Learning decomposable models by coarsening

George Orfanides GORFANIDES@BCAMATH.ORG

Aritz Pérez APEREZ@BCAMATH.ORG

Basque Center for Applied Mathemtatics (BCAM)

Abstract

During the last decade, some exact algorithms have been proposed for learning decomposable models by maximizing additively decomposable score functions, such as Log-likelihood, BDeu, and BIC. However, up to the date, the proposed exact approaches are practical for learning models up to 20 variables. In this work, we present an approximated procedure that can learn decomposable models over hundreds of variables with a remarkable trade-off between the quality of the obtained solution and the amount of the computational resources required. The proposed learning procedure iteratively constructs a sequence of coarser decomposable (chordal) graphs. At each step, given a decomposable graph, the algorithm adds the subset of edges due to the actual minimal separators that maximizes the score function while maintaining the chordality. The proposed procedure has shown competitive results for learning decomposable models over hundred of variables using a reasonable amount of computational resources. Finally, we empirically show that it can be used to reduce the search space of exact procedures, which would allow them to address the learning of high-dimensional decomposable models.

Keywords: Probabilistic graphical models, decomposable models, efficient learning

1. Motivation

This work tackles the problem of learning decomposable models (DM) from data. The family of DMs is a subclass of Bayesian networks with an undirected chordal graph (decomposable graph, DG) representation. The family of DMs is one of the most important probabilistic graphical models due to its theoretical properties (Lauritzen, 1996). For instance, they represent probability distributions that factorize according to a product of lower-order marginals, and they are the basis of the most exact probabilistic inference algorithms.

The problem of learning DMs consists of learning the structure and the parameters. Given a data set, the parametric learning of DMs can be performed in closed form using likelihood estimate and Bayesian estimates. Thus, in this work, we concentrate on the structural learning of DMs.

There are two main approaches to deal with the structural learning problem: To find the DG that better codifies the conditional (in)dependence relations between the involved random variables (de Campos and Huete, 1997), e.g. PC algorithm (Spirtes et al., 2000); and to find the structure that maximizes a quality measure, such as Bayesian Dirichlet equivalent uniform score (BDeu) (Heckerman et al., 1995). In this work, we focus on this last approach from a combinatorial optimization point of view over the space of DGs. A particularly interesting family of quality measures correspond to the additively decomposable scores (Koller and Friedman, 2009), e.g. log-likelihood, Bayesian information criteria (BIC) (Schwarz, 1978) and BDeu (Heckerman et al., 1995). These scores can be expressed in terms of sums of measures defined over subsets of vertices.

Problem 1 (Maximum weighted decomposable graph (MWDG)) *Let $\mathcal{G}$ be the set of DGs defined over n vertices, V, and let w be a score function (weight) that assigns a real number to every decomposable graph,*

$w(G) \mapsto \mathbb{R}$, *that is additively decomposable over all the complete subsets of the given DG:*

$$w(G) = \sum_{R \in \mathcal{R}(G)} w(R), \tag{1}$$

where $\mathcal{R}(G) \subseteq 2^V$ *is the set of all complete subsets of vertices of* G*. The maximum weighted DG (MWDG) problem is defined as finding a graph* $G \in \mathcal{G}$ *that maximizes the score function* $w(G)$*:*

$$\arg\max_{G \in \mathcal{G}} w(G). \tag{2}$$

The MWDG problem is NP-hard (Dasgupta, 1997), and during the last decade, several exact procedures have been proposed to solve it (Corander et al., 2013; Kangas et al., 2014; Studený and Cussens, 2016; Janhunen et al., 2017; Rantanen et al., 2017; Studený and Cussens, 2017). In (Corander et al., 2013) the authors propose a junction tree characterization described in terms of constraints over separators. They translate these constraints into different optimization problems, such as the Maximum Satisfiability Problem. Janhunen et al. (2017) replace the balancing condition and maximum spanning tree considerations proposed in (Corander et al., 2013) by a perfect elimination ordering, and they obtain a more compact representation in terms of model constraints. Kangas et al. (2014) propose a recursive characterization of junction trees that is used together with a dynamic programming algorithm for tackling the MWDG problem. In (Rantanen et al., 2017) a branch and bound approach is proposed based on an alternative recursive characterization of the junction trees. They combine the branch and bound algorithm with dynamic programming for upper bounding the score function and pruning the search space. Studený and Cussens (2017) propose an alternative characterization of DGs using characteristic imsets (Hemmecke et al., 2012), binary vectors indexed by subsets of nodes. The space of DGs is represented as the convex hull of the characteristic imsets associated to each DG. They propose a set of inequalities that appear to define the facets of the convex hull, and a method for finding the most suitable inequality as the base for a cutting plane method. Up to the date, the proposed exact approaches are practical for learning models up to 20 variables, and they can not deal with the MWDG problem in high-dimensional domains.

An alternative to tackle the problem in higher dimensions is to consider a reduced search space. A particularly important reduction of the search space can be performed a priori by bounding the maximum clique size K (Kangas et al., 2014; Studený and Cussens, 2017). For instance, for $K = 2$ the problem reduces to finding a maximum weighted forest, which can be solved using Prim's algorithm with a computational complexity of $\mathcal{O}(n^2)$ (Eisner, 1997), where n is the number of vertices of the problem. In (Studený and Cussens, 2017), the authors suggest a more general way for reducing the search space using a set of subsets of nodes closed under inclusion.

Another alternative is to use approximate algorithms, which sacrifice the quality of the learned DG to achieve a reduction in computational cost. In this work we are particularly interested in greedy algorithms that construct the approximation to the MWDG problem incrementally (Malvestuto, 1991; Srebro, 2000; Deshpande et al., 2001; Karger and Srebro, 2001; Chechetka and Guestrin, 2008; Malvestuto, 2012; Pérez et al., 2016). From a combinatorial optimization point of view, greedy algorithms belong to the family of local search algorithms, and they mainly differ on the neighborhood considered in the iterative construction of the DG (Aarts et al., 2003), i.e. the set of DGs that can be obtained from a given one. In (Malvestuto, 1991), given a K value, the authors propose a local search algorithm that considers the addition of a clique that is of size K and shares $K - 1$ vertices with a previously added clique. Deshpande et al. (2001) propose an efficient implementation of a local search algorithm which neighborhood is given by all the DGs that can be obtained by the addition of a single edge. In (Srebro, 2000; Karger and Srebro, 2001) the authors present an approximate algorithm for learning K-hypertrees, a special class of DGs with all the cliques of size K and minimal separators of size $K - 1$. The algorithm is shown to have weak theoretical guarantees at expenses of being exponential in K. Chechetka and Guestrin (2008) propose a probably approximately correct algorithm for leaning DGs with bounded treewidth with a computational complexity exponential in the maximum clique size considered. In (Malvestuto, 2012), the authors propose an iterative deletion procedure of a set of edges for the special case of K-hypertrees. Pérez et al. (2016) propose a neighborhood that includes all the DGs that can be obtained by considering the addition of every subset of edges in the neighborhood of the minimal separators.

Contributions In this work we propose a divide-and-conquer approach to Problem 1 (see Section 3), called iterative coarsening algorithm (IC). This approach consists of generating a sequence of coarser DGs $G_1 \prec ... \prec G_K$, where G_1 is the empty graph. Each coarsening step is performed using an adaptation of the Integer Linear Programming (ILP) formulation presented in (Pérez et al., 2014). IC is a local search algorithm that adds the optimal set of edges due to the minimal separators of a given DG. It includes a parameter that allows controlling the trade-off between the quality of the obtained solution and the computational resources required for learning DMs. IC has obtained competitive results learning DMs compared with exact and approximated approaches (see Section 4). Finally, we adapt IC to reduce the search space of exact procedures (see Section 5). The adaptation has obtained promising results in terms of the percentage of edges recovered from the optimal solution to the MWDG problem.

2. Notation and main concepts

Let $G = (V, E)$ be an undirected graph, where $V = \{1, ..., n\}$ is the set of vertices and E is a set of pairs of vertices $\{u, v\}$ called edges. We say that $G^+ = (V^+, E^+)$ is *coarser* than G (or equivalently, G is thinner than G^+) when $V = V^+$ and $E \subsetneq E^+$. The coarser and thinner terms are used to induce a partial ordering among the graphs, $G \prec G^+$. An undirected graph is called a decomposable graph[1] (DG) if for any cycle of length greater than 3 there exists a chord. Henceforth, we deal with DGs only.

A set of vertices is said to be complete in G when it induces a complete subgraph. A maximal complete set C is called clique, and the set of all the cliques of G is denoted by $\mathbb{C}$. The neighborhood of u in G is the set of vertices connected by an edge to u, $\mathbb{N}_u = \{v \in V : \{u, v\} \in E\}$. We define the neighborhood of a set of vertices S in G as the set of vertices connected to all the vertices in S, $\mathbb{N}_S = \{v \in V : \forall u \in S, \{u, v\} \in E\} = \bigcap_{u \in S} \mathbb{N}_u$. Note that this definition of neighborhood is the intersection of the neighborhood of each vertex instead of the union. The set of (minimal) separators of G is denoted by $\mathbb{S}$. The set of minimal separators between u and v is denoted by $\mathbb{S}_{u,v}$. The set of connected components that is obtained by the removal of a separator S from G is denoted by $\mathbb{V}_S$. The set of substantial components of a separator S correspond to $\{R \in \mathbb{V}_S : \mathbb{N}_S \cap R \neq \emptyset\}$, and its size is called the degree of the separator S, d_S. The degree of a separator corresponds to one plus its multiplicity.

Any DG coarser than a given DG, G, can be obtained by sequential addition of edges that maintains the decomposability of all the intermediate coarser graphs (Lauritzen, 1996). We call potential edge an edge whose addition maintains the chordality of a DG. An edge $\{u, v\}$ is a potential edge iff there exists $S \subseteq V \backslash \{u, v\}$ such that (1) S is a separator for u and v in G, $S \in \mathbb{S}_{u,v}$, and (2) u and v are both completely connected to S, $\{u, v\} \subseteq \mathbb{N}_S$. In this case, we say that $\{u, v\}$ can be added due to S. Clearly, the addition of a potential edge $\{u, v\}$ due to S creates the clique $\{u, v\} \cup S$. Therefore, the obtained coarser DG has the additional complete subsets $\{\{u, v\} \cup R : R \subseteq S\}$, and thus we can associate the next weight to the addition of the potential edge $\{u, v\}$ due to S:

$$w(u, v|S) = \sum_{R \subseteq S} w(\{u, v\} \cup R). \tag{3}$$

1. Chordal graph, triangulated graph.

Table 1: Main concepts

$G \prec G^+$	G^+ is coarser than G	$\mathbb{C}$	Cliques of G
$\mathbb{N}_u$	Neighborhood of u	$\mathbb{N}_S$	Neighborhood of S
$\mathbb{S}$	Minimal separators of G	$\mathbb{S}_{u,v}$	Minimal separators for $\{u, v\}$
$\mathbb{V}_S$	Connected comps. separated by S	d_S	Degree of S
$\mathbb{E}_S$	Candidate edges due to S	$w_{u,v\|S}$	Weight of $\{u, v\}$ due to S

Finally, we call $\mathbb{E}_S = \{\{u, v\} : S \in \mathbb{S}_{u,v}\}$ the set of candidate edges due to S in G, and when $\{u, v\} \in \mathbb{E}_S$ we say that $\{u, v\}$ is a candidate edge due to S. We define the length of a candidate edge $\{u, v\} \notin E$ as the number of minimal separators for $\{u, v\}$, $l_{u,v} = |\mathbb{S}_{u,v}|$. The set of candidate edges due to the separators of a DG will be the basis for the coarsening step of the proposed learning algorithm: a subset of candidate edges will be added at each iteration of the algorithm. The selected candidate edges have to guarantee that, at least, there exists an order of addition for which they become potential edges.

The notation introduced in this section is summarized in Table 1.

3. Learning by coarsening

In this section, we propose an algorithm to deal with Problem 1. The algorithm follows the general heuristic proposed in (Pérez et al., 2016) that consists of creating a sequence of coarser DGs, $G_1 \prec G_2 \prec ... \prec G_K$, where G_1 and G_K are the empty graph and the obtained DG, respectively.

In the previous section, we have indicated that any DG can be learned from a thinner DG by adding potential edges iteratively. In this work, we propose to perform a coarsening step, which consists of learning G_{k+1} from G_k, by adding iteratively potential edges from the set of candidate edges due to the separators of G_k, for $k = 1, ..., K - 1$. Each coarsening step increases the maximum clique size at most in one. The next problem formalizes the coarsening steps of the algorithm:

Problem 2 (The coarsening problem) *Given a DG, G, and a decomposable score, w (see Eq. 2), find the maximum weighted coarser DG G^+ that can be obtained by adding a subset of the candidate edges $\mathbb{E}_S$ due to S for all $S \in \mathbb{S}$.*

This problem can be solved by finding the sequence of the addition of the candidate edges in which they are chordal. That is, before adding $\{u, v\}$ due to S we have to ensure that $S \in \mathbb{S}_{u,v}$ and $\{u, v\} \in \mathbb{N}_S$, by adding other candidate edges when it is required. The proposed procedure can be seen as a local search that finds the best neighbor in a huge neighborhood by solving Problem 2.

The coarsening problem can be solved by using Integer Linear Programming (Pérez et al., 2014). Given a DG, G, the set of decision variables in the ILP formulation of the coarsening problem corresponds to the candidate edges due to the separators of G, $\mathcal{X} = \{X_{u,v|S} : S \in \mathbb{S}, \{u, v\} \in \mathbb{E}_S\}$, where $X_{u,v|S} \in \{0, 1\}$, and $X_{u,v|S} = 1$ represents that the edge $\{u, v\}$ is added to G due to the separator S.

Given a DG $G = (V, E)$, a decomposable score function w (see Equation 2 and Equation 3) and the set of decision variables $\mathcal{X}$, the ILP formulation of Problem 2 is given by

$$\max \sum_{S \in \mathbb{S}} \sum_{\{u,v\} \in \mathbb{E}_S} w(u, v|S) \cdot X_{u,v|S},$$

subject to the constraints:

1) For $\{u, v\} \notin E$:
$\sum_{S \in \mathbb{S}_{u,v}} X_{u,v|S} \leq 1$

2) For $S \in \mathbb{S}$ and $\{u, v\} \in \mathbb{E}_S$:
$[\sum_{s \in S} \sum_{R \in \mathbb{S}_{u,s}} X_{u,s|R}] - |S| \cdot X_{u,v|S} + \sum_{s \in S} 1_{u,s} \geq 0$, and
$[\sum_{s \in S} \sum_{R \in \mathbb{S}_{v,s}} X_{v,s|R}] - |S| \cdot X_{u,v|S} + \sum_{s \in S} 1_{v,s} \geq 0$,
where $1_{u,v}$ is 1 if $\{u, v\} \in E$, and 0 otherwise.

3) For $S \in \mathbb{S}$ and $\mathcal{V} \subseteq \mathbb{V}_S$:
$\sum_{T \in \mathcal{V}} \sum_{U \in \mathcal{V} \setminus \{T\}} \sum_{u,v \in T,U} X_{u,v|S} \leq |\mathcal{V}| - 1$

Constraints (1) guarantee that the edges are added at most due to a single separator; constraints (2) guarantee that to allow the addition of $\{u, v\}$ due to S both vertices are in the neighborhood of S, $\{u, v\} \subseteq \mathbb{N}_S$; and constraints (3) ensures that the addition of a set of edges due to a separator S can not form cycles among the set of connected components separated by S, $\mathbb{V}_S$, and thus that S is the minimal separator for all the added edges. In summary, for every set of candidate edges satisfying constraints (1), (2) and (3) there is at least an order of addition for which they become potential, and thus the obtained graph is a DGs.

We can control the number of variables and the number (and the size) of constraints of the ILP formulation proposed to solve Problem 2 by considering only the subset of edges with a maximum length of l. By constraining the maximum length of the edges considered to tackle Problem 2 we are discarding the edges that have a lower prior probability of being included in the optimal solution (Pérez et al., 2018). Selecting an appropriate value of l, we can effectively control the trade-off between the required computational resources for solving Problem 1 and the quality of the obtained solution.

At this point, we would like to highlight that some of the edges considered in Problem 2 can not be added due to the separators in $\mathbb{S}$, and thus the ILP formulation can be reduced drastically by removing them. For example, given the DG G with $\mathbb{C} = \{\{1, 2\}, \{2, 3, 4\}, \{3, 4, 5\}\}$ we can not add $\{1, 5\}$ due to $\{3, 4\}$ because we have to add both $\{1, 3\}$ and $\{1, 4\}$, and we have a single separator $\{2\}$ to do it. A subset of the non-eligible edges can be efficiently characterized as follows: $\{u, v\}$ due to S can not be part of the solution to Problem 2 if exists a pair of vertices $\{r, s\} \subseteq S$ not connected to u for which $\mathbb{S}_{u,r} = \mathbb{S}_{u,s}$. The rest of non-eligible edges can be determined by inspecting when the constraints (2) for the addition of the edge $\{u, v\}$ due to S can not be fulfilled. From here on, we will consider that $\mathbb{E}_S$ represents the set of candidate edges due to S that can not form part of the solution to Problem 2.

3.1 The iterative coarsening algorithm

In this section, we propose an algorithm for learning DMs using decomposable scores such as BIC and BDeu. The algorithm is an approximated approach to Problem 1 that progresses by solving Problem 2 iteratively. The pseudo-code of the iterative coarsening (IC) algorithm is shown in Algorithm 1.

IC starts by learning the maximum weighted forest (MWF) over n vertices using Prim's algorithm, which solves Problem 2 given the empty graph with a computational complexity of $\mathcal{O}(n^2)$. Next, the algorithm uses the ILP formulation for solving the coarsening problem ($LearnCoarser$ procedure) iteratively until convergence. The set of weights W indicates to $LearnCoarser$ the edges that can be considered to deal with Problem 2.

The algorithm has two optional parameters that can be used to control its running time: K is a positive integer that bounds the maximum clique size of the obtained DGs, and it can be used to explicitly control the complexity of the associated decomposable models (by default $K = \infty$); l is a positive integer that bounds the maximum length of the candidate edges considered at each coarsening step, and thus it controls the search space of the coarsening steps (by default $l = \infty$).

Algorithm 1: The iterative coarsening (IC) algorithm

```
Input: A decomposable scoring function $w$ (see Eq. 2), the number of vertices $n$.
Optionally: The maximum clique size, $K = \infty$, the maximum length of the candidate
  edges, $l = \infty$.
Output: A DG, $G^+$.
$G = (V, E)$ with $E = \emptyset$
$G^+ = (V, E^+)$ with $E^+ = MWF(w, n)$
while $G \neq G^+$ do
    $G = G^+$
    $W = \{w_{u,v|S} : \{u, v\} \in \mathbb{E}_S(G), S \in \mathbb{S}(G), l_{u,v} \leq l, |S| < K - 2\}$
    $G^+ = LearnCoarser(G, W)$
return $G^+$
```

4. Experiments

In this section, we present three types of experiments that illustrate the effectiveness of IC for learning DMs using the BDeu score (Heckerman et al., 1995). First, we deal with Problem 1 using data sets of small dimensionality, and we compare the obtained solutions with GOBNILP (Studený and Cussens, 2016, 2017), the maximum spanning tree (MST) and the local search algorithm that considers the addition of the best chordal edge at each step (Greedy) (Deshpande et al., 2001). We have selected Greedy because it has polynomial complexity in n and K, and its one of the most popular algorithms for learning DGs. Then, we deal with Problem 1 using high-dimensional data sets and we compare the obtained solutions with MST and Greedy.

In the experiments, we have solved the ILP formulations of Problem 2 using Gurobi® as it allows us to solve ILPs in high-dimensional problems. This is also the same solver used by GOBNILP. As our approach requires solving a sequence of ILPs, we set the `MIPFocus = 1`. This causes Gurobi® to focus on locating feasible solutions quickly. Additionally, we set the time limit for each ILP to be 500 seconds. The implementation of Greedy is taken from the python library `pgmpy`, which follows the design given in (Koller and Friedman, 2009, Section 18.4.3) with a slight modification: the allowed edges are chordal. The IC algorithm has been implemented in Python and the source code is available at `https://github.com/georgeAO/IterativeCoarsening/`.

All experiments are conducted on the DIPC Atlas cluster which contains Intel Xeon® E5-2680/2863, Xeon Gold® 6140, and Xeon Platinum® processors. Individual problems are limited to using at most 12 cores and 128 GB of RAM, however, most experiments are completed using less that 64 GB.

4.1 Low-dimensional domains

In this section, we consider some standard data sets which are typically used in learning probabilistic graphical models. In total, ten data sets from the URLearning Bayesian network data repository[2] are used for learning DM. Scores will be compared from MST, Greedy, IC, and GOBNILP.

Following the default settings of GOBNILP, we set the maximal clique size of each approach to be four and use a BDeu scoring function with an equivalent sample size of one, $\alpha = 1$. Additionally, we change the default settings of GOBNILP to `chordal = True` and `pruning = False`.

2. `http://urlearning.org/datasets.html`

Table 2: This table shows the obtained results using 10 low-dimensional data sets. The column *Data Set* shows the names of the data sets used in the experiment, n represents the number of variables of each data set and MST shows the scores of MST. The columns w represent the increment in the scores obtained with Greedy, IC, and GOBNILP with respect to MST. The columns t show the computation time (in seconds) for Greedy, IC, and GOBNILP. All the values have been rounded to the closest integer.

			Greedy		IC		GOBNILP	
Data Set	n	MST	w	t	w	t	w	t
Wine	14	-1274	13	3	13	0	13	8
Voting	17	-4654	5	4	5	1	11	13
Hepatitis	20	-1343	1	1	4	1	4	22
Heart	23	-2455	32	3	32	2	32	52
Autos	26	-1711	42	13	97	2	138	11072
Horse	28	-4588	25	8	27	3	28	183
Flag	29	-2811	19	13	24	3	34	926
Water1000	32	-13322	29	19	35	12	35	497
Alarm	37	-14638	171	29	200	16	200	1788
Bands	39	-5252	29	19	47	10	70	4669

This allows GOBNILP to search for the optimum MWDG with a maximum clique size of four. The maximum length of the allowed edges for IC, l, is unbounded.

The summary of the results obtained is shown in Table 2. The table shows the BDeu score for MST (MST column), and for Greedy, IC, and GOBNILP the increment in the score with respect to MST (w columns). The running times for Greedy, IC, and GOBNILP are reported in their corresponding t columns. IC and Greedy obtain the optimal DG in Wine, Hepatitis, and Heart datasets (three of the four smallest data sets). In the rest of the data sets, IC consistently outperforms Greedy in terms of both the score and the running time. The highest improvements of IC with respect to Greedy in the BDeu score are obtained in Autos (55), Alarm (29), and Bands (28), where the numbers in parenthesis represent the difference in the obtained BDeu scores. IC learns DGs very close to the optimal (GOBNILP) in 7 out of 10 datasets: Wine (0), Hepatitis (0), Voting (6), Heart (0), Horse (1), Flag (10), Watter100 (<1) and Alarm (<1).

4.2 High-dimensional domains

The following experiment considers three high-dimensional data sets, namely: *Semeion Handwritten Digit Data Set* (SHD, $n = 256$ variables, 1593 instances), *QSAR Androgen Receptor* (QAR, $n = 1024$ variables, 1687 instances) and *QSAR Oral Toxicity* (QOT, $n = 1024$ variables, 8992 instances). These high-dimensional data sets would not be approachable with exact methods for Problem 1, and we will analyze the scalability of IC compared to MST and Greedy. For the IC algorithm, we restrict the search space by changing the maximum edge length value, $l \in \{4, 8, 12\}$. Unlike the low-dimensional experiment, no upper bound on the clique size is enforced. The maximum permitted running time for each algorithm in each data set is one hour. Hence, we only report results from experiments which could be completed within this time. Again, we score DGs using the BDeu score with an equivalent sample size of $\alpha = 1$.

The summary of the result obtained in SHD is shown in Table 3, which follows the same format as Table 2. IC outperforms Greedy in terms of the BDeu score for all the values of l considered in the experiment. The difference in the score increases as l increases. We would like

Table 3: This table summarizes the results obtained with the SHD data set. The organization of the table is equivalent to Table 2. The column l shows the maximum length of the edges allowed for IC.

		Greedy		IC		
n	MST	w	t	l	w	t
				4	20466	167
256	-156383	979	3601	8	23351	1389
				12	23953	2934

Table 4: This table summarizes the results obtained with the QAR and QOT data sets. The organization of the table is equivalent to Table 2. The column l shows the maximum length of the edges allowed for IC.

			Greedy				IC				
n	MST		w		t		l	w		t	
							4	208	2504	14	174
50	-17784	-94297	190	2092	114	557	8	221	2647	48	520
							12	222	2693	82	898
							4	761	7364	58	373
100	-34101	-176914	647	5510	3158	3600	8	808	7959	216	1790
							12	806	-	551	-
							4	2113	18600	186	1162
200	-64673	-337545	561	3185	3601	3601	8	2242	-	1026	-
							12	2252	-	2758	-
							4	5423	46997	401	2155
400	-122350	-640162	218	1228	3602	3603	8	5859	-	2653	-

to highlight that the difference in the score has increased several orders of magnitude with respect to low dimensional data sets, which motivates the next experiment.

In order to study the scalability of IC with respect to the number of variables n, for the QAR and QOT data sets, we have selected uniformly at random $n \in \{50, 100, 200, 400\}$ variables. For each n value, we have repeated the experiment 5 times.

The summary of the result obtained in QAR and QOT is shown in Table 4, which follows the same format as Table 2. The scores and running times have been averaged over the 5 runs of the experiment. IC consistently outperforms Greedy in terms of the score for different n and l values. As n increases the differences in the improvements with respect to MST among Greedy and IC increases being more than 25 and 40 times bigger for $n = 400$ with QAR and QOT, respectively. This strong empirical evidence suggests that the use of IC instead of Greedy is advisable for learning DMs in high dimensional domains.

5. Reducing the search space

In this section, by using IC, we propose a simple procedure for reducing the search space for exact methods to tackle Problem 1. Then, we empirically evaluate the proposal comparing the found structures with the optimal solution to the problem (GUROBI).

The proposed algorithm is based on a slight modification of the ILP formulation introduced in Section 3 by imposing equality conditions over the next subset of constraints (3):

$$\text{For } S \in \mathbb{S}: \qquad \sum_{T \neq U \in \mathbb{V}_S} \sum_{u,v \in T,U} X_{u,v|S} = d_S - 1$$

By using these equality constraints in ILP formulation we are forcing to add $d_S - 1$ edges due to separator S, for $S \in \mathbb{S}$. Intuitively, we are constructing a maximum weighted tree between the substantial components of S, for $S \in \mathbb{S}$. Thus, we are forcing to add a fixed number of candidate edges to the DG due to the current set of separators $\mathbb{S}$, even when the score of the obtained DG decreases. It is important to note, that the set of equality constraints can be satisfied for any DG, even when we only allow the addition of edges of length $l = 1$ because every substantial component of S has at least one vertex in the neighborhood of S, for $Sin\mathbb{S}$. In the particular case of $l = 1$, we can obtain the solution to the problem by learning a maximum weighted tree in the neighborhood of S, $\mathbb{N}_S$, using the weights $\{w(u,v|S) : l_{u,v} = 1, \{u,v\} \in \mathbb{E}_S\}$, for $S \in \mathbb{S}$ (Pérez et al., 2016). This solution can be obtained with a computational complexity of $\mathcal{O}(\sum_{S \in \mathbb{S}} d_S^2)$.

Algorithm 2: The forced iterative coarsening (FIC) algorithm

```
Input: An oracle of a decomposable scoring function $w$ (see Eq. 2), the number of vertices
  $n$, and the number of iterations with the maximal coarsening process, $K^+$.
Optionally: The maximum clique size, $K = \infty$, the maximum lengths of the candidate
  edges $l = \infty$, $l^+ = \infty$
Output: A DG, $G^+$, where $G^+$ has at least a maximum clique size of $\min\{K^+, K\}$.
$G = (V, E)$ with $E = \emptyset$
$G^+ = LC(w, n; K, l)$
for $k = 1, ..., K^+$ do
    $G = G^+$
    $W = \{w_{u,v|S} : \{u,v\} \in \mathbb{E}_S(G), S \in \mathbb{S}(G), l_{u,v} \leq l^+, |S| < K - 2\}$
    $G^+ = LearnMaximalCoarser(G, W)$
return $G^+$
```

The pseudo-code of the procedure proposed for reducing the search space of exact methods (forced iterative coarsening, FIC) is shown in Algorithm 2. FIC starts by learning a DG utilizing IC, and then it forces the addition of a maximal set of edges by using the ILP formulation with equality constraints ($LearnMaximalCoarser$) during K^+ iterations. In the pseudo-code, the set of weights W implicitly indicates to $LearnMaximalCoarser$ the set of candidate edges considered in Problem 2, which depends on the maximum clique size value K and the maximum length of the edges l^+.

To analyze the behavior of FIC for constraining the search space of exact procedures we compare the obtained DGs with the optimal solutions found by GOBNILP. For this purpose, we have used the low-dimensional data sets of the experiments in Section 4.1 with the same experimental conditions. The experiments have been performed with different number of iterations for the maximal coarsening process ($LearnMaximalCoarser$), $K^+ \in \{0, 1, 2\}$, where $K^+ = 0$ represents the DG obtained with IC. The maximum length of allowed edges in the maximal coarsening is $l^+ = 1$.

The summary of the results is shown in Table 5. The columns associated to MST, Greedy and FIC for $K^+ = 0, 1, 2$ show the percentage of edges recovered from the optimal solution (V, E^*) (i.e., the sensitivity, $|E^* \cap E|/|E^*|$), and the percentage of edges that are not included in the optimal solution (i.e., the false discovery rate $|E \setminus E^*|/|E|$) separated by $-$. As K^+ increases,

Table 5: This table shows the sensitivity and the false discovery rate of the edges found (separated by –) for MST, Greedy and FIC compared to the optimal DGs (GUROBI).

				FIC, K^+		
Data Set	n	MST	Greedy	0	1	2
Wine	14	81 - 0	100 - 0	100 - 0	100 - 36	100 - 43
Voting	17	52 - 19	64 - 16	64 - 16	80 - 39	80 - 39
Hepatitis	20	83 - 0	87 - 0	100 - 0	100 - 43	100 - 44
Heart	23	82 - 0	100 - 0	100 - 0	100 - 29	100 - 31
Autos	26	35 - 16	45 - 36	75 - 25	85 - 37	85 - 37
Horse	28	72 - 4	89 - 6	100 - 3	100 - 39	100 - 41
Flag	29	49 - 11	65 - 13	80 - 13	92 - 33	92 - 33
Water1000	32	65 - 10	84 - 16	88 - 16	91 - 46	93 - 48
Alarm	37	66 - 3	93 - 6	100 - 2	100 - 35	100 - 36
Bands	39	44 - 24	52 - 28	65 - 20	80 - 38	80 - 40

the sensitivity of FIC increases. The highest increase in the sensitivity is obtained from $K^+ = 0$ to $K^+ = 1$ for all the data sets. For $K^+ = 1$, it is 90% in 8 out of 10 data sets, and the lowest value is 80%. A remarkable property of IC ($K^+ = 0$) is its low false discovery rate: lower than 5% in 5 out of 10 data sets and between 10% and 20% in 4 out of 10. It is possible to obtain better results considering higher values for l^+ at the expense of using more computational resources. The obtained results suggest that FIC can be used for reducing the search space of the exact algorithm to tackle the problem of learning DMs in high-dimensional domains.

6. Conclusions

This work proposes an efficient algorithm called iterative coarsening (IC) for learning the structure of decomposable models through the maximization of a given decomposable score, i.e. the maximum weighted decomposable graph problem (MWDG). The algorithm is a local search algorithm with a huge neighborhood from which the optimal decomposable graph (DG) is obtained by using Integer Linear Programming (ILP). IC constructs a sequence of coarser DGs iteratively. At each coarsening step, given a DG, the algorithm identifies the set of edges that can be added given the current minimal separators, and finds the optimal subset. The algorithm allows to control the trade off between the quality of the obtained solution and the amount of computational resources required, and to bound the maximum clique size of the obtained DG.

IC has shown very competitive results for learning decomposable models compared to GOBNILP (Studený and Cussens, 2016, 2017), an exact procedure for solving the MWDG problem. IC outperforms the popular greedy heuristic proposed in (Deshpande et al., 2001), especially in high-dimensional domains for which the application of GOBNILP is unfeasible. Finally, based on a slight modification of the ILP formulation used in the coarsening steps of IC, we have proposed a suitable approach for reducing the search space of exact algorithms in high-dimensional domains.

Acknowledgments.

We would like to thank the technical help of James Cussens in the use of GOBNILP for learning decomposable graphs with a bounded clique size.

This work has been partially supported through the BERC 2018-2021 program and the ELKA-RTEK program (Basque Government), the BCAM Severo Ochoa excellence accreditation SEV-2017-0718 (Spanish Ministry of Science, Innovation and Universities - AEI) and the project TIN2017-82626-R (Spanish Ministry of Economy and Competitiveness).

References

E. Aarts, E. H. Aarts, and J. K. Lenstra. *Local search in combinatorial optimization*. Princeton University Press, 2003.

A. Chechetka and C. Guestrin. Efficient principled learning of thin junction trees. In *Advances in Neural Information Processing Systems*, pages 273–280, 2008.

J. Corander, T. Janhunen, J. Rintanen, H. Nyman, and J. Pensar. Learning chordal Markov networks by constraint satisfaction. In *Advances in Neural Information Processing Systems*, pages 1349–1357, 2013.

S. Dasgupta. Learning polytrees. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 134–141, 1997.

L. M. de Campos and J. F. Huete. Algorithms for learning decomposable models and chordal graphs. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, 1997.

A. Deshpande, M. Garofalakis, and M. I. Jordan. Efficient stepwise selection in decomposable models. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 128–135, 2001.

J. Eisner. State-of-the-art algorithms for minimum spanning trees – a tutorial discussion. Research report, University of Pennsylvania, 1997.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

R. Hemmecke, S. Lindner, and M. Studenỳ. Characteristic imsets for learning Bayesian network structure. *International Journal of Approximate Reasoning*, 53(9):1336–1349, 2012.

T. Janhunen, M. Gebser, J. Rintanen, H. Nyman, J. Pensar, and J. Corander. Learning discrete decomposable graphical models via constraint optimization. *Statistics and Computing*, 27(1):115–130, 2017.

K. Kangas, T. Niinimäki, and M. Koivisto. Learning chordal Markov networks by dynamic programming. In *Advances in Neural Information Processing Systems*, pages 2357–2365, 2014.

D. R. Karger and N. Srebro. Learning markov networks: maximum bounded tree-width graphs. In *SODA*, pages 392–401, 2001.

D. Koller and N. Friedman. *Probabilistic Graphical Models. Principles and Techniques*. The MIT Press, Cambridge, Massachusetts, 2009.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, New York, 1996.

F. M. Malvestuto. Approximating discrete probability distributions with decomposable models. *IEEE Transactions on Systems, Man and Cybernetics*, 21(5):1287–1294, 1991.

F. M. Malvestuto. A backward selection procedure for approximating a discrete probability distribution by decomposable models. *Kybernetika*, 48(5):825–844, 2012.

A. Pérez, C. Blum, and J. A. Lozano. Learning maximum weighted (k+1)-order decomposable graphs by Integer Linear Programming. In *Proceedings of the Seventh European Workshop on Probabilistic Graphical Models*, volume 8754, pages 396–408. Springer, 2014.

A. Pérez, I. Inza, and J. A. Lozano. Efficient approximation of probability distributions with k-order decomposable models. *Journal of Approximate Reasoning*, 74:58–97, 2016.

A. Pérez, C. Blum, and J. A. Lozano. Approximating the maximum weighted decomposable graph problem with applications to probabilistic graphical models. In *Proceedings of the Ninth International Conference on Probabilistic Graphical Models*, pages 320–331, 2018.

K. Rantanen, A. Hyttinen, and M. Järvisalo. Learning chordal markov networks via branch and bound. In *Advances in neural information processing systems*, pages 1847–1857, 2017.

G. Schwarz. Estimation of the dimension of a model. *Annals of Statistics*, 6:416–446, 1978.

P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman. *Causation, prediction, and search*. MIT press, 2000.

N. Srebro. Maximum likelihood Markov networks: An algorithmic approach. Master thesis, Massachuset Institute of Technology, 2000.

M. Studený and J. Cussens. The chordal graph polytope for learning decomposable models. In *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*, pages 499–510, 2016.

M. Studený and J. Cussens. Towards using the chordal graph polytope in learning decomposable models. *International Journal of Approximate Reasoning*, 88:259–281, 2017.