An Approximate Formulation based on Integer Linear Programming for Learning Maximum Weighted (k+1)-order Decomposable Graphs

Aritz Pérez
Basque Center for Appied Mathematics (BCAM)
Bilbao, Spain
aperez@bcamath.org

Christian Blum
Dep. of Computer Science and Artificial Intelligence
University of the Basque Country, UPV/EHU
San Sebastian, Spain
and IKERBASQUE, Basque Foundation for Science
Bilbao, Spain
christian.blum@ehu.es

Jose A. Lozano
Dep. of Computer Science and Artificial Intelligence
University of the Basque Country, UPV/EHU
San Sebastian, Spain
ja.lozano@ehu.es

Abstract

In this work we deal with the problem of learning a maximum weighted $(k+1)$-order decomposable graph coarser than a given maximal k-order decomposable graph. An Integer Linear Programming formulation for the problem has been recently proposed and used in order to solve instances of the problem with a moderate number of vertices [8]. However, as the problem is known to be NP-hard, it is of practical interest to develop approximate algorithms able to work with a limited amount of computational resources. In this paper we propose an approximate Integer Linear Programming formulation for the problem using a distance-based criteria that selects a subset of edges. This criteria discards the set of edges that, on average, have lower probability to be part of the optimal solution than any of the selected edges. Experiments have been carried out with probabilistic graphical models. Using the approximate formulation we have obtained results close to the optimum, even with small threshold distance values. The obtained good results indicate that the approximate formulation has important applications for learning probabilistic graphical models using decomposable scores.

1 Introduction

We denote by k**DG** to a decomposable[1] graph with a maximum clique size of k. The problem of learning a maximum weighted kDG (**Problem** k**DG**) is known to be an NP-hard combinatorial

[1] Decomposable graphs are also known as triangulated or chordal

problem [9, 10]. kDGs are also known in the literature as hyperforests of treewidth $k-1$ [9]. Problem kDG is of practical interest for learning probabilistic graphical models with a bounded treewidth using decomposable scores [5]. Recently, two exact algorithms with exponential computational complexity have been proposed in order to deal with Problem kDG [1, 4]. However, given a reasonable amount of computational resources, the proposed algorithms can only learn maximum weighted kDGs with a low number of vertices, e.g, $n < 30$.

In order to learn graphs with higher number of variables several approximate methods have been proposed in the literature, among which we highlight the approach presented in [8]. In this work the authors presented a problem related to Problem kDG: Given a maximal kDG (**MkDG**) to learn a coarser maximum weighted $(k+1)$DG (**Problem $(k+1)$DG**) [8]. MkDGs are also known in the literature as hypertrees of treewidth $k-1$ [9]. In other words, Problem $(k+1)$DG consists of constructing a maximum weighted $(k+1)$DG by adding edges to an input MkDG. Due to Lemma 2.21 in [6], an algorithm that solves this problem can be used in order to approach Problem kDG by constructing a sequence of coarser iDGs for $i = 2, ..., k$. Unfortunately, Problem $(k+1)$DG is known to be NP-hard for $k > 2$ [8]. In [8], the authors propose an Integer Linear Programming (ILP) formulation of Problem $(k+1)$DG (exact formulation, **EF**). This formulation takes advantage of the theoretical properties of MkDGs [7, 8].

EF is based on the notion of **candidate edges** [2], i.e. those which addition maintains the decomposability of the graph. The formulation makes use of the characterization of candidate edges proposed in [2] (see Theorem 2.2), which is focused on the (minimal) **separators** of the graph. In order to solve Problem $(k+1)$DG, EF identifies all the candidate edges that can be added to any $(k+1)$DG coarser than the given MkDG which create cliques of size $k+1$. The authors call to this subset of candidate edges the edges of interest (**EOI**). A decision variable of EF is associated to the presence (or absence) in the solution of an EOI $\{u, v\}$ due to one of its separator S of size $k-1$ in the input MkDG. It should be noted that the addition of $\{u, v\}$ due to S creates the clique $\{u, v\} \sup S$ of size $k+1$ in the solution. This leads to $\mathcal{O}(n^3)$ binary decision variables in the worst case. The formulation includes three types of constraints in order to guarantee that a $(k+1)$DG is constructed. The **type I** constraints guarantee that each edge is added at most due to one separator. The number of type I constraints is $\mathcal{O}(n^2)$ in the worst case. The **type II** constraints control that an edge $\{u, v\}$ can be added due to a separator S only if u and v are in the common neighborhood of S. In other words, type II constraints guarantee that, if the EOI $\{u, v\}$ due to S is included in the solution to Problem $(k+1)$DG, then all the edges $\{u, w\}$ and $\{v, w\}$ for $w \in S$ are also included. The number of type II constraints is $\mathcal{O}(n^3)$ in the worst case. Finally, **type III** constraints guarantee that the added EOIs due to S do not form a cycle. The number of this constraints can be exponential, however, as it was pointed out in [8] on average they tend to grow linearly with the number of vertices n of the input MkDG.

The exact formulation can be understood as a treewidth friendly operator [3] that, at most, increases the treewidth of the learned graph in one. The experimental results provided in [8] suggest that EF can effectively solve instances of Problem $(k+1)$DG with $n = 75$ vertices given a reasonable amount of computational resources. However, in practice, EF is unable to solve instances with larger graphs due to the high number of decision variables and type II constraints. In this work we propose an approximate formulation to Problem $(k+1)$DG (**AF**). AF selects a subset of EOIs using an intuitive distance-based criterion. With this distance-based constraint we empirically show that it is possible to learn approximate solutions to instances of Problem $(k+1)$DG with a reasonable quality.

2 Approximate formulation

The experiments carried out in [8] suggest that EF is unable to solve instances of Problem $(k+1)$DG with $n = 100$ vertices given a reasonable amount of computational resources. AF tries to avoid this shortcoming by discarding the set of EOIs that, on average, have a lower probability of being part of the optimal solution to instances of Problem $(k+1)$DG.

We define the distance between the vertices u and v, $d(u, v)$, as the length of the shortest path from u to v in the MkDG. We define the distance between the vertex u and the separator S as the distance between u and the farthest vertex of S in the MkDG, $d(u, S) = max_{w \in S} d(u, w)$. We say that an **edge** $\{u, v\}$ **has length** l when $d(u, v) = l$.

In this work we propose to consider only the addition of an EOI if its length is lower or equal to a maximum length l_{max}. We would highlight that we can always create solutions to Problem $(k+1)$DG using only the subset of EOIs with a maximum length of l_{max}. Moreover, any of the selected EOIs can be part of a solution to the problem. In this sense, we can say that the subset of candidate edges determined by a maximum length of l_{max} is appropriate for approximating Problem $(k+1)$DG. Next we show that the proposed criterion can control the number of decision variables and type II constraints of the ILP formulation of Problem $(k+1)$DG. Besides, we argue that this criterion discards the EOIs that have, on average, lower probability than any of the selecte edges of being part of the optimal solution to instances of Problem $(k+1)$DG.

In practice, as it was indicated in [8], the bottleneck of EF is related with the number of decision variables and the number of type II constraints. As noted before, in EF the decision variables and type II constraints are $\mathcal{O}(n^3)$ in the worst case. The worst case corresponds to an MkDG with a chain structure. In this situation AF considers a subset of $\mathcal{O}(n \cdot l_{max}^2)$ and, therefore, the number of decision variables and type II constraints grows linearly with the number of vertices n of the input MkDG. The worst case for AF correspond to an MkDG with l_{max} separators. In this case AF has a number of decision variables and type II constraints of $\mathcal{O}(n^2 \cdot l_{max})$, which grows quadratically with the number of vertices. Note that in this case AF is equivalent to EF. Therefore, we can conclude that using AF it is possible to control both the number of decision variables and the number of type II constraints by selecting an appropriate value of l_{max}.

By Theorem 2.2 in [2], we know that an EOI $\{u, v\}$ due to S can be form part from the solution to Problem $(K+1)$DG only if u and v are in the common neighborhood of S. As a consequence, in order to add the EOI $\{u, v\}$ due to S at least $(d(u, S)-1)\cdot(k-1)$ edges of length lower than $d(u, S)$ and $(d(v, S)-1)\cdot(k-1)$ edges of length lower than $d(v, S)$ have to be added to the input MkDG to create a solution to Problem $(k+1)$DG. We can extract two conclusions from this observation. First, as the length of an EOI increases its associated type II constraint is harder to fulfill. And second, the number of edges of length l in the solution tends to decrease on average as l increases. If we assume that, on average, the weight function has no preference for any EOI, then as the length of the EOI increases its probability of being included in the solution decreases. In this sense, we say that, on average, a discarted edge has lower probability to be part of the optimal solution than any of the selected EOIs.

3 Experiments learning decomposable models

This section includes a set of experimental results that illustrate the effectiveness of AF to deal with Problem $(k+1)$DG even when a small value of l_{max} is selected. The experiments are focused on learning decomposable models using, as the weighting function, the Bayesian Dirichlet equivalent uniform score [5]. Without loss of generality, the experiments are focused on learning 3DGs from M2DGs (Problem $(k+1)$DG for $k=2$). We would highlight that the same conclusion should be obtained for other values of k. Further experimentation is left to future work. The experiments have been designed to study the relation between the values of l_{max}, the size of the AF and the trade-off between the quality of the learned graph and the spent computational resources.

The probability distributions underlying the sampled data have been generated as follows : (1) A M2DG $\boldsymbol{G}$ with n vertices is generated at random using the Kruskal algorithm for maximization where the weights associated with the edges are randomly generated. (2) Take a root vertex from $\boldsymbol{G}$ at random and transform $\boldsymbol{G}$ into a DAG. (3) Add arcs at random to the DAG until all the vertices (except the root) have two parents to form $\boldsymbol{G}^+$. (4) Learn a Bayesian network with the structure $\boldsymbol{G}^+$ where its parameters have been obtained randomly from a Dirichlet distribution with hyperparameter 0.5. (5) Sample the Bayesian network to obtain a data set of size 500. (6) Compute the set of weights required to solve Problem 2 starting from the M2DG $\boldsymbol{G}$. The experiments have been repeated 100 times for each $n \in \{20, 40, 80, 160\}$. We would like to highlight that, in general, the distribution represented by the generated Bayesian network can not be represented by a decomposable model with an M3DG structure. Moreover, the treewidth of the obtained DAG is higher than 3 with a high probability, the treewidth tends to increase with n and the optimum solution can contain edes of arbitrary lenght. The generated instances of Problem $(k+1)$DG can be considered difficult because the percentage of EOIs with positives scores ranges from 30% for $n=20$ to 5% for $n=160$.

Table 1: Summary of the results obtained for the generated instances.

	Problem features					Greedy	ILP			
n	l_{max}	Vars.	Cons-1	Cons-2	Cons-3	Weight	Weight	time	Gap	APD
20	2	26.00	26.00	0.00	39.40	394.75	237.54	0.01	0.00	-39
	4	176.06	86.02	180.08			573.93	0.02	0.00	64
	8	552.98	159.58	786.80			737.81	0.23	0.00	126
	16	653.64	171.00	965.28			747.12	0.36	0.00	130
	∞	653.64	171.00	965.28			747.12	0.40	0.00	130
40	2	57.58	57.58	0.00	91.72	514.57	290.68	0.01	0.00	-40
	4	461.00	216.48	489.04			884.18	0.04	0.00	136
	8	2255.92	548.68	3414.48			1257.55	3.85	0.00	269
	16	4097.04	738.20	6717.68			1304.75	330.92	0.97	284
	∞	4143.14	741.00	6804.28			1305.63	319.49	1.68	284
80	2	120.46	120.46	0.00	200.88	472.60	323.62	0.01	0.00	-31
	4	1054.26	486.02	1136.48			998.26	0.04	0.00	147
	8	6839.56	1526.08	10626.96			1649.52	168.36	0.17	333
	16	21694.60	2923.66	37541.88			1838.99	2924.95	42.72	385
	∞	24501.84	3081.00	42841.68			1789.32	2411.26	53.88	372
160	2	250.40	250.40	0.00	454.16	395.17	316.26	0.01	0.00	-19
	4	2321.34	1058.00	2526.68			1107.13	0.14	0.00	220
	8	17210.62	3703.58	27014.08			2225.06	779.55	4.80	592

The experimental results outlined in this section were obtained on a cluster of PCs with "Intel(R) Xeon(R) CPU E5450" CPUs of 8 nuclii of 3000 MHz and 32 Gigabyte of RAM. Moreover, the Integer Linear Programm (ILP) was solved with IBM ILOG CPLEX V12.1 [11]. A run of CPLEX was stopped once (at least) 3600 CPU seconds had passed. The output (result) of CPLEX is the value of the best feasible integer solution found within the allowed CPU time limit. We have considered ILP models with less than $50,000$ decision variables. The AF has been compared against the EF, which corresponds to the AF with $l_{max} = \infty$. Besides, the results obtained by CPLEX are compared against the ones of a greedy approach where, at each step, the algorithm considers the addition of the edge $\{u, v\}$ due to S with the maximum weight among the set of candidate edges.

The summary of the obtained results is shown in Table 1. The column l_{max} represents the threshold distance of the AF where the value ∞ represents the EF. The columns Vars., Cons-1, Cons-2 and Cons-3 show the average number of decision variables and constraints of type I, II and II, respectively, for the Integer Linear Programming formulation of the problem. The column Weight shows the average objective function values (divided by the number of added edges) obtained by Greedy and ILP. The column gap provides information about the average optimality gap (in percent), which refers to the gap between the value of the best valid solution that was found and the current lower bound obtained by CPLEX concerning the tackled ILP model at the time of stopping a run. The column APD is the average percentage deviation of ILP with respect to Greedy.

The results obtained for the AF with $l_{max} = 8$ are very similar to those of the EF in terms of the quality of the solution. Besides, the best solutions have been obtained using a small percentage of the time spent by EF. Moreover, with $l_{max} = 16$ AF has obtained better values than EF in instances with $n = 80$ variables. AF always obtains better solutions than Greedy for $l_{max} > 2$, and the APD obtained with respect to Greedy grows dramatically as n increases, even for $l_{max} = 4$. For $n = 180$ the maximum value for l_{max} that generate ILP models with less than $50,000$ decision variables is 8, and it has obtained an average GAP of 4.8% with respect to its best approximate solution to the problem. In terms of the size of the produced model, we can see that for a given l_{max} the number of variables and type II constraints seems to grow linearly with respect to n.

In summary, the parameter l_{max} of AF seems to be effective in order to control the size of the ILP model. Besides, AF leads to ILP models with a good trade-off between the computational resources required and the quality of the obtained solutions when an appropriate value for l_{max} is selected.

4 Conclusions

In this work, we propose an approximate ILP formulation to Problem $(k+1)$DG that only takes into account the subset of EOIs with a maximum length of l_{max}. This formulation discards the EOIs that, on average, have a lower probability than any of the selected EOIs to be part of the optimal solution to instances of Problem $(k+1)$DG. Moreover, the approximate formulation can be used in order to control the size of the ILP model as the number of vertices of the input MkDG increases. We have shown empirically that the approximate formulation can obtain solutions of good quality to Problem $(k+1)$DG, even for small values of l_{max}. In addition, this formulation can be used in order to control the trade-off between the available computational resources and the quality of the obtained solutions.

We believe that the use of AF in combination with strategies from ILP, such as column addition or cutting planes, will allow to produce solutions of high quality to instances of Problem $(k+1)$DG with hundreds and, possibly, thousands of vertices. Following with this idea, in the future we plan to investigate column addition strategies inspired by the proposed distance-based criterion.

Acknowledgments.

A. Pérez and J. A. Lozano were partially supported by the Saiotek and IT609-13 programs (Basque Government), TIN2010-14931 (Spanish Ministry of Science and Innovation), COMBIOMED network in computational bio-medicine (Carlos III Health Institute). C. Blum was supported by project TIN2012-37930 of the Spanish Government. In addition, support is acknowledged from IKERBASQUE (Basque Foundation for Science).

References

[1] Corander, J., Janhunen, T., Rintanen, J., Nyman, H. L., Pensar, J., 2013. Learning chordal markov networks by constraint satisfaction. In: Advances in Neural Information Processing Systems (NIPS). pp. 1349–1357.

[2] Deshpande, A., Garofalakis, M., Jordan., M. I.: Efficient stepwise selection in decomposable models. Proceedings of UAI, 128–135 (2001)

[3] Elidan, G., Gould, S.: Learning bounded treewidth Bayesian networks. Journal of Machine Learning Research, 9:2699–2731 (2008)

[4] Kangas, K., Miinimäki, T., Koivisto, M., 2014. Learning chordal markov networks by dynamic programming. In: Advances in Neural Information Processing Systems (NIPS). pp. 2357–2365.

[5] Koller, D. and Friedman, N.: Probabilistic Graphical Models. Principles and Techniques. The MIT Press, Cambridge, Massachusetts (2009)

[6] Lauritzen, S. L.: Graphical Models. Oxford University Press, New York (1996)

[7] Malvestuto, F. M.: Approximating discrete probability distributions with decomposable models. IEEE Trans on SMC, 21(5):1287–1294 (1991)

[8] Pérez, A., Blum, C. and Lozano, J. A.: Learning Maximum Weighted (k+1)-Order Decomposable Graphs by Integer Linear Programming. Probabilistic Graphical Models, Lecture Notes on Computer Science, 8754:396-408 (2014)

[9] Srebro, N.: Maximum likelihood markov networks: An algorithmic approach. Master thesis, MIT (2000)

[10] Srebro, N.: Maximum likelihood bounded tree-width Markov networks. Artificial Intelligence, 143:123-138 (2003)

[11] IBM ILOG CPLEX VI2.1: *User's Manual for CPLEX*, International Business Machines Corporation (2009)