F}_v$. We refer to this variant as *BNSL2MAS (Bounded parent set)*.

A more sophisticated approach is to restrict the solutions of the MAS problem to be DAGs whose parent sets are among candidate parents sets. That is, A_v can be the parent set of v in $\widetilde{G}_{opt}$ only if $A_v \in \mathcal{F}_v$. To achieve this, we can add an additional binary variable $K_v(W)$ for all $W \in \mathcal{F}_v$. The variable $K_v(W)$ equals to 1 if and only if W is the parent set of v. This can be achieved by adding the following constraints:

$$\begin{aligned} \sum_{w \in W} J_v(w) &\geq |W| K_v(W), \\ \sum_{w \notin W} J_v(w) &\leq n - n K_v(W). \end{aligned}$$

Then, we add a constraint that each node has exactly one parent set, that is, $\sum_{W \in \mathcal{F}_v} K_v(W) = 1$. We refer to the variant of BNSL2MAS that includes all the candidate parent set constraints as *BNSL2MAS (Complete candidate parent)*.

Another way to restrict the parent sets to candidate parent sets is to add the following constraints: $\forall v \in N$ and $\forall W \notin F_v$ it is required

$$\sum_{w \in W} J_v(w) \leq \sum_{w' \in \hat{W} \setminus W} J_v(w') + |W| - 1,$$

Name	n	Local scores	Name	n	Local scores
autos	26	25,238	letter	17	18,841
carpo_100	60	5,068	mushroom	23	13,025
carpo_10000	60	16,391	Pigs_10000	441	304,219
Diabetes_1000	413	21,493	wdbc_Bde	31	13,473
Diabetes_10000	413	262,129	wdbc_BIC	31	14,613

Table 1: Data sets with the number of nodes n and the number of local scores

where

$$\hat{W} = \bigcup_{W' \in F_v : W' \supset W} W'.$$

Intuitively, this means that if the parent set of v contains a set $W \notin \mathcal{F}_v$ then the parent set must include at least one additional node. The additional node must be a member of some candidate parent set that is a superset of W. We note $\hat{W} = N \setminus W$ would be also a valid constraint but the above mentioned constraint is tighter. The previous formulation is not completely unproblematic: We need to have one constraint for every $W \notin \mathcal{F}_v$ which means that usually we will have exponential number of constraints. Thus, adding all of them is typically not possible. Fortunately, we can again add the constraints as cutting planes during the optimization process. Whenever we find a graph G with $A_v \notin \mathcal{F}_v$, we can add the corresponding constraint. We refer to the variant that adds candidate parent set constraints lazily as cutting planes as *BNSL2MAS (Lazy candidate parent)*.

It should be noted that removing the constraint $S_v(W) \leq \widetilde{S}_v(W)$ in Equation 1 that forces the approximate score to be an upper bound for the corresponding true score would make the loss smaller and thereby the approximation would be tighter. However, the constraint is essential for the quality guarantees and after removing it we could not guarantee that the solution of the MAS problem would upper bound the score of the optimal solution of the BNSL problem.

4. Experiments

4.1 Test Setup

4.1.1 Implementation

The method was implemented using Python. We used Gurobi (version 9.0) to solve linear, quadratic, and integer linear programs.

4.1.2 Data sets

We use a subset of the data sets listed in the GOBNILP home page[4]. As we are interested in scalability of our method, we restrict our analysis to the most challenging data sets. To this end, we selected data sets for which GOBNILP used at least 100 seconds (or GOBNILP was not able to find the optimal network at all). The selected data sets can be seen in Table 1.

We used the same scoring criteria that were used in the original data sets, that is, either BDe or BIC. For wdbc, we have two versions, one for each scoring criterion.

4. https://www.cs.york.ac.uk/aig/sw/gobnilp/

4.1.3 EXPERIMENTS

In Section 3, we proposed several different variants. The goal of this experiment is to investigate which of the variants of the proposed method work well. We consider the following variants.

For computing edge scores, we compare two loss functions: absolute loss and squared loss.

For different variants of BNSL2MAS, we consider BNSL2MAS (Base), BNSL2MAS (Bounded parent sets), BNSL2MAS (Complete candidate parents), BNSL2MAS (Lazy candidate parents). As mentioned in Section 3.2, we can solve one of the two equivalent optimization problems: maximum acyclic subgraph (MAS) or feedback arc set (FAS). Both formulations give the same result (if we run them long enough to the optimality) but the running time can be different. Therefore, if we run the algorithms for a fixed time and at least one of them does not finish then we can also get different graphs. Thus, we compare both MAS and FAS formulations.

Performance measures. In all of the experiments, we measured the performance of the methods by scores of found DAGs and running time. For ILP-based methods, we also recorded upper bounds. For the proposed method, we computed both the approximate score $\widetilde{S}$ and the "true" score S for each DAG.

All methods had a time limit of one hour of running time. We evaluated methods in anytime fashion: Whenever a method found a DAG that has higher score than any of the previously found DAGs, we recorded the score and time of the newly found DAG. Furthermore, all algorithms were given pre-computed local scores[5] as an input and computing the local scores did not count towards the running time except for hill climbing, for which raw datasets were used.

Benchmarks. We benchmark the proposed method against both an exact algorithm and two heuristic algorithms. As the exact benchmark, we use GOBNILP (Bartlett and Cussens, 2017) which can scale up to networks with few hundred variables if the network is sparse.

As another benchmark we use the greedy hill climbing. In greedy hill climbing, there are three operators that are used to improve the DAG: adding a new edge, removing an existing edge or flipping an edge. The greedy hill climbing starts with an empty DAG and greedily applies the operator that increases the score most. This is continued until none of the operators increases the score of the DAG. We benchmarked hill climbing from bnlearn (Scutari, 2010), an optimized implementation that uses score caching for maximized efficiency.

Our last benchmark is WINASOBS (Scanagatta et al., 2017), where an ordering-based search algorithm is coupled with an iterated local search meta-heuristic that uses a custom window insertion operator. WINASOBS currently yields state-of-the-art performance on large graphs with several thousands of nodes.

Computer. Tests were conducted using a computer with Intel(R) Core(TM) i5-7500 processor with four 3.40GHz cores. The computer had 32Gb RAM and its operating system was Ubuntu 18.04.4 LTS (64bit).

4.2 Results

4.2.1 COMPUTING EDGE SCORES

We computed edge scores with both squared and absolute loss. We observed that computing the edge scores is very fast. In the case of squared loss, for all but two of our data sets computing edge scores took less than 2 seconds in total. The two most time-consuming data sets were Diabetes_10000 and Pigs_10000 for which computing the edge scores took 35 and 72 seconds, respectively. Computing edge scores with absolute loss is even faster: computing edge scores for any data set took always less than 2 seconds.

4.2.2 BNSL2MAS

Next, we will review some of the experimental results. Due to space constraints, we show only part of the results. Let us start by analysing the BNSL2MAS (Base). Results from selected data sets are shown in

5. Loaded from `https://www.cs.york.ac.uk/aig/sw/gobnilp/data/`.

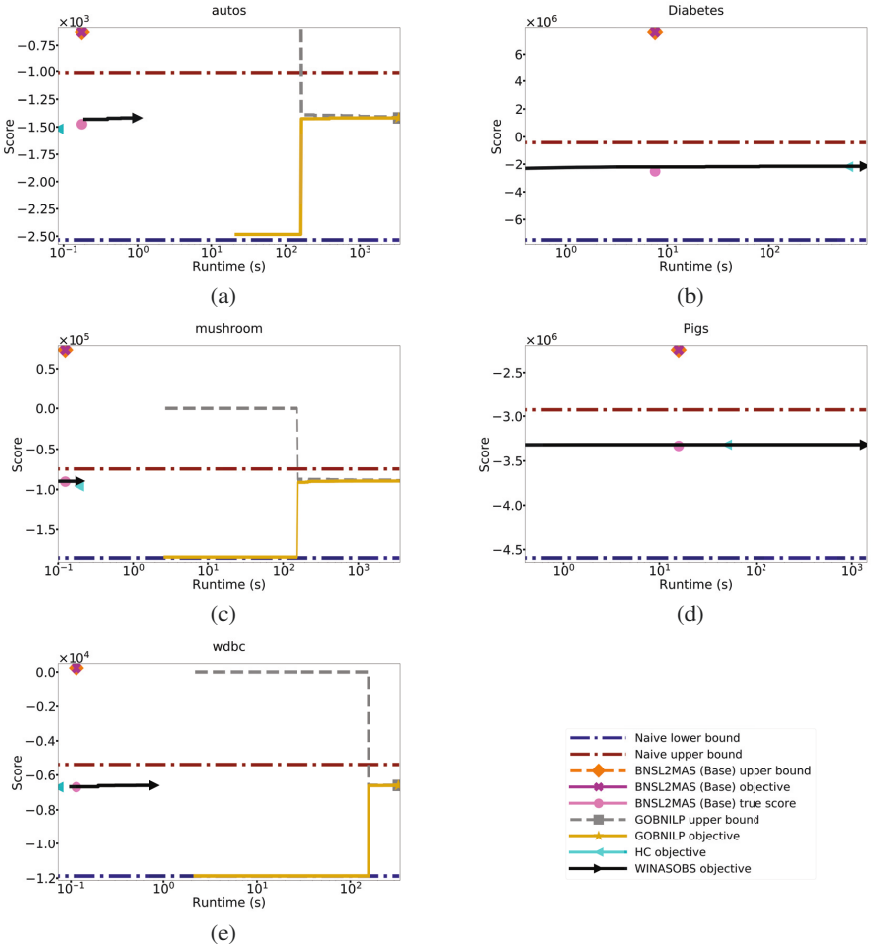

Figure 1: Scores and running times for BNSL2MAS (Base) and BNSL2MAS (Lazy candidate parents) with the benchmarks. Scores are the score of the best DAG found at a given data point. Edge scores were computed using squared loss. Marker at the end of a curve denotes that the method has finished (otherwise, the execution was stopped at the time limit). Data sets: (a) autos, (b) Diabetes_10000, (c) mushroom, (d) Pigs_10000, and (e) wdbc.

Figure 1. To help us to put the scores and bounds in perspective, we compare them to naive bounds. As a *naive lower bound* we use the score of the empty graph. To get a *naive upper bound* for the optimal DAG, we select the highest scoring parent set for each node and compute the sum. Clearly, there cannot be a DAG that has a higher score. Furthermore, we show the hill climbing algorithm, WINASOBS, and GOBNILP as

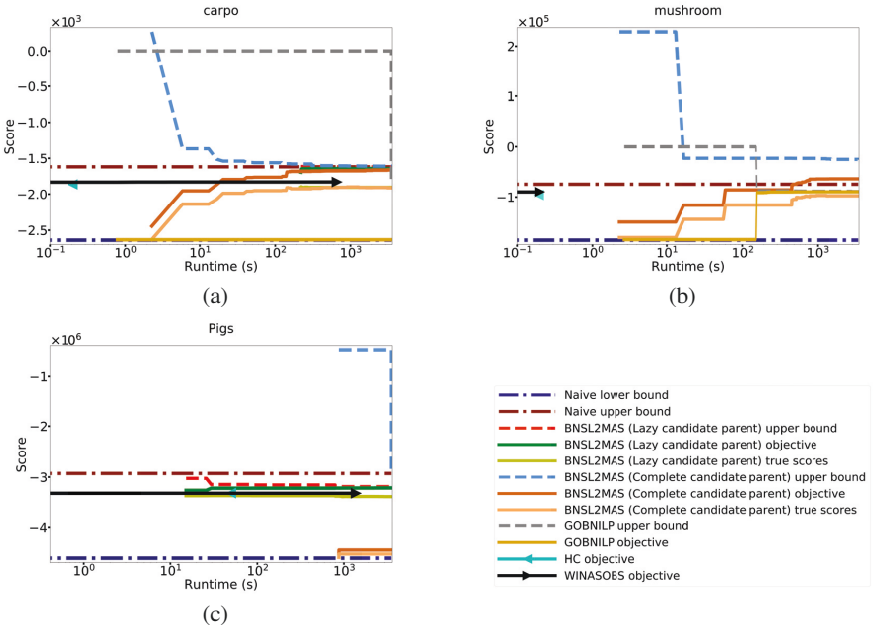

Figure 2: Score and running times for BNSL2MAS (Complete candidate parents) and BNSL2MAS (Lazy candidate parent) with benchmarks. Scores are the score of the best DAG found at a given data point. Edge scores were computed using squared loss. Marker at the end of a curve denotes that the method has finished (otherwise, the execution was stopped at the time limit). Datasets: (a) carpo_100, (b) mushroom, and (c) Pigs_10000.

benchmarks. In the plots, whenever a curve for some method is missing that means that the method did not find any feasible solutions before the time limit.

Our first observation is that BNSL2MAS (Base) is very fast. With the absolute loss, the slowest cases were the networks with over 400 nodes. Solving the MAS problem took less than 11 seconds for Diabetes_10000 and less than 25 seconds for Pigs_10000. With squared loss, the algorithm was even faster: running times for Diabetes_10000 and Pigs_10000 were less than 3 seconds and 13 seconds, respectively.

If we take a look at the DAGs found, we can see that, indeed, as mentioned in Section 3.3, solving BNSL2MAS without additional constraints for parent sets leads to dense networks. For example, the mushroom data set which consist of 23 nodes. The DAG found has 228 edges (average in-degree is 9.9) and maximum in-degree is 20. A graph with 23 nodes can have at most 253 edges. Thus, the found graph is almost complete. In comparison, the optimal graph found by GOBNILP has only 67 edges and maximum in-degree 4. As each edge that is added has a positive score, the upper bound is always much higher than the naive upper bound. Thus, in this case we cannot really claim that the base version has quality guarantees.

To evaluate the quality of the DAGs found by BNSL2MAS (Base), we selected the highest scoring candidate parent set for each node among the parents in the found graph and computed the true score of this DAG. The base version seems to perform roughly as well as the hill climbing heuristic: With squared

loss, the base version found a higher scoring network 4 times out of 10. WINASOBS found a higher-scoring DAG for all of the data sets. WINASOBS also finds good solutions very fast. We observe that the curves for WINASOBS in Figure 1 are almost horizontal indicating that there is no significant improvement after the initial phase.

To get tighter bounds, we can use either bounded parent set constraints or candidate parent set constraints. Selected results comparing BNSL2MAS (Complete candidate parent) and BNSL2MAS (Lazy candidate parent) are shown in Figure 2. The results are mixed. Still sometimes the upper bound is higher than the naive upper bound. Both of the versions are significantly slower than BNSL2MAS (Base). However, neither of them consistently overperforms the other. As a rule of thumb, BNSL2MAS (Complete candidate parent) seems to perform better on small data sets and significantly slow down with larger data sets; this is natural because it has one constraint for every existing node-parent set pair in the local scores. Furthermore, sometimes BNS2MAS (Complete candidate parent) and BNSL2MAS (Lazy candidate parent) are faster than GOBNILP and sometimes they are slower. We also note that in many cases BNSL2MAS (Base) finds better solutions than both BNSL2MAS (Complete candidate parent) and BNSL2MAS (Lazy candidate parent).

Due to space constraints, we do not show any plots about the performance of BNSL2MAS (Bounded parent sets). However, in general BNSL2MAS (Bounded parent sets) ends up somewhere in the middle of BNSL2MAS (Base) and the candidate parent set variants with respect to speed and loses to BNSL2MAS (Base) almost always with respect to the score. We also compared MAS and FAS version of BNSL2MAS (Plots not shown). There were no systematic differences, one was faster than the other for about half of the time. We also note that the differences between MAS and FAS were reasonably small.

5. Discussion

On the positive side, we observed that approximating local scores with edge scores and converting BNSL to MAS can significantly speed up learning Bayesian network structures. BNSL2MAS (Base) is very fast and could work with even larger data sets than what were considered in this paper. However, in practice the base version behaves like an order-finding heuristic and does not have quality guarantees.

Our empirical results are somewhat disappointing with respect to quality guarantees. The main result is that developing a scalable method with good quality guarantees seems to be a difficult task. There is a clear tradeoff: Adding seemingly simple constraints to tighten the upper bounds makes the optimization dramatically more difficult increasing the running time by several orders of magnitude even though the size of the search space decreases. We also observed that having quality guarantees can actually decrease the quality of the found solutions! The observation that additional constraints can significantly slow down ILP is not unique. For example, ILP-based methods of learning bounded tree-width Bayesian networks (Parviainen et al., 2014; Scanagatta et al., 2016) are orders of magnitude slower than GOBNILP even though bounded tree-width Bayesian networks are a small subset of all Bayesian networks. These results seem to suggest that the most practical version of BNSL2MAS is BNSL2MAS (Base).

We also note that the state-of-the-art anytime algorithm WINASOBS clearly outperformed BNSL2MAS. The main weakness of BNSL2MAS seems to be that the additive approximation is rather limiting. Thus, making BNSL2MAS really competitive would require finding a less restrictive approximate score without significantly increasing running time.

Acknowledgments

We thank James Cussens for fruitful conversations. The authors are affiliated with the CEDAS center in Bergen.

References

A. Baharev, H. Schichl, and A. Neumaier. An exact method for the minimum feedback arc set problem. 2015.

M. Bartlett and J. Cussens. Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence*, 244:258–271, 2017.

D. M. Chickering. *Learning Bayesian Networks is NP-Complete*, pages 121–130. Springer-Verlag, learning from data: artificial intelligence and statistics v edition, January 1996.

D. M. Chickering. Optimal Structure Identification With Greedy Search. *Journal of Machine Learning Reseach*, 3:507–554, 2002.

J. Cussens. Bayesian network learning with cutting planes. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, pages 153–160, 2011.

T. Jaakkola, D. Sontag, A. Globerson, and M. Meila. Learning Bayesian network structure using lp relaxations. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 358–365, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

C. Lee and P. van Beek. Metaheuristics for score-and-search Bayesian network structure learning. In *Proceedings of the 30th Canadian Conference on Artificial Intelligence*, 2017.

P. Parviainen, H. S. Farahani, and J. Lagergren. Learning Bounded Tree-width Bayesian Networks using Integer Linear Programming. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 751–759, 2014.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 0-934613-73-7.

M. Scanagatta, C. P. de Campos, G. Corani, and M. Zaffalon. Learning Bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1864–1872, 2015.

M. Scanagatta, G. Corani, C. P. de Campos, and M. Zaffalon. Learning treewidth-bounded Bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems 29*, pages 1462–1470. 2016.

M. Scanagatta, G. Corani, and M. Zaffalon. Improved local search in Bayesian networks structure learning. In *Proceedings of The 3rd International Workshop on Advanced Methodologies for Bayesian Networks*, volume 73 of *Proceedings of Machine Learning Research*, pages 45–56, 2017.

M. Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3): 1–22, 2010.

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

V. Ziegler. Approximation algorithms for restricted Bayesian network structures. *Information Processing Letters*, 108(2):60–63, Sept. 2008.

Article III

Convergence of Feedback Arc Set-Based Heuristics for Linear Structural Equation Models

Pierre Gillot & Pekka Parviainen

Proceedings of the 11th International Conference on Probabilistic Graphical Models, PMLR 186:157-168, 2022

Convergence of Feedback Arc Set-Based Heuristics for Linear Structural Equation Models

Pierre Gillot Pierre.Gillot@uib.no

Pekka Parviainen Pekka.Parviainen@uib.no

University of Bergen HIB - Thormøhlens gate 55 Postboks 7803 5020 Bergen

Abstract

Score-based structure learning in Bayesian networks, where local structures in the graph are given a score and one seeks to recover a high-scoring DAG from data, is an NP-hard problem. While the general learning problem is combinatorial, the more restricted framework of linear structural equation models (SEMs) enables learning Bayesian networks using continuous optimization methods. Large scale structure learning has become an important problem in linear SEMs and many approximate methods have been developed to address it. Among them, feedback arc set-based methods learn the DAG by alternating between unconstrained gradient descent-based step to optimize an objective function and solving a maximum acyclic subgraph problem to enforce acyclicity. In the present work, we build upon previous contributions on such heuristics by first establishing mathematical convergence analysis, previously lacking; second, we show empirically how one can significantly speed-up convergence in practice using simple warmstarting strategies.

Keywords: Bayesian networks; Structure learning; Linear structural equation models; Convex optimization; Maximum acyclic subgraph.

1. Introduction

Bayesian networks are a class of probabilistic graphical models where the conditional independencies between variables are expressed using a directed acyclic graph (DAG). We are interested in the structure learning problem, that is, how to construct the DAG based on data. We take a score-based approach to structure learning where every DAG is assigned a score based on how well it fits to the data and one tries to find a DAG that optimizes the score. Typically, the score decomposes into a sum of local scores that are computed for node-parent set pairs. This yields a combinatorial optimization problem where one picks a parent set for each node and tries to maximize the sum of the local scores while constraining the resulting graph to be acyclic. This problem is known to be NP-hard (Chickering, 1996).

In this paper, we concentrate on linear structural equation models (linear SEMs) which are a subclass of Bayesian networks. They are used to model continuous variables and the value of a variable depends linearly on values of its parents. From the learning perspective, linear SEMs simplify the optimization because the score function depends only on the arc weights and not the node-parent set pairs. However, the structure learning problem remains combinatorial due to the acyclicity constraint imposed to the graph.

Recently, Zheng et al. (2018) introduced a continuous acyclicity constraint that enables learning SEMs with continuous optimization instead of combinatorial optimization. However, learning DAGs using the acyclicity function proposed in Zheng et al. (2018) proves impractical in large scale settings owing to the complexity of the matrix exponential, which

exhibits cubic time complexity and quadratic space complexity with respect to the number of nodes. Various methods have been recently developed in order to circumvent this problem and enable learning large structures. As a rule, these methods avoid encoding acyclicity with hard constraints and instead formulate alternative problems that can be solved with lower complexity per iteration. We note however that these new problems remain largely non-convex in nature and one cannot hope to find a global minimizer in general. In Yu et al. (2021), a cyclic solution is first computed and then projected to the DAG space using a novel characterization based on Hodge decomposition of graphs. In Zhu et al. (2021), the hard constraint encoded by the acyclicity function is relaxed and an upped-bound on the spectral radius of a non-negative adjacency matrix of the graph is derived instead. In Dong and Sebag (2022), low-rank solutions are combined with an efficient approximation for the computation of the gradient of the acyclicity function.

Alternatively, combining feedback arc set heuristics with continuous optimization schemes has proved successful in learning large scale DAGs. Such methods consist in decoupling the optimization of the objective function from acyclicity itself by alternating between fast gradient-based optimization steps without acyclicity and projection of cyclic solutions to "close" acyclic approximations; while Park and Klabjan (2017) greedily fit parameters of a newly discovered acyclic structure at every step, instead Gillot and Parviainen (2022) dynamically construct a sequence of convex objective functions penalized to remain in the vicinity of a trail of acyclic solutions discovered online, resulting in better scalability but losing theoretical guarantees on the convergence of their method.

The present paper has two contributions. The first contribution is theoretical. We show that the *ProxiMAS* algorithm presented in Gillot and Parviainen (2022) converges under certain conditions (Lemma 2, Theorem 3). We note that the conditions are stronger than for the *GD* algorithm by Park and Klabjan (2017). Second, we analyse the convergence of ProxiMAS empirically. We also show that clever warmstarting strategies can lead to substantially faster convergence for feedback arc set heuristic-based structure learning.

2. Background

2.1 Linear Structural Equation Models and Bayesian Network Structure Learning

Let V be a node set. Furthermore, let $G = (V, A)$ be a DAG where A is the arc set. The parent set of node v in G is denoted by A_v.

Formally, a Bayesian network is a pair (G, Θ) where its structure G is a DAG and Θ are its parameters. The joint distribution factorizes as follows:

$$P(V) = \prod_{v \in V} P(v|A_v, \theta_v)$$

where θ_v are parameters of the conditional distribution of v given its parents.

Linear SEMs are a special case of Bayesian networks. To specify a model, we have a weight matrix $W \in \mathbb{R}^{d \times d}$, where d is the cardinality of the node set V. The weight matrix specifies both the structure and parameters of the Bayesian network. Specifically, $W(i, j) \neq 0$ entails there is an arc going from i to j in the DAG. Given a d-dimensional

data vector x, a linear SEM can be written as

$$x = xW + \epsilon$$

where ϵ is a d-dimensional error vector. The elements of ϵ are independent. The data consists of n such samples x, forming a data matrix $X \in \mathbb{R}^{n \times d}$. We note that when the errors are Gaussian, linear SEMs encode multivariate Gaussian distributions.

To learn a linear SEM, we need to constrain W to represent an acyclic graph. Furthermore, one typically uses the least squares loss and adds a regularization term inducing sparsity in the structure. Thus, the objective function in structure learning becomes

$$\underset{W}{\operatorname{argmin}} \frac{1}{2n}||XW - X||^2 + \lambda g(W) \quad \text{s.t. } W \text{ is acyclic} \tag{1}$$

where $\|\cdot\|$ is the Frobenius norm and $g(W)$ is for regularization, whose strength is controlled by the hyperparameter $\lambda > 0$.

2.2 Feedback Arc Set-Based Structure Learning

At a general level, feedback arc set-based methods learn a DAG (under the linear SEMs framework) by iteratively repeating the following steps:

- Given an acyclic graph, find a graph (possibly cyclic) which is better in terms of the objective function value.
- Given a cyclic graph, find a close acyclic graph by approximately solving a maximum acyclic subgraph instance.

In particular, these methods entirely decouple acyclicity from the optimization process itself, via the integration of (weighted) maximum acyclic subgraph (MAS) problems, whose definition we recall now: given a directed graph $G = (V, E)$ and a weight function $w(e)$ that assigns a weight for each arc $e \in E$, the goal is to find an acyclic graph $G' = (V, E')$ such that $E' \subset E$ and $\sum_{e \in E'} w(e)$ is maximized. The dual problem is called the feedback arc set (FAS) problem: given a directed graph $G = (V, E)$ and a weight function $w(e)$, the goal is to find an arc set $E'' \subset E$ such that $G'' = (V, E \setminus E'')$ is acyclic and $\sum_{e \in E''} w(e)$ is minimized. Given a cyclic graph G as input, it is well known that G' is an optimal solution of MAS if and only if $G \setminus G'$ is an optimal solution of FAS. Moreover, both problems are NP-hard (Karp, 1972).

Intuitively, using the maximum acyclic subgraph problem in order to learn linear SEMs DAGs is sensible, in that given any acyclic solution to linear SEMs, one can always extend this solution into a tournament (a dense acyclic graph) having the exact same score, by completing the solution with zero-weight arcs. Unlike traditional approaches that involve a smooth characterization of acyclicity (Zheng et al., 2018; Ng et al., 2020; Yu et al., 2021; Zhu et al., 2021; Dong and Sebag, 2022), feedback arc set-based methods also offer the clear advantage that they return strictly acyclic solutions, in the sense that one never needs to threshold a solution as a form of postprocessing in order to recover a DAG.

Two variants have been studied so far. In Park and Klabjan (2017), the authors propose the GD algorithm which works by repeating the following steps:

1. Fix the structure of the last obtained acyclic solution, then fit the linear SEMs objective constrained by this structure to get a new fitted acyclic solution.

2. Make an unconstrained optimization step on the linear SEMs loss at the previously obtained fitted acyclic solution to get a new cyclic solution.

3. Project the previously obtained cyclic solution to its maximum acyclic subgraph approximation to get a new acyclic solution.

The key design choice in GD lies in the fact that unconstrained optimization steps are only performed after a newly found structure has been fitted with respect to the linear SEMs objective. From a theoretical perspective, this leads to a simplified convergence analysis and GD is guaranteed to converge in a fix number of iterations under mild conditions (see (Park and Klabjan, 2017), Lemma 1). On the practical side however, the GD algorithm "greedily" explores the search space which can lead to overfitting and incurs solving a LASSO subproblem for every node in the graph at every iteration, heavily impacting the scalability of the algorithm. In Gillot and Parviainen (2022), an alternative approach is proposed that would fix the scalability concern observed in GD. This new variant changes steps 1 and 2 from GD (step 3 is left unchanged) as follows:

1'. Construct a new objective function as the sum of the linear SEMs loss plus a least-squared term penalizing deviation from the last obtained acyclic solution.

2'. Make an unconstrained optimization step on the previously constructed objective function to get a new cyclic solution.

In other words, this second approach jumps from an acyclic structure to another, without fitting these structures to optimality. As a trade-off, the optimization process now evolves dynamically, making a convergence analysis less straightforward (and such analysis is presently missing, to the best of our knowledge). The pseudocode of this variant is described in Algorithm 1.

Algorithm 1 (Gillot and Parviainen, 2022)

Input: $X \in \mathbb{R}^{n \times d}$, $\lambda > 0$, $\mu > 0$

$\widetilde{W}_0, W_0 = 0^{d \times d}$

for $1 \leq k \leq \dots$ **do**

  New objective function: $\phi_k \colon W \mapsto \frac{1}{2n}\|XW - X\|^2 + \frac{\mu}{2}\|W - W_{k-1}\|^2 + \lambda\|W\|_1$

  Optimization step: $\widetilde{W}_k = \text{step}(\phi_k,\ \text{optimizer})$

  MAS projection: $W_k = \text{MAS}(\widetilde{W}_k)$

In short, the algorithm keeps track of both cyclic and acyclic solutions, represented respectively by $\widetilde{W}_k$ and W_k. At every iteration, a new objective function is constructed: let $f \colon W \mapsto \frac{1}{2n}\|XW - X\|^2$ and $g \colon W \mapsto \lambda\|W\|_1$ represent the linear SEMs loss and the sparsity inducing penalization term respectively; let $f_k : W \mapsto f(W) + \frac{\mu}{2}\|W - W_{k-1}\|^2$ represent the linear SEMs loss penalized to remain in the vicinity of the previously discovered acyclic structure; then the new objective function is $\phi_k = f_k + g$, where both f_k and g are convex,

the f_k are differentiable and their gradient share the same optimal Lipschitz constant $L = \frac{1}{n}\|X^tX + n\mu I_d\|_*$ (where $\|\cdot\|_*$ is the spectral norm). From a practical standpoint, this means that one can exploit the stationary properties of the ϕ_k in order to make fast progress with a proximal gradient-based optimizer, though Algorithm 1 can embed instead any gradient-based first-order optimizer. The authors dub the former *ProxiMAS* and the latter *OptiMAS*. We note that while the objective functions ϕ_k are all convex, the overall optimization scheme itself remains largely non-convex, in that every function ϕ_k carries structural information which can evolve in a non-convex fashion from an iteration to another. This structural information is encapsulated within the acyclic solutions W_k. In order to construct them a feedback arc set heuristic is used, which at every iteration k constructs a topological order π_k from the cyclic solution $\widetilde{W}_k$; the acyclic projection W_k is obtained by nullifying those weights in $\widetilde{W}_k$ corresponding to feedback arc set arcs (with respect to π_k). Both Park and Klabjan (2017) and Gillot and Parviainen (2022) make use of a variant of the greedy feedback arc set heuristic originally presented in Eades et al. (1993). More specifically, this variant iteratively constructs a topological order from its last/rightmost up to its first/leftmost element. A node is greedily selected if it has the smallest sum of incoming squared weights among the remaining nodes. In other words, this heuristic treats forward arcs (with respect to the constructed topological order) as feedback arc set arcs. It is described in Algorithm 2.

Algorithm 2 Greedy feedback arc set heuristic

Input: $\widetilde{W} \in \mathbb{R}^{d\times d}$

1: $V_1 = \{0, \dots, d-1\}, \pi = 0^{d\times 1}$
2: **for** $1 \leq r \leq d$ **do**
3: $\quad \pi[-r] = \underset{j\in V_r}{\operatorname{argmin}} \left\|\widetilde{W}[:, j]\right\|^2_{V_r\setminus\{j\}}$
4: $\quad V_{r+1} = V_r \setminus \pi[-r]$
5: **return** π

3. Convergence Analysis

Minimizing a composite convex function is a standard problem in convex analysis: let $\phi \coloneqq f + g : U \mapsto \mathbb{R}$ denote a composite convex function on a convex open set $U \subset \mathbb{R}^m$ such that: f and g are convex on U, f is differentiable and its gradient is Lipschitz-continuous with constant L on U. Then it is well known that the non-accelerated proximal gradient descent optimizer generating the sequence $(x_k)_k$ defined as

$$x_k = \underset{x\in U}{\operatorname{argmin}} \left\{ \frac{\gamma_k^{-1}}{2}\left\|x - \left(x_{k-1} - \gamma_k \nabla f(x_{k-1})\right)\right\|^2 + g(x)\right\} \quad \text{where } 0 < \gamma_k \leq L^{-1} \tag{2}$$

achieves $\mathcal{O}\left(\frac{1}{k}\right)$ convergence rate in function value (where k is the number of iterations) (Beck and Teboulle, 2009a). A key aspect of the convergence analysis is to show that one in fact always has (see for instance (Beck and Teboulle, 2009a), Lemma 1.6):

$$0 < \gamma_k \leq L^{-1} \implies \phi(x_k) \leq \phi(x_{k-1}) - \frac{\gamma_k^{-1}}{2}\|x_k - x_{k-1}\|^2, \tag{3}$$

that is the proximal gradient descent update generates a sequence guaranteed to decrease the objective function value. Algorithm 1 equipped with the same convex optimizer subtly

differs from this framework, in that at every iteration a convex descent step is performed on a new composite convex function $\phi_k = f_k + g$: this describes a dynamic system and the notion of optimal solution is ill-defined, thus the $\mathcal{O}\big(\frac{1}{k}\big)$ convergence rate in function value is lost. In the rest of this section, by ProxiMAS we refer to Algorithm 1 equipped with both Algorithm 2 for the FAS heuristic and the non-accelerated proximal gradient descent optimizer described above, i.e. ProxiMAS makes convex descent steps of the form:

$$\widetilde{W}_k = \underset{W\in\mathbb{R}^{d\times d}}{\operatorname{argmin}} \left\{ \tfrac{\gamma_k^{-1}}{2} \Big\| W - \Big(\widetilde{W}_{k-1} - \gamma_k \nabla f_k\big(\widetilde{W}_{k-1}\big)\Big) \Big\|^2 + \lambda \|W\|_1 \right\} \quad \text{where } 0 < \gamma_k \leq L^{-1}, \tag{4}$$

where all f_k have Lipschitz-continuous gradient with the same constant L. We aim to derive a set of conditions such that ProxiMAS converges to a fixed acyclic structure in a finite number of iterations, that is the acyclic solutions W_k have the same support, or equivalently the topological orders π_k constructed by Algorithm 2 are the same. Lemma 1 provides a necessary condition, agnostic from the choice of the optimizer (in Algorithm 1: line 4):

Lemma 1 *Let $\big(\widetilde{W}_k\big)_k$, $(W_k)_k$ and $(\pi_k)_k$ respectively denote the sequence of cyclic solutions, acyclic solutions and topological orders in Algorithm 1. Assume topological orders stabilize, i.e. $\exists k_1 : \forall k \geq k_1,\ \pi_k = \pi$. Then the following convergence condition necessarily holds:*

$$\exists k_0 : \forall k \geq k_0,\ \big\|\widetilde{W}_k - W_k\big\| \leq \big\|\widetilde{W}_k - W_{k-1}\big\|. \tag{5}$$

Proof Notice that $\big\|\widetilde{W}_k - W_k\big\|^2 \leq \big\|\widetilde{W}_k - W_{k-1}\big\|^2 \iff \big\|W_k - W_{k-1}\big\|^2 \geq 2\big\langle \widetilde{W}_k - W_k, W_{k-1}\big\rangle$. Assuming that for large k, $\pi_k = \pi$, one must then have $\big\langle \widetilde{W}_k - W_k, W_{k-1}\big\rangle = 0$. Indeed, non-zero values in $\widetilde{W}_k - W_k$ must correspond to forward arcs whereas non-zero values in W_{k-1} must correspond to backward arcs (both with respect to π for large enough k). ■

In order to get the convergence of acyclic solutions W_k, we must first ensure we get the convergence of cyclic solutions $\widetilde{W}_k$. We stress that by convergence we imply toward a local extremum and that converging does not guarantee good performance of found solutions, that is we are concerned with the stability of ProxiMAS. We prove the following:

Lemma 2 *Let $\big(\widetilde{W}_k\big)_k$, $(W_k)_k$ and $(\gamma_k)_k$ respectively denote the sequence of cyclic solutions, acyclic solutions and learning rates in ProxiMAS. Assume the learning rate decreases with rate $\mathcal{O}\big(\frac{1}{k^\alpha}\big)$ where $\alpha > 2$, and assume the convergence condition from Lemma 1 holds: $\exists k_0 : \forall k \geq k_0,\ \big\|\widetilde{W}_k - W_k\big\| \leq \big\|\widetilde{W}_k - W_{k-1}\big\|$. Then $\widetilde{W}_k$ admits a convergent subsequence.*

Proof Notice the ϕ_k have stationary properties (composite convex functions, same optimal Lipschitz constant L for the gradient of smooth components) hence Equation 3 holds for every ϕ_k at step k: $\forall k \geq 1,\ 0 < \gamma_k \leq L^{-1} \implies \phi_k\big(\widetilde{W}_k\big) \leq \phi_k\big(\widetilde{W}_{k-1}\big) - \frac{\gamma_k^{-1}}{2}\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\|^2$. Now, by definition: $\phi_k\big(\widetilde{W}_{k-1}\big) = \phi_{k-1}\big(\widetilde{W}_{k-1}\big) + \frac{\mu}{2}\Big(\big\|\widetilde{W}_{k-1} - W_{k-1}\big\|^2 - \big\|\widetilde{W}_{k-1} - W_{k-2}\big\|^2\Big)$. Due to the convergence condition, we thus get $\phi_k\big(\widetilde{W}_k\big) \leq \phi_k\big(\widetilde{W}_{k-1}\big) \leq \phi_{k-1}\big(\widetilde{W}_{k-1}\big)$ for large k, implying the (non-negative) sequence $\big(\phi_k\big(\widetilde{W}_k\big)\big)_k$ converges to a limit l. Furthermore, we can now write that for large k, $\frac{\gamma_k^{-1}}{2}\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\|^2 \leq \phi_{k-1}\big(\widetilde{W}_{k-1}\big) - \phi_k\big(\widetilde{W}_k\big)$. We then use the fact that the right-hand side in the previous inequality is a telescopic term, along with $\phi_k\big(\widetilde{W}_k\big) \underset{k\to+\infty}{\longrightarrow} l$, to deduce that the infinite series $\sum_k \gamma_k^{-1}\big\|\widetilde{W}_k - \widetilde{W}_{k-1}\big\|^2$ converges;

necessarily, $\gamma_k^{-1}\left\|\widetilde{W}_k - \widetilde{W}_{k-1}\right\|^2 = o(1)$ [1] holds, which in turn implies $\left\|\widetilde{W}_k - \widetilde{W}_{k-1}\right\| = \mathcal{O}\left(\sqrt{\gamma_k}\right)$. Now by assumption $\sqrt{\gamma_k} = \mathcal{O}\left(\frac{1}{k^{\alpha/2}}\right)$ where $\alpha > 2$, hence $\left\|\widetilde{W}_k - \widetilde{W}_{k-1}\right\| = \mathcal{O}\left(\frac{1}{k^{\beta}}\right)$ where $\beta > 1$ such that the infinite series $S \coloneqq \sum_k \left\|\widetilde{W}_k - \widetilde{W}_{k-1}\right\|$ converges. The triangular inequality finally yields:

$$\forall K,\ \left\|\widetilde{W}_K - \widetilde{W}_0\right\| = \left\|\sum_{k\leq K}\widetilde{W}_k - \widetilde{W}_{k-1}\right\| \leq \sum_{k\leq K}\left\|\widetilde{W}_k - \widetilde{W}_{k-1}\right\| \leq S < +\infty,$$

therefore $\sup_k \left\|\widetilde{W}_k\right\| < +\infty$. The Bolzano-Weierstrass theorem concludes the proof. ■

We are now ready to present our main result:

Theorem 3 *Let $\left(\widetilde{W}_k\right)_k$ and $(\pi_k)_k$ respectively denote the sequence of cyclic solutions and topological orders in ProxiMAS. Assume $\left(\widetilde{W}_k\right)_k$ admits a converging subsequence: $\widetilde{W}_* \coloneqq \lim_{k\to+\infty}\left(\widetilde{W}_{\psi(k)}\right)_k$. Define π_* to be the topological order constructed by Algorithm 2 given $\widetilde{W}_*$ as input and assume for all $r \in [1, d]$, Algorithm 2 makes a strictly optimal decision when constructing $\pi_*[-r]$ (i.e. argmin in Algorithm 2: line 3 is strict at every step r given $\widetilde{W}_*$ as input). Then the topological orders constructed by ProxiMAS in the subsequence ψ stabilize after a finite number of iterations: $\exists k' : \forall k \geq k',\ \pi_{\psi(k)} = \pi_*$.*

Proof idea The proof is technical and revolves around a similar argument as in Park and Klabjan (2017): Lemma 1. Due to space constraints, we leave out the full proof. ■

We note that the assumption in Theorem 3 is mild: although one never has access to the limit of a converging subsequence, arc weights are continuous thus Algorithm 2 easily makes strictly optimal choices. However, columns of zeros can occur in practice (e.g. when learning sparse structures), in which case convergence cannot be guaranteed. We also comment on Lemma 2's assumptions: the convergence condition from Lemma 1 ensures feedback arc set costs eventually become less than the distance between past acyclic solutions and new cyclic solutions; the learning rate must decrease sufficiently fast which can deteriorate the quality of found solutions. These two assumptions are not needed in the theoretical convergence of GD (Park and Klabjan, 2017), meaning the theoretical convergence of ProxiMAS (Gillot and Parviainen, 2022) is weaker. This was expected since unlike GD, ProxiMAS does not solve LASSO subproblems at every iteration.

4. Experiments

We now conduct an empirical study of feedback arc set-based heuristics for linear SEMs. This study is divided into three experiments. First, we empirically validate the convergence analysis of ProxiMAS by investigating the stability of the method in various settings; second, we assess the influence of the MAS penalization hyperparameter μ with different convex optimizers in ProxiMAS; third, we compare different warmstarting strategies in order to speed-up the practical convergence of feedback arc set-based heuristics.

1. If $k\gamma_k^{-1}\left\|\widetilde{W}_k - \widetilde{W}_{k-1}\right\|^2$ has a limit in $\mathbb{R}_+ \cup \{+\infty\}$, one in fact has $\gamma_k^{-1}\left\|\widetilde{W}_k - \widetilde{W}_{k-1}\right\|^2 = o\left(\frac{1}{k}\right)$ (due to the divergence of the harmonic series); in that case $\gamma_k = \mathcal{O}\left(\frac{1}{k^{\alpha}}\right)$ where $\alpha > 1$ suffices for Lemma 2 to hold.

4.1 Setup

We consider a setup similar to that found in Zheng et al. (2018). Data generation is as follows: we start by generating an undirected graph with d nodes from two classes of random graphs, namely Erdős-Rényi ("ER") and scale-free ("SF"). Assuming the graph is sampled to have average degree δ, we refer to this graph as "ERδ" (respectively "SFδ"). A random permutation is then sampled and assigned to the graph which yields a DAG. Next, arc weights W are uniformly sampled in the range $[-2,-0.5] \cup [0.5,2]$. The last step is to generate linear SEMs samples $X = \mathcal{E}(I-W)^{-1}$, where $\mathcal{E} \in \mathbb{R}^{n \times d}$ represents n noise samples, with $n = 0.1 \times d$ (low sample count) or $n = 10 \times d$ (large sample count). We restrict $\mathcal{E}$ to be generated from Gaussian noise only and study both the equal variance setting ("EV": all σ equal 1.0) and the non-equal variance setting ("NV": all σ uniformly sampled in $[0.5, 1.5]$). In all considered experiments, 20 instances are randomly generated as described above; we represent variance in our figures with shaded regions.

We always fix the sparsity-inducing hyperparameter λ to 0.1, as in Gillot and Parviainen (2022). The number of iterations allowed for tested methods is always set to ten times the number of nodes (e.g. 10000 iterations when $d = 1000$). Every 100 iterations a snapshot is recorded and different metrics are extracted, such as the loss (Equation 1) of the current acyclic solution and its average precision with respect to the true DAG (Markov equivalence is ignored). We consider as well metrics to assess the convergence of tested heuristics, such as: the order matching metric which gives the percentage of matching nodes in two consecutive topological orders constructed by Algorithm 2; the convergence condition metric which evaluates the quantities $\left\|\widetilde{W}_k - W_k\right\| - \left\|\widetilde{W}_k - W_{k-1}\right\|$ (remember that these quantities must remain negative after a finite number of iterations to guarantee structural convergence, see Lemma 1). Both the order matching and the convergence condition metrics are averaged over the past 100 iterations to get smoother estimates. Implementation is based on pytorch 1.10 and experiments were run on a cluster with Intel Xeon-Gold 6138 2.0 GHz / 6230R 2.1 GHz CPU cores. Table 1 lists the hyperparameters for each experiment. To save space we only show a subset of all figures.

Exp	d	δ	μ	**Convex opti**	**Const lr %**	**Cyclic %**	**Convex %**
1	1000	$1, 2, 4$	10^{δ}	I, F	$0, 50, 100$	0	100
2	1000	4	$10^{i}, 1 \leq i \leq \delta$	I, F, G, N	100	0	100
3	2000	$4, 8$	10^{δ}	F	100	$0, 50$	$0, 20, \ldots, 100$

Table 1: Experiments hyperparameters (I: ISTA; F: FISTA; G: Greedy FISTA; N: Nesterov)

4.2 Experiment 1

In the first experiment we consider the classical iterative shrinkage-thresholding algorithm (ISTA) implementing in closed-form Equation 2 and its well known accelerated variant FISTA (Beck and Teboulle, 2009b), with different learning rate strategies. The learning rate is implemented to decrease with rate $\frac{1}{k^{1.001}}$, but remains constant for $x\%$ of the total number of iterations before decreasing (x varies as described in Table 1: column "Const lr %"). Fig-

ure 1 illustrates Experiment 1. Looking at the average precision curves, clearly ProxiMAS stabilizes once the learning rate starts decreasing (orange curves). When applied too early, decrease in learning rate hurts performance (pink curves). Comparing the optimizers, ISTA is a much slower learner than FISTA and fails to learn denser graphs ($\delta = 4$). Looking at the convergence condition metric, a decreasing learning rate yields infinitesimal quantities $\left\|\widetilde{W}_k - W_k\right\| - \left\|\widetilde{W}_k - W_{k-1}\right\|$ for ISTA. With constant learning rate (teal curves) FISTA learns fast and eventually satisfies the convergence condition $\left\|\widetilde{W}_k - W_k\right\| \leq \left\|\widetilde{W}_k - W_{k-1}\right\|$. Experiment 1 suggests that ProxiMAS is stable in practice: even when the convex optimizer is accelerated, the learning rate is constant and the convergence condition from Lemma 1 does not exactly hold, ProxiMAS reaches a performance plateau.

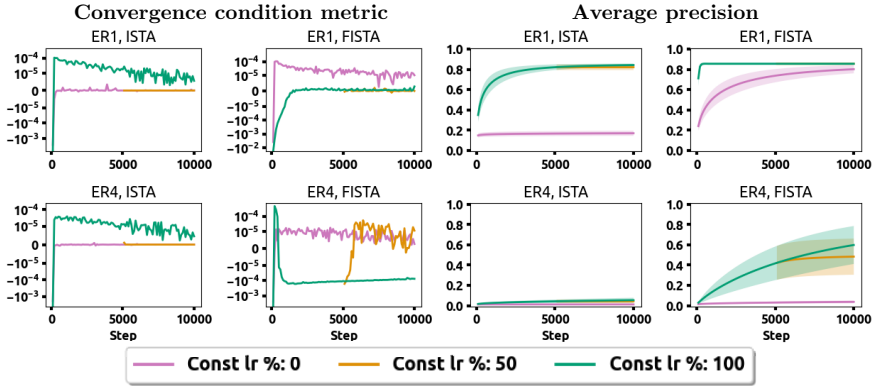

Figure 1: Experiment 1: learning rate policy varies ($d = 1000$, $n = 10000$, NV).

4.3 Experiment 2

In the second experiment we compare the behaviour of various convex optimizers with respect to the hyperparameter μ controlling the strength of the MAS penalization terms. In addition to the classical ISTA and FISTA optimizers, we consider the Greedy FISTA variant described in Liang et al. (2022) that relies on restarting. We consider as well the Nesterov variant outlined in Nesterov (2014) (refer to "Constant Step Scheme, III") which unlike aforementioned optimizers exploits the fact that the objective functions ϕ_k are μ-strongly convex rather than just convex. We focus on denser graphs ($\delta = 4$) for which obtaining good solutions is challenging. Figure 2 illustrates Experiment 2. We notice that no matter the choice of μ, ISTA satisfies the convergence condition but fails to learn anything significant. Both FISTA and its greedy variant display similar behavior and performance as they learn significantly better solutions when μ is set high. This explains the lower performance of ProxiMAS in Gillot and Parviainen (2022) for denser graphs ($\delta = 4$) since the authors used FISTA with $\mu = 20$ in all experiments. As a rule, we observe that the larger the μ the more the convergence condition $\left\|\widetilde{W}_k - W_k\right\| \leq \left\|\widetilde{W}_k - W_{k-1}\right\|$ is satisfied. Interestingly, the behavior of the Nesterov optimizer is opposite to that of FISTA: it learns better DAGs when

μ is set smaller. Our hypothesis is that since it accounts for the μ-strong convexity of the ϕ_k objectives, it optimizes too well the MAS penalization terms $\frac{\mu}{2}\|W - W_{k-1}\|^2$, preventing progress due to new cyclic solutions $\widetilde{W}_k$ remaining too close to last acyclic solutions W_{k-1}.

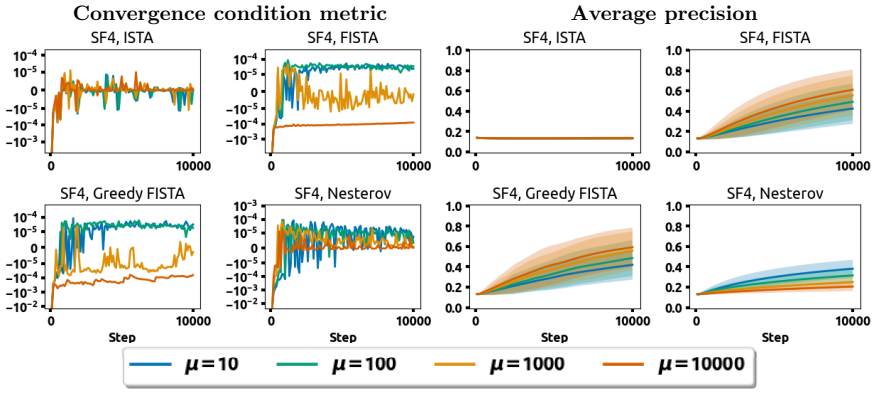

Figure 2: Experiment 2: convex optimizer and μ vary ($d=1000$, $n=10000$, NV).

4.4 Experiment 3

In the third experiment, we investigate different warmstarting strategies in order to speed-up practical convergence of FAS-based heuristics. A first form of warmstarting consists in presolving Algorithm 1 without enforcing acyclicity ($\iff \mu=0$); a second form is to first use a convex optimizer, then use a non-convex one. The hyperparameter "Cyclic %" controls the ratio of iterations dedicated to "cyclic presolving"; "Convex %" controls the ratio of iterations (excluding cyclic presolving) that use the FISTA optimizer before swapping for the adaptative optimizer Adam (Kingma and Ba, 2014) (see Table 1). For instance, assuming 20000 iterations in total, Cyclic %=50 and Convex %=20 means cyclic presolving occurs up to iteration 10000, FISTA is used up to iteration 12000, after which we use Adam. Figure 3 illustrates Experiment 3. Based on the empirical study in Gillot and Parviainen (2022), cyclic presolving is ideal when learning very sparse DAGs ($\delta \leq 2$). Experiment 3 suggests that when $\delta \geq 4$, an hybrid optimizer strategy yields superior performance boost. These boosts are more pronounced when both the number of nodes d and the number of samples n are sufficiently large. We suspect the non-linear nature of Adam makes it efficient at learning complex structures, but the non-differentiability of the ϕ_k in Algorithm 1 could explain why Adam benefits from warmstarting instead of starting from the zero matrix.

5. Discussion

We have studied theoretical convergence and demonstrated that FAS-based heuristics as presented in Gillot and Parviainen (2022) with a non-accelerated convex optimizer have

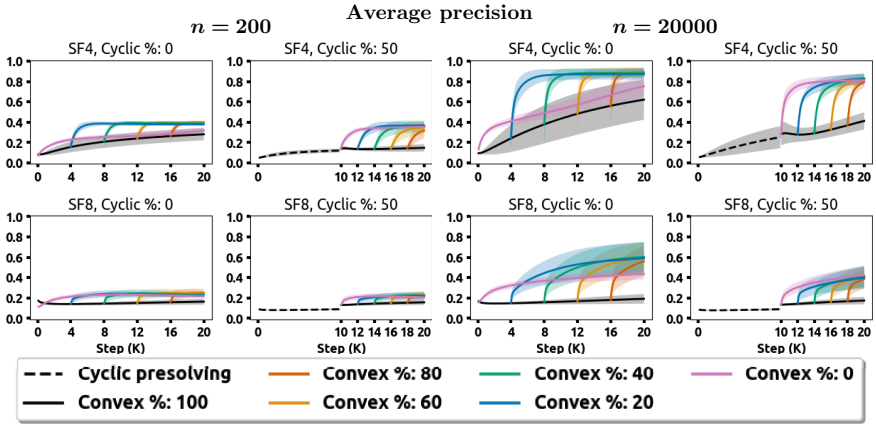

Figure 3: Experiment 3: warmstarting strategy varies ($d = 2000$, NV).

provable structural convergence (of subsequences) in a finite number of iterations, albeit a weaker form than in Park and Klabjan (2017). More specifically, additional assumptions are necessary, in the form of a) a learning rate decreasing sufficiently fast and b) a convergence condition ensuring feedback arc set costs eventually become less than the distance between past acyclic solutions and new cyclic solutions. Our empirical study provides evidence that these assumptions are mild: in practice, FAS-based heuristics are sufficiently stable in that they tend to reach a performance plateau even with constant learning rate and using an accelerated convex optimizer, thus one can decrease the learning rate only at a later stage, as a safeguard. Moreover, our study suggests that setting the hyperparameter μ sufficiently high helps satisfying the convergence condition, especially when learning denser acyclic structures. Finally, we investigated different forms of warmstarting strategies to speed-up the practical convergence of FAS-based heuristics. We uncovered an interesting effect, in that an hybrid optimizer strategy (convex optimizer followed by non-convex optimizer) consistently provides tangible acceleration when learning sufficiently dense and large DAGs.

Acknowledgments

Parts of this work have been done in the context of CEDAS (Center for Data Science, University of Bergen, UiB). The computations were performed on resources provided by UNINETT Sigma2 - the National Infrastructure for High Performance Computing and Data Storage in Norway. We thank Madhumita Kundu for her valuable input.

References

A. Beck and M. Teboulle. Gradient-based algorithms with applications to signal-recovery problems. In *Convex Optimization in Signal Processing and Communications*, page 42–88.

Cambridge University Press, 2009a.

A. Beck and M. Teboulle. A fast Iterative Shrinkage-Thresholding Algorithm with application to wavelet-based image deblurring. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 693–696, 2009b.

D. M. Chickering. Learning Bayesian Networks is NP-Complete. In *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer New York, 1996.

S. Dong and M. Sebag. From graphs to DAGs: a low-complexity model and a scalable algorithm. *CoRR*, 2022.

P. Eades, X. Lin, and W. Smyth. A fast and effective heuristic for the feedback arc set problem. In *Information Processing Letters*, volume 47, pages 319–323, 1993.

P. Gillot and P. Parviainen. Learning Large DAGs by Combining Continuous Optimization and Feedback Arc Set Heuristics. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, 2022.

R. M. Karp. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations*, pages 85–103. Springer US, 1972.

D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2014.

J. Liang, T. Luo, and C.-B. Schönlieb. Improving "Fast Iterative Shrinkage-Thresholding Algorithm": Faster, Smarter, and Greedier. In *SIAM Journal on Scientific Computing*, volume 44, pages A1069–A1091, 2022.

Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer New York, 2014.

I. Ng, A. Ghassami, and K. Zhang. On the Role of Sparsity and DAG Constraints for Learning Linear DAGs. In *Advances in Neural Information Processing Systems*, 2020.

Y. W. Park and D. Klabjan. Bayesian Network Learning via Topological Order. In *Journal of Machine Learning Research*, volume 18, pages 1–32, 2017.

Y. Yu, T. Gao, N. Yin, and Q. Ji. DAGs with No Curl: An Efficient DAG Structure Learning Approach. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 12156–12166, 2021.

X. Zheng, B. Aragam, P. Ravikumar, and E. P. Xing. DAGs with NO TEARS: Continuous Optimization for Structure Learning. In *Advances in Neural Information Processing Systems*, 2018.

R. Zhu, A. Pfadler, Z. Wu, Y. Han, X. Yang, F. Ye, Z. Qian, J. Zhou, and B. Cui. Efficient and Scalable Structure Learning for Bayesian Networks: Algorithms and Applications. In *IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2613–2624, 2021.

Graphic design: Communication Division, UiB / Print: Skipnes Kommunikasjon AS

uib.no

ISBN: 9788230869017 (print)
9788230857007 (PDF)