Improved Acyclicity Reasoning for Bayesian Network Structure Learning

Supplementary materials of IJCAI'2021 paper

June 22, 2021

We made a longer run with a 90-hour CPU-time limit for 41 very large instances on a cluster of Intel Xeon E5-2683 v4 at 2.10 GHz and 128 GB of RAM. We used the following parameters $l_{min} = 20, l_{max} = 20, r_{min} = 15, r_{max} = 30$ for partition lower bound min/max sizes l_{min}, l_{max} and local search min/max number of restarts r_{min}, r_{max}. *E.g.,* we ran `./elsa -B 20 -r 15 -R 30 -t 324000 kdd.test.jkl` for solving `kdd.test` within the 90-hour CPU-time limit. We compared with the original version of CPBayes and with GOBNILP version 1.6.3 based on SCIP version 3.2.1 and cplex version 12.8.0 with default parameters, except optimality gap set to zero.

1 Quality of Lower Bounds and Time per Node

We measured the quality of the initial lower bound lb found at the root node of the search tree as the relative distance $\frac{o-lb}{o}$ to the best solution o found by any method within the 90-hour CPU-time limit. Results are given in Table 1. GOBNILP usually produced the best lower bounds. CPBayes got the worst ones and ELSA made good progress towards GOBNILP but still remained below except a few exceptions (GOBNILP stopped at the root node due to exhausted memory for `accidents.valid`, `dna`, `kosarek.valid`, `msweb.test`, and `msweb.valid`).

In Figure 1, we show the mean CPU-time per search node when information was available for a subset of the large instances. We sort the data by increasing CPU-time independently for each solver. Here, GOBNILP is several orders of magnitude slower than CPBayes and our approach. ELSA was usually slower than CPBayes, up to 457.4 times for `bnetflix.test`, except on a few cases, *e.g.,* ELSA was 9.7 faster than CPBayes on `pumsb_star.test`.

In Figure 2, we compare the number of nodes in the search tree for CPBayes and ELSA on instances solved by the two approaches. Because this common number of solved instances is too small, we compared on the smaller instances with less than 64 variables as used in the main paper. The points below the line

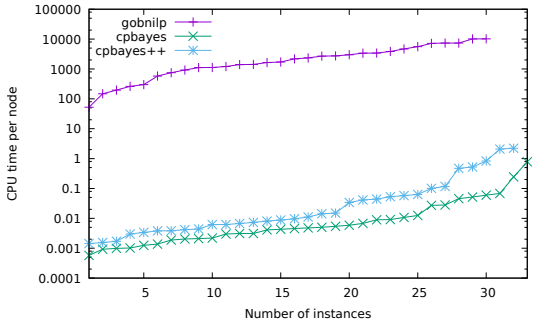

Figure 1: CPU-time (in seconds) per search node on some large instances.

$y = x$ indicate a smaller search tree for ELSA. The reduction in the search tree size is very clear.

2 Quality of Solutions and Optimality Proofs

We measured the quality of a BNSL solution s as the relative distance $\frac{s-o}{o}$ to the best solution o found by any method within the 90-hour CPU-time limit. The results are given in Table 2. Notice that CPBayes and ELSA have a local search procedure in preprocessing. It helps the search to find good initial solutions.

Concerning optimality proofs, ELSA solved 17 instances, GOBNILP solved 9, and CPBayes 4 only. GOBNILP ran out-of-memory (128GB) on 11 instances whereas ELSA took less than 8GB on all the benchmark.

3 Impact of the Cut Pool Activity Threshold

We made a last experiment to see if the activity threshold has an impact on the overall performance of our approach. Figure 3 gives the total number of cluster cuts in the pool at the end of the search for 17 solved instances. Let $a_{\mathbf{C}}$ be the number of times cluster $\mathbf{C}$ has improved the lower bound. We found that keeping in the pool clusters during $100 \times a_{\mathbf{C}}$ or up to $100,000 \times a_{\mathbf{C}}$ search nodes does not change the whole search effort in terms of total search nodes to solve a problem (Figure 4). However it may increase CPU-time, but this could also be due to performance fluctuations of the cluster (Figure 5).

By default, we choose an activity threshold of $\frac{1}{1,000}$, *i.e.* keeping each cluster $\mathbf{C}$ during $1,000 \times a_{\mathbf{C}}$ search nodes. Clusters of size less than or equal to 10 were always kept. Such clusters are often more productive and their number is bounded, see for example cluster activities in `kdd.test` (Figure 6). On 41 large instances, the number of cuts found in preprocessing ranged from 328 (`accidents.test`) to 4,011 (`kosarek.valid`) with a mean of 2,011.4. At the end of the search (or 90-hour CPU-time limit), there were from 85

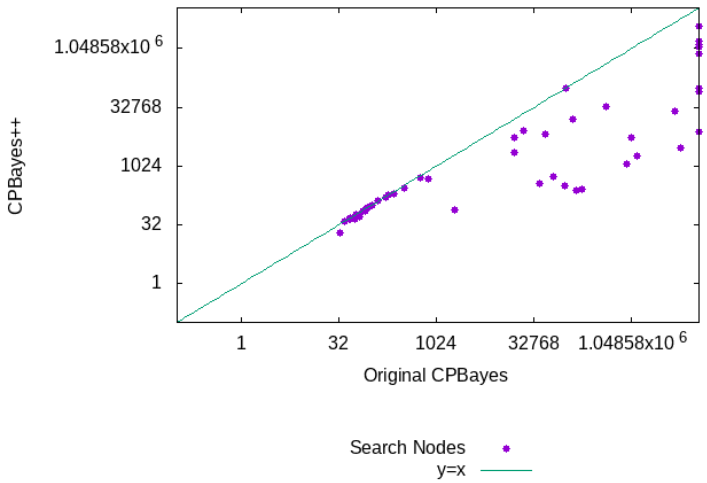

Figure 2: Number of search tree nodes for CPBayes and ELSA on smaller instances with less than 64 variables.

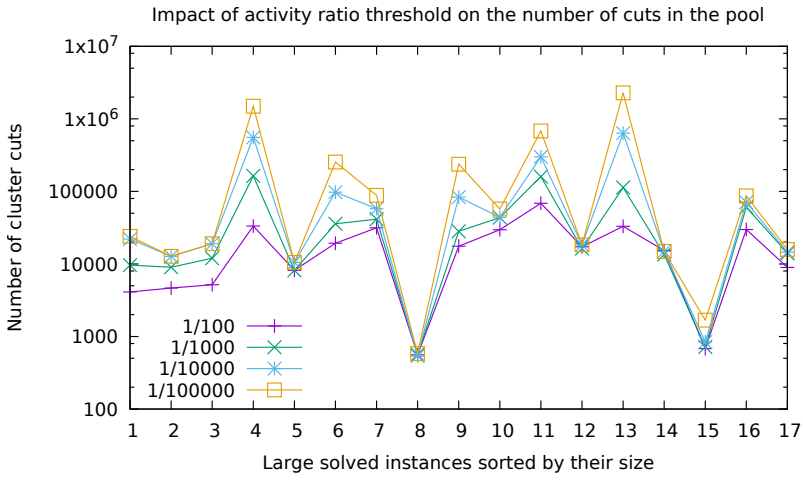

Figure 3: Number of cluster cuts in the pool at the end of the search when solving large instances depending on the activity ration threshold.

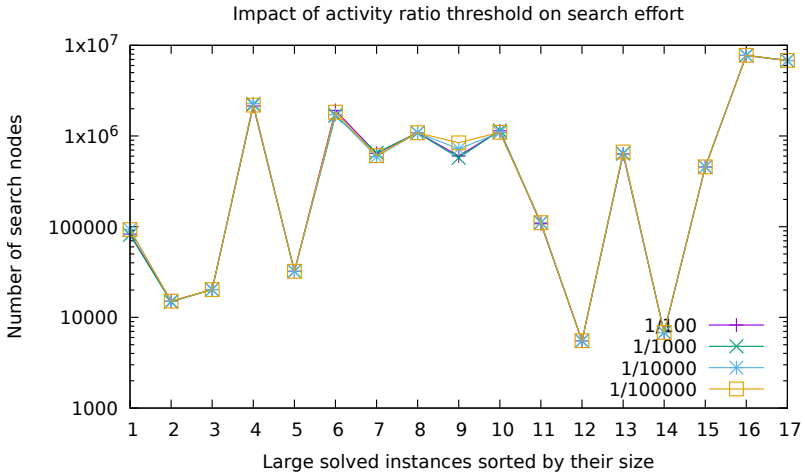

Figure 4: Number of search nodes for solving large instances depending on the activity ration threshold.

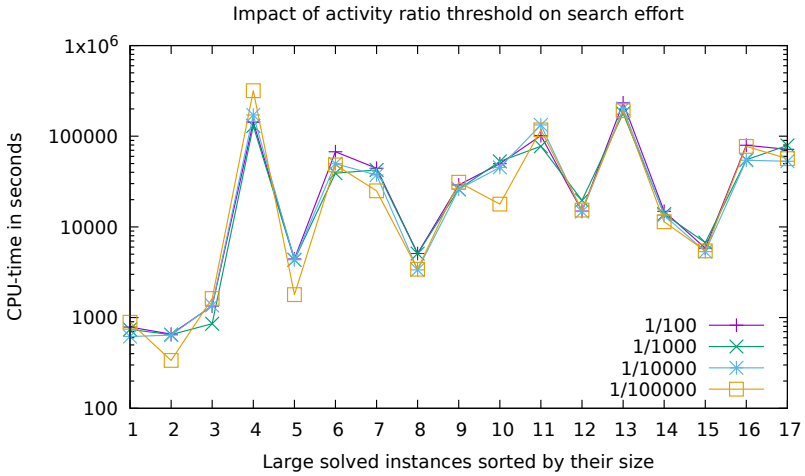

Figure 5: CPU-time for solving large instances depending on the activity ratio threshold.

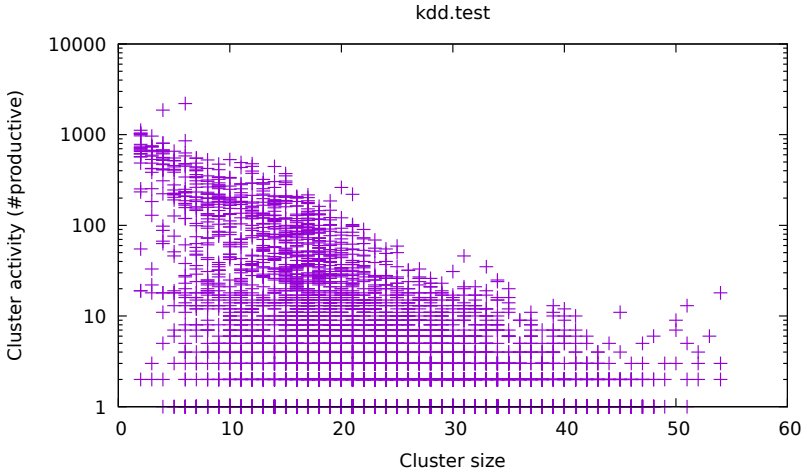

Figure 6: Cluster activity (*#productive*) w.r.t. size for 9,018 cuts kept in the cluster pool at the end of the search, after 14,840 search nodes in 613 seconds on `kdd.test` with 64 random variables.

(`pumsb_star.valid`) to 163,710 (`baudio.ts`) clusters in the pool with a mean of 26,638.3 clusters.

Problem	$\|\mathbf{V}\|$	$\sum_{v\in\mathbf{V}} \|ps(v)\|$	GOBNILP	CPBayes	ELSA
kdd.ts	64	43584	⋆**0.03%**	2.51%	⋆0.50%
kdd.test	64	152873	⋆**0.02%**	⋆1.75%	⋆0.20%
plants.ts	69	164640	⋆**2.42%**	35.01%	7.28%
kdd.valid	64	197546	⋆**0.13%**	2.60%	⋆0.49%
baudio.ts	100	371117	**2.00%**	11.34%	⋆3.63%
pumsb_star.ts	163	394992	⋆**0.19%**	24.69%	0.46%
tretail.ts	135	435976	⋆0.17%	0.48%	**0.07%**
bnetflix.ts	100	446406	**3.23%**	⋆20.86%	⋆8.73%
plants.test	69	520148	**2.58%**	34.32%	⋆5.98%
jester.ts	100	531961	**4.20%**	13.03%	⋆6.44%
kosarek.ts	190	556189	⋆**0.00%**	2.13%	0.44%
accidents.ts	111	568160	⋆**0.00%**	8.36%	⋆0.38%
plants.valid	69	684141	**2.57%**	34.61%	⋆6.66%
msweb.ts	294	732213	**0.25%**	1.83%	0.62%
diabetes-5000	413	754563	**0.26%**	0.67%	0.67%
jester.test	100	770950	**4.76%**	13.29%	⋆6.37%
tretail.test	135	897478	**2.88%**	4.44%	3.16%
baudio.test	100	1016403	**2.22%**	11.92%	⋆3.10%
pumsb_star.test	163	1034955	23.79%	68.12%	**21.61%**
tretail.valid	135	1087404	**0.67%**	2.60%	2.60%
bnetflix.test	100	1103968	**5.49%**	⋆24.48%	⋆11.69%
kosarek.test	190	1192386	**1.06%**	4.38%	2.60%
baudio.valid	100	1235928	**2.37%**	12.09%	⋆3.32%
pumsb_star.valid	163	1271525	33.69%	75.89%	**21.63%**
kosarek.valid	190	1312600	**2.20%**	5.84%	5.84%
bnetflix.valid	100	1325818	**5.38%**	⋆25.26%	⋆11.24%
accidents.test	111	1425966	⋆**0.00%**	20.03%	⋆3.71%
jester.valid	100	1463335	**5.64%**	17.03%	⋆7.63%
msweb.test	294	1597487	0.53%	3.81%	**0.37%**
accidents.valid	111	1617862	**3.57%**	34.77%	7.83%
dna.ts	180	1849860	**1.07%**	5.53%	1.52%
msweb.valid	294	1869374	**0.78%**	3.59%	3.59%
tmovie.ts	500	1918883	**35.15%**	-	-
pigs-5000	441	1984359	**24.50%**	-	-
dna.test	180	2019003	**1.07%**	5.66%	⋆1.62%
book.ts	500	2071722	**7.56%**	-	-
dna.valid	180	2561134	3.90%	9.92%	**3.72%**
tmovie.valid	500	2610026	**49.06%**	-	-
tmovie.test	500	2778556	**53.78%**	-	-
book.test	500	2794588	**15.08%**	-	-
book.valid	500	3020475	**20.24%**	-	-

Table 1: Quality of initial lower bounds on 41 large instances. The best method is shown in bold. '⋆': optimum proved, '-': no lower bound reported.

Problem	$\|\mathbf{V}\|$	$\sum_{v \in \mathbf{V}} \|ps(v)\|$	GOBNILP	CPBayes	ELSA
kdd.ts	64	43584	⋆**0.00%**	0.17%	⋆**0.00%**
kdd.test	64	152873	⋆**0.00%**	⋆**0.00%**	⋆**0.00%**
plants.ts	69	164640	⋆**0.00%**	3.02%	1.48%
kdd.valid	64	197546	⋆**0.00%**	0.16%	⋆**0.00%**
baudio.ts	100	371117	1.62%	1.29%	⋆**0.00%**
pumsb_star.ts	163	394992	⋆**0.00%**	4.76%	4.76%
tretail.ts	135	435976	⋆**0.00%**	0.29%	0.29%
bnetflix.ts	100	446406	0.01%	⋆**0.00%**	⋆**0.00%**
plants.test	69	520148	0.01%	2.61%	⋆**0.00%**
jester.ts	100	531961	10.29%	0.50%	⋆**0.00%**
kosarek.ts	190	556189	⋆**0.00%**	1.62%	1.62%
accidents.ts	111	568160	⋆**0.00%**	0.37%	⋆**0.00%**
plants.valid	69	684141	1.74%	2.40%	⋆**0.00%**
msweb.ts	294	732213	9.87%	**0.00%**	**0.00%**
diabetes-5000	413	754563	2.86%	**0.00%**	**0.00%**
jester.test	100	770950	7.45%	0.18%	⋆**0.00%**
tretail.test	135	897478	6.93%	0.03%	**0.00%**
baudio.test	100	1016403	13.69%	0.92%	⋆**0.00%**
pumsb_star.test	163	1034955	102.62%	**0.00%**	**0.00%**
tretail.valid	135	1087404	17.60%	0.38%	**0.00%**
bnetflix.test	100	1103968	0.41%	⋆**0.00%**	⋆**0.00%**
kosarek.test	190	1192386	†15.70%	**0.00%**	**0.00%**
baudio.valid	100	1235928	12.46%	0.49%	⋆**0.00%**
pumsb_star.valid	163	1271525	155.68%	**0.00%**	**0.00%**
kosarek.valid	190	1312600	†25.15%	**0.00%**	**0.00%**
bnetflix.valid	100	1325818	7.07%	⋆**0.00%**	⋆**0.00%**
accidents.test	111	1425966	⋆**0.00%**	1.71%	⋆**0.00%**
jester.valid	100	1463335	7.86%	0.68%	⋆**0.00%**
msweb.test	294	1597487	†14.99%	**0.00%**	**0.00%**
accidents.valid	111	1617862	†433.17%	0.32%	**0.00%**
dna.ts	180	1849860	†16.82%	**0.00%**	**0.00%**
msweb.valid	294	1869374	†14.82%	**0.00%**	**0.00%**
tmovie.ts	500	1918883	**0.00%**	-	-
pigs-5000	441	1984359	†**0.00%**	-	-
dna.test	180	2019003	†22.33%	0.51%	⋆**0.00%**
book.ts	500	2071722	**0.00%**	-	-
dna.valid	180	2561134	†25.77%	**0.00%**	**0.00%**
tmovie.valid	500	2610026	**0.00%**	-	-
tmovie.test	500	2778556	**0.00%**	-	-
book.test	500	2794588	†**0.00%**	-	-
book.valid	500	3020475	†**0.00%**	-	-

Table 2: Solution quality on 41 large instances. The best method is shown in bold. '⋆': optimum found, '†': out-of-memory, '-': no solution found.