Benchmark on DAOOPT and GUROBI with the PASCAL2 Inference Challenge Problems

Junkyu Lee, William Lam, Rina Dechter
Department of Computer Science
University of California, Irvine, U.S.A
junkyul@uci.edu,{willmlam,dechter}@ics.uci.edu

Abstract

We report the PASCAL2 benchmark for DAOOPT and GUROBI on MPE task with 330 optimally solved instances from 8 benchmark domains. DAOOPT outperformed GUROBI in 3 domains, while GUROBI was faster than DAOOPT in the rest of the 5 domains. We show that DAOOPT performed well in domains where it could have high quality initial solutions for pruning the AND/OR search space, or skip search when the heuristic upper bounds were converged to the optimal due to MPLP/JGLP algorithms. GUROBI presented excellent performance if cutting planes were applied progressively and its heuristic algorithms could find the optimal solution at the root of branch-and-cut tree.

1 Introduction

We benchmarked DAOOPT [1] and GUROBI [2] with selected problems from the 2011 PASCAL2 Inference Challenge [3] to see how DAOOPT, a state-of-the-art competitive Branch-and-Bound solver for graphcial models that uses different principles for generating a lower bound and pruning than ILP solvers, compares with GUROBI, a professional MILP solver that is optimized and includes lessons studied over 30-40 years. In addition, a comparison can direct ideas to improve both solvers by combining their benefits.

DAOOPT is a Breadth Rotating AND/OR Branch-and-Bound (BRAOBB) [4-5] based solver for finding Most Probable Explanation (MPE) over graphical models, and is equipped with algorithmic components for enhancing the approximate upper-bounds, finding variable orderings, and picking up initial solutions via stochastic local search. More specifically, Message Passing Linear Programming(MPLP) [6], Join Graph Linear Programming(JGLP) [7], and Mini Bucket Elimination with Moment Matching(MBE-MM) [8] produce tighter upper bounds for the maximization problem on the top of the static heuristics from Mini-Bucket Elimination(MBE) [9]. Stochastic greedy ordering scheme(CVO) [10] efficiently generates orderings with low induced widths by using min-fill with stochastic tie-breaking and early cut-offs. This plays an important role the efficiency of BRAOBB as its time complexity is exponential in the induced width w ($O(n \cdot k^w)$). Guided Local Search+(SLS) [11] is used to find a good initial solution to allow pruning of a significant amount of subtrees, improving search time. GUROBI is a commercial mathematical programming solver, and we are interested in its MILP solver.

We used MPE to 0-1 ILP conversion illustrated in [12], and compared the runtime of DAOOPT and GUROBI. It should be noted that the performance of GUROBI is subjected to the property of the conversion employed, and we do not address the issue of performance variation due to graphical model to ILP conversion methods. Section 2 introduces the benchmark setup, Section 3 reports benchmark results of selected problem sets with additional benchmark results available on our website[1]. We conclude in section 4.

[1] http://sites.uci.edu/automatedreasoninggroup/

2 Benchmark Setup

Benchmark Platform We ran the benchmarks on a platform with two 2.66 GHz Intel Core2 Duo Processors and 6 GB memory, running a Linux operating system. The total amount of memory available for both solvers was limited to 4 GB. We also focus these benchmarks on serial performance, so we limited the cores used to 1.

Parameter	set 1	set 2	set 3
MBE-MM i-bound	maximum i selected by DAOOPT within 4GB memory		
MPLP iteration limit	2 sec	30 sec / 500 iter	60 sec / 2000 iter
JGLP iteration limit	2 sec	30 sec / 250 iter	60 sec / 2000 iter
CVO iteration limit	3 sec / 500 iter	60 sec/ 10000 iter	180 sec / 30000 iter
SLS time limit	2 x 2 sec	10 x 6 sec	20 x 10 sec
BRAOBB time limit	To termination or timeout greater than termination time of GUROBI		

Table 1: DAOOPT input parameters. The same sets were used for the PASCAL2 competition, and we matched them to the difficulty of benchmark domains; the set 1 was used for easy instances expected to be terminated within a minute while the set 3 cope with the hardest problem instances.

DAOOPT Input Parameters DAOOPT has five algorithmic components, MBE-MM, MPLP, JGLP, CVO, and SLS, that require input parameters; Table 1 defines three input parameter sets to cope with different levels of difficulty based on the increased time at preprocessing step. The i-bound for the MBE is chosen to be the maximum available within the memory limit, and the time limit of BRAOBB was specified per problem. we set iteration limit and time limit for MPLP, JGLP, and CVO, so that they could be terminated when one of the limits reached. SLS executed multiple times with preset time duration.

GUROBI Input Parameters GUROBI is comprised of numerous algorithms for finding heuristic solutions, cutting planes, managing the branch-and-cut tree, etc. We used the default options and explicitly set the dual simplex method as the root relaxation.

Graphical Model A graphical model is a tuple $\mathbf{R} =< \mathbf{X}, \mathbf{D}, \mathbf{F}, \prod >$, where $\mathbf{X} = \{x_1, x_2, \cdots, x_n\}$ is a set of variables, $\mathbf{D} = \{D_1, D_2, \cdots, D_n\}$ is a set of domains of variables, $x_i \in D_i$, $\mathbf{F} = \{F_1, F_2, \cdots, F_m\}$ is a set of factors defined over a set $S = \{S_1, S_2, \cdots, S_m\}$ of a scope $S_j \subseteq \mathbf{X}$ for the factor F_j, and $\prod$ is a combination operator. The MPE task is to find $max_{\mathbf{X}} \prod_{j=1}^{m} F_j(\mathbf{x_j})$, where the $\mathbf{x_j}$ is a vector of variables on the S_j. In particular, a probability distribution can be written as $\mathbf{F}(\mathbf{X}) = \frac{1}{Z} \prod_{j=1}^{m} F_j(\mathbf{x_j}) = \frac{1}{Z} \exp(-\sum_{j=1}^{m} g_j(\mathbf{x_j}))$, and the MPE task is $min_{\mathbf{X}} \sum_{j=1}^{m} g_j(\mathbf{x_j})$, where the Z is a normalization constant, and the $g_j(\mathbf{x_j}) = -\log F_j(\mathbf{x_j})$.

MPE to 0-1 ILP Conversion [12] A set of binary decision variables $V = \{v_j^k\}$ is defined over the kth row of the jth factor tables. Mutual exclusivity constraints enforce a decision variable to select a row exclusively within a factor, $\sum_{k=1}^{j_r} v_j^k = 1$, the sum of all decision variables in the jth factor table equals 1. Cross consistency constraints constrain such selections to be consistent across all pairs of factor assignments, $\sum_{k \sim z} v_p^k = \sum_{l \sim z} v_q^l$, where $z \in \prod_i D_i$, $i \in S_p \cap S_q$, the sum of all decision variables over all instantiations of an intersection of scope of the factor p and q is the same. The objective is $max \sum_j \sum_k v_j^k \cdot F_j^k$ maximizing the sum of decision variables weighted by its corresponding factor value. Thus, the number of decision variables is $O(m \cdot K^S)$, where the m is the numer of factor tables, the K is the maximum domain size and the S is the maximum arity of factors, and the number of constraints is $O(m^2 \cdot K^{(S-1)})$.

Benchmark Problem Domains We report the size of benchmarks and the problem statistics; *Grid(Markov)*, *Segmentaion* and *WCSP* were divided into subgroups based on the input parameters to avoid bias due to the different level of difficulty. There are three parts in Table 2. The first two columns are benchmark domain information: the name of benchmark domain with a tuple $(\bar{n}, \bar{f}, \bar{k}, \bar{s}, \bar{w}, \bar{h})$ showing average of the number of variables, the number of factors, maximum domain sizes, maximum arities, induced widths of orderings, and pseudotree heights, the number of optimally solved instance out of the total number of instances $(\frac{\#opt}{\#tot})$ for each solver. The following part is the problem statistics based on the graphical model representation for DAOOPT: the minimum and maximum number of nodes, factors, domain sizes, arities, induced widths, and pseudotree heights. The last two columns are dimension of constraint matrices for GUROBI: the minimum and maximum number of rows and columns.

		DAOOPT						GUROBI	
Name	$(\frac{\#opt}{\#tot})_{da}$	n_{min}	f_{min}	k_{min}	s_{min}	w_{min}	h_{min}	row_{min}	col_{min}
$(\bar{n},\bar{f},\bar{k},\bar{s},\bar{w},\bar{h})$	$(\frac{\#opt}{\#tot})_{gu}$	n_{Max}	f_{Max}	k_{Max}	s_{Max}	w_{Max}	h_{Max}	row_{Max}	col_{Max}
Grid(Bayes)	32/32	144	145	2	3	16	53	914	1058
(649,650,**2**,**3**,35,121)	32/32	2500	2501	2	3	81	309	17102	19602
Pedigree	22/22	298	335	3	4	15	53	2518	4476
(736,918,5,4,24,99)	22/22	1015	1290	7	5	35	160	8429	9986
Grid(Markov).20x20	5/5	400	1161	2	2	26	66	8529	3841
(**400**,1185,**2**,**2**,38,70)	5/5	400	1201	2	2	46	73	9201	4001
Grid(Markov).40x40	8/8	1600	4721	2	2	53	144	35449	15681
(**1600**,4761,**2**,**2**,74,147)	8/8	1600	4801	2	2	95	149	36801	16001
Grid(Markov).80x80	0/8	6400	19041	2	2	107	307	14489	63361
(**6400**,19121,**2**,**2**,152,310)	8/8	6400	19201	2	2	197	313	147201	64001
Segmentation.K2	50/50	221	823	2	2	15	44	8823	2845
(229,852,**2**,**2**,17,55)	50/50	237	887	2	2	19	68	9699	3071
Segmentation.K21	50/50	221	823	21	2	15	44	84823	269263
(229,852,**21**,**2**,17,55)	50/50	237	887	21	2	19	68	93451	291187
Promedas	68/68	196	201	2	3	4	33	625	1001
(832,843,**2**,**3**,37,79)	68/68	1911	1928	2	3	116	162	9856	11565
WCSP.Spot5	6/6	67	272	4	2	6	15	4291	2572
(132,648,**4**,3,14,39)	5/6	209	1395	4	3	26	87	88620	12390
WCSP.Easy	8/8	25	82	3	2	6	15	1357	730
(38,264,5,**2**,13,21)	8/8	71	686	11	2	21	30	168583	9479
WCSP.Hard	3/4	16	208	4	2	7	8	10585	3777
(72,1948,18,**2**,23,37)	3/4	179	7110	44	2	42	90	104103361	340305
Protein Folding	7/7	337	1360	81	2	22	58	297301	27094
(926,3710,**81**,**2**,30,111)	1/7	1364	5220	81	2	38	164	1030679	2699568
DBN	60/60	70	16167	2	2	29	29	12986347	34691
(**70**,**16167**,2,2,**29**,**29**)	0/60	70	16167	2	2	29	29	12986347	34691

Table 2: Benchmark domain statistics: The domain name with average number of variables, factors, maximum domain sizes, maximum arities, induced widths, pseudotree heigths; The number of optimally solved instances out of the total number of instances; the range of problem statistics based on graphical models for DAOOPT; the range of constraint matrice dimension for GUROBI.

3 Benchmark Results

Runtime Comparison Per Instance In Figure 1, we show runtimes from selected instances from benchmark domains. Both solvers often shows similar difficulty for instances in *Grid(Bayes)*, *Pedigree*, *Grid(Markov)*, *Segmentation* or *WCSP*. For example, each of *Grid.90-50-5*, *Pedigree19*, *Grid.40x40.f10.wrap*, *10-14-s.21*, *WCSP.bwt4ac* was harder than that of *Grid.90-30-5*, *Pedigree20*, *Grid.40x40.f10*, *10-14-s.bin*, *WCSP.Spot5.54*.

Several challenging instances are reported separately in Table 5. For DAOOPT: *pedigree40*, 2650 seconds; *Grid.90-46-5*, 9336 seconds; All 80x80 Grid instances, where DAOOPT was 5400 seconds time out (DAOOPT could find the optimal solution for *Grid.80x80.f2* after 4677 seconds with pre-computed variable order). For GUROBI: *WCSP.Spot5.42*, 2 day time out, which is the only time out instance for GUROBI; *WCSP.myciel5g-3*, 1012 seconds; *WCSP.myciel5g-4*, 5281 seconds.

Mean Runtime Comparison Per Domain Optimal solutions from 330 instances are obtained, after excluding indeterminate cases. Table 3 reports mean runtime on each benchmark domain, where some domains further divided into subgroups based on the difficulty. *Par* is the input parameters, the tuple $(i_m, \bar{i}, i_M)$ shows the range and average of i-bounds used, $\frac{\#daowin}{\#ins}$ is the number of DAOOPT wins over total instances, following four columns are the arithmetic and geometric mean of runtimes, and the arithmetic mean of ratios of runtimes of GUROBI over DAOOPT, where a ratio was calculated per instance at the last column.

Name	Par	$(i_m,\bar{i},i_M)$	$\frac{\#daowin}{\#Ins}$	$(\sum T_d)/N$	$(\sum T_g)/N$	$(\prod T_d)^{1/N}$	$(\prod T_g)^{1/N}$	$(\sum \frac{T_g}{T_d})/N$
Grid(Bayes)	set1	(16,22.31,25)	0/32	382.8	2.7	51.01	1.35	0.038
Pedigree	set2	(10,17.91,23)	0/22	232.4	27.3	181.41	9.66	0.096
Grid(Markov).20x20	set1	(20,20,20)	0/5	14.6	7.2	14.28	7.21	0.524
Grid(Markov).40x40	set2	(20,20,20)	0/8	212.1	84.1	212.10	78.22	0.396
Grid(Markov).80x80	set3	(18,18,18)	0/8	Tout(5400)	2125	Tout(5400)	1661	-
Segmentation.K2	set1	(15,17.14,19)	0/50	7.14	0.26	7.12	0.26	0.037
Segmentation.K21	set1	(4,4,4)	0/50	55.4	11.36	55.33	11.23	0.205
Promedas	set1	(4,20.60,27)	0/68	143.71	0.2	40.57	0.16	0.006
WCSP.Spot5	set1	(6,12,19)	4/6	59.5	197.65 (5)	18.91	47.31 (5)	23.23
WCSP.Easy	set1	(6,10,15)	8/8	43	620.33	19.88	94.99	14.83
WCSP.Hard	set3	(3,7.5,11)	3/4	1208 (3)	5388 (2)	729 (3)	5386 (2)	-
ProteinFolding.4G	set3	(3,3,3)	6/7	3005	79 (1)	1004	79 (1)	-
ProteinFolding.(No M Lim)	-	-	-	-	702	-	448	-
DBN	set3	(28,28,28)	60/60	342.35	Mout(4GB)	342.35	Mout(4GB)	-

Table 3: Mean runtime (sec) comparison per benchmark domain. The input parameters for DAOOPT (*par*) the minimum, average, and maximum i-bounds, the number of instances DAOOPT won out of total. The arithmetic and geometric mean of runtimes from DAOOPT and GUROBI, and the arithmetic mean of ratios of each instance runtime. A number in parentheis is the number of instances averaged to exclude time or memory out cases.

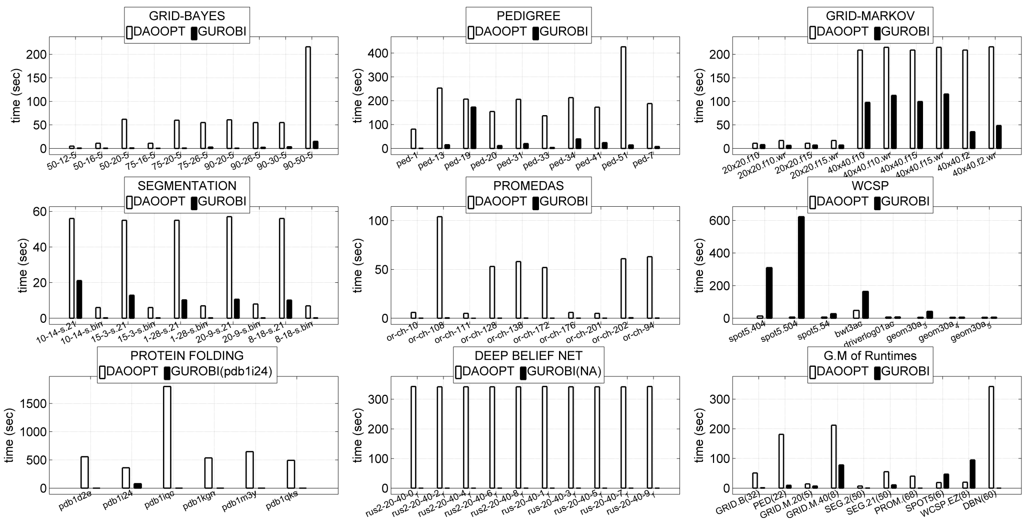

Figure 1: Runtime comparison per selected instances. Instance names in each domain are at the horizontal axis. GUBORI was memory out for all in *DBN*, and all except *pdb1i24* in *Protein Folding*. The last subplot visualizes geometric mean of runtimes.

DAOOPT found 319 optimal solutions except 8 instances in *Grid(Markov).80x80*, and *myciel5g-5* in *WCSP.Hard* due to the time limit; GUROBI found 259 except *spot5.42* in *WCSP.Easy*, *bwt4ac* and *CELAR6-SUB0* in *WCSP.Hard*, 6 instances in *Protein Folding*, and 60 instances in *DBN* due to the memory limit. The geometric mean of runtimes shows that GUROBI was superior to DAOOPT in *GRID(Bayes)*, *Pedigree*, *Grid(Markov)*, *Segmentation*, and *Promedas*; DAOOPT was favorable in *WCSP*, *Protein Folding*, and *DBN*. Note that GUROBI was beaten by DAOOPT in *Protein Folding* and *DBN* because of the memory limit; releasing memory limit in *Protein Folding* resulted in returning 7 optimal solutions equipped with 2 new optimal solutions in 447.61 sec on average.

Log Summary Table 4 has two parts and each part summarizes log data produced by DAOOPT and GUROBI. T_{sls} is the amount of time spent for SLS, $E[Gap_{sls-opt}]$ and $E[Gap_{sls-ub}]$ are the average gap from SLS solutions to optimal solutions, $|\frac{OptimalSolution-SLSSolution}{OptimalSolution}|$, and from SLS solutions to heuristic upperbounds in similar manner. *#sls hit opt* and *#sls hit ub* are the number of SLS solutions met optimal solutions and heuristic upper bounds. The fifth column is the average number of AND/OR nodes explored during search.

GUROBI log statistics are presented in the second part. $E[node]$ is the average number of nodes branched, and *#solve at root* counts cases where GUROBI found optimal solution at the root node. The last two columns are the average number of simplex iterations ($iter$), gomory cuts ($gomory$), zero half cuts ($zerohalf$), and clique cuts ($clique$).

	DAOOPT				GUROBI		
Name	Par	$E[Gap_{sls-opt}]$	$E[Gap_{sls-ub}]$	$E[OR]$	$E[node]$	$E[iter]$	$E[zerohalf]$
$(\bar{n},\bar{f},\bar{k},\bar{s},\bar{i},\bar{w},\bar{h})$	T_{sls}	#sls hit opt	#sls hit ub	$E[AND]$	#solved at root	$E[gomory]$	$E[clique]$
Grid(Bayes)	set1	0.40%	0.95%	29779829.13	0.5	4804.16	71.69
(649,650, 2, 3,22,35,121)	4	14/28	8/28	32069771.13	24/32	5.59	68.66
Pedigree	set2	0.01%	0.76%	14040647	210.32	41678.45	240.64
(736,918,5,4,18,24,99)	60	15/22	4/22	17328814.82	1/22	5.95	14.32
Grid(Markov).20x20	set1	1.28%	1.39%	1326.2	0	4259.8	336.4
(400,1185, 2, 2,20,38,70)	4	0/5	0/5	1379.2	5/5	10.4	7.2
Grid(Markov).40x40	set2	1.86%	1.90%	24042.38	1.13	16378.25	994.88
(1600,4761, 2, 2,20,74,147)	60	0/8	0/8	24835.75	7/8	16.25	26
Grid(Markov).80x80	set3	2.46%	2.59%	Tout(5400)	176.88	101722.5	3828.5
(6400,19121, 2, 2,18,152,310)	200	0/8	0/8	Tout(5400)	4/8	42.75	513.13
Segmentation.K2	set1	0.00%	0.00%	133.34	0	1225	0
(229,852, 2, 2,17,17,55)	4	50/50	50/50	134.34	50/50	0	0
Segmentation.K21	set1	0.00%	0.00%	96.08	0	4080.6	0
(229,852, 21, 2,4,17,55)	4	50/50	50/50	97.08	50/50	0	0
Promedas	set1	0.64%	14.37%	9011410.53	0	1351.79	0.09
(832,843, 2, 3,21,37,79)	4	64/68	23/68	9673994.4	68/68	0.03	0
WCSP.Spot5	set1	0.00%	1.73%	2110421.67	982.8	112844	407
(132,648, 4,3,12,14,39)	4	6/6	5/6	2737228.5	1/5	14.6	741.8
WCSP.Easy	set1	0.00%	16.72%	1724404.38	27037.5	514113	1.75
(38,264,5, 2,10,13,21)	4	8/8	5/8	2890242.25	2/8	0	0
WCSP.Hard	set3	0.00%	82.22%	52348330.33	100192.5	3742815.5	206.5
(72,1948,18, 2,8,23,37)	200	4/4	0/4	88802002	0/2	0	0
Protein Folding	set3	0.00%	0.26%	20609318.71	0	9487	0
(926,3710, 81, 2,3,30,111)	200	7/7	1/7	22866496.29	1/1	0	0
DBN	set3	0.00%	3.12%	2456.73	Mout(4GB)	Mout(4GB)	Mout(4GB)
(70,16167,2,2,3,29,29)	200	60/60	1/60	3096.98	0/0	Mout(4GB)	Mout(4GB)

Table 4: Log summary. DAOOPT log statistics: the average gap from SLS solutions to optimal solutions, the number of optimal SLS solutions, the average gap from SLS solutions to heuristic upper bounds, and the number of exact heuristic upper bounds. GUROBI log statistics: the average number of nodes branched, the number of instances terminated at the root node, the average number of simplex iterations, gomory cuts, zero half cuts, clique cuts.

DAOOPT : Quality of Initial Lower Bounds from SLS and Static Heuristic Upper Bounds The comparison between the inital solution and the optimal solution shows that for all DAOOPT winning instances except *WCSP.bwt4ac* instances, SLS found the optimal solution; search often used to prove its optimality. If a good initial solution is combined with an exact heuristic upper bound, the runtime of DAOOPT improves significantly. In Table 4, it is shown that SLS solution was the same as the heuristic upper bound in *Segmentation*, so search terminated only after exploring 134.34 and 97.08 AND nodes on average. Similary, 10 instances out of 18 *WCSP* instances also terminated immediately after preprocessing step.

GUROBI : Number of nodes branched and Cutting Planes GUROBI showed its superior performance over DAOOPT in *Grid(Bayes)*, *Pedigree*, *Grid(Markov)*, *Segmentation*, and *Promedas*. For 209 instances among 243 instances GUROBI terminated at the root node. Thus, GUROBI was able to find optimal solution before branching by cutting plane and heuristic algorithms.

Detailed Benchmark Results from Selected Instances Table 5 reports individual results from selected instances. The tuple (n, f, k, s, w, h) is the problem statistics for each instance as previously defined. The first part of the table corresponds to DAOOPT: i is the i-bound used, *MBytes* is the amount of memory used to compile the Mini-Bucket Heuristics with the i-bound, *Sol* is the optimal solution found, $Gap_{sls-opt}$ is the gap between SLS solution and the optimal solution, *MBE-UB* is the Mini-Bucket heuristic upper bound, Gap_{sls-ub} is the gap between SLS solution and the MBE-UB, T_{pre} is preprocessing time, and T_d is total elapsed time. The following three columns are results from GUROBI: T_g is total elapsed time, T_g/T_d is the ratio of elapsed times, *iter* is the number of simplex iterations, and *nodes* is the number of nodes explored. DAOOPT found the optimal SLS solution, and the exact heuristic upper bound in many cases: pedigree1, 15-3-s.21, pdb1d2e, and etc. GUROBI solved instances extremely fast when it could find the optimal solution and prove its optimality at the root node. For example, *90-30-5*, *pedigree1*, *15-3-s.21*, *pdb1i24*, and etc. DAOOPT explores the AND/OR search space efficiently when the initial solution was close to the optimal though the static heuristic upper bound was not converged to the optimal. For *myciel5g-4*, DAOOPT terminated in 2,890 seconds after exploring 156,881,144 OR nodes and 265,874,839 AND nodes, while GUROBI had to branch 85,887 nodes and found optimal solutions after 5,281 seconds.

4 Conclusion

Running the PASCAL benchmarks extended our knowledge of its problem sets and equips it with optimal solutions that can be useful for additional testing by other solvers. High quality SLS solutions often lead DAOOPT to terminate early because it could prune huge portion of the AND/OR search space. DAOOPT was favorable in benchmark domains like *WCSP* where SLS almost picked optimal solutions, along with being able to use mini-bucket heuristics with a i-bound close to the induced width, where the heuristic upper bound becomes more accurate or even converged to the optimal solution. GUROBI showed fast performance when it could apply cutting planes and find solutions within the desired gap from the best bound before branching. In future work, DAOOPT could consider the advantage of dynamic heuristic evaluation schemes, but the overhead of mini-bucket compilation time needs to be overcome. GUROBI may improve its performance further by incorporating stronger heuristic algorithms similar to SLS.

	DAOOPT						GUROBI		
Name (n,f,k,s,w,h)	i-bound / mbytes	sol / Par	sls sol / $Gap_{sls-opt}$	mbe-ub / Gap_{sls-ub}	or / and	T_{pre} / T_d	T_g / T_g/T_d	sol / status	iter / nodes
Grid(Bayes)									
75-20-5	25	-12.72	-12.82	-12.64	426	60	0.99	-12.72	2809
(400,401,2,3,28,95)	3666.53	Set 1	0.01	0.01	428	60	0.02	opt	1
90-46-5	19	-28.374	log(0)	-26.18	816759831	59	13.54	-28.374	18021
(2116,2117,2,3,74,240)	2618.6	Set 1	-	-	869534974	9336	0.0014	opt	1
90-30-5	21	-13.12	-13.33	-12.8	3691	55	3.61	-13.12	6529
(900,901,2,3,45,160)	2557.86	Set 1	0.02	0.04	3757	55	0.07	opt	0
Pedigree									
pedigree1	15	-45.58	-45.58	-45.58	0	81	0.7	-45.58	2759
(298,335,4,5,15,60)	46.38	Set 2	0	0	1	81	0.01	opt	0
pedigree19	16	-97.09	-97.19	-94.96	3365767	183	173.66	-97.09	286437
(693,794,5,5,23,143)	2796.57	Set 2	0	0.02	4133870	207	0.84	opt	1326
pedigree40	14	-130.345	-130.345	-126.951	269860028	165	221.15	-130.345	318223
(842,1031,7,5,28,160)	1829.93	Set 2	0.000774867	0.02603859	337356564	2796	0.079095136	opt	1666
pedigree41	18	-120.74	-120.74	-118.72	958379	167	24.5	-120.74	48112
(885,1063,5,5,30,120)	2089.86	Set 2	0	0.02	1117709	173	0.14	opt	503
Grid(Markov)									
grid20x20.f10.wrap	20	1324.3	1302.01	1326.05	2954	17	6.52	1324.3	4052
(400,1201,2,2,46,73)	505.19	Set 1	0.02	0.02	3113	17	0.38	opt	0
grid40x40.f10	20	5504.38	5413.87	5505.01	7815	209	97.88	5504.38	15820
(1600,4721,2,2,53,149)	1659.18	Set 2	0.02	0.02	8081	209	0.47	opt	0
grid40x40.f10.wrap	20	5674.43	5551.37	5676.88	11312	215	112.82	5674.43	19740
(1600,4801,2,2,95,144)	2065.05	Set 2	0.02	0.02	11724	215	0.52	opt	0
grid80x80.f2	18	4677.73	4601.17	4681.23	-	541	577.2	4677.73	53907
(6400,19042,2,2,107,307)	2784.87	Set 3	0.016368	0.017102343	-	Tout(5400)	-	opt	0
Segmenation									
15-3-s.21	4	-138.71	-138.71	-138.71	231	55	12.91	-138.71	6602
(231,859,21,2,17,53)	351.15	Set 1	0	0	232	55	0.23	opt	0
15-3-s.binary	17	-30.17	-30.17	-30.17	231	6	0.27	-30.17	1298
(231,859,2,2,17,53)	10.03	Set 1	0	0	232	6	0.05	opt	0
Promedas									
or-chain-10.fg	18	-9.3	-9.3	-9.3	0	6	0.08	-9.3	774
(453,462,2,3,18,50)	10.09	Set 1	0	0	1	6	0.01	opt	0
or-chain-172.fg	22	-7.5	-7.5	-4.2	38700	52	0.17	-7.5	1170
(739,751,2,3,36,84)	2378.99	Set 1	0	0.44	41410	52	0	opt	0
WCSP									
Spot5.404	19	-2.78	-2.78	-2.78	100	14	310.58	-2.78	213579
(100,711,4,3,19,45)	204.17	Set 1	0	0	101	14	22.18	opt	3563
Spot5.503	9	-2.2	-2.2	-2.2	0	7	622.49	-2.2	288714
(143,636,4,3,9,44)	1.37	Set 1	0	0	1	7	88.93	opt	1134
GEOM30a-3	6	-44	-44	-44	48	6	42.42	-44	101117
(30,82,3,2,6,15)	0.02	Set 1	0	0	61	6	7.07	opt	11735
myciel5g-3	14	-64	-64	-24.96	2498486	33	1012.8	-64	907835
(47,237,3,2,21,24)	981.14	Set 1	0	0.61	3834486	54	18.76	opt	31991
queen5-5-4	12	-48	-48	-20.6	10857881	69	977.24	-48	913025
(25,161,4,2,18,20)	1804.73	Set 3	0	0.57	18721553	183	5.34	opt	26154
myciel5g-4	11	-16	-16	0	156881144	319	5281.76	-16	4933083
(47,237,4,2,21,24)	1296	Set 3	0	1	265874839	2890	1.83	opt	85887
bwt4ac	7	-0.21237	-0.21	-0.13	11120	397	-	-	-
(179,7110,18,2,42,90)	1332.35	Set3	0	0.37	12974	400	-	Mout	-
Protein Folding									
pdb1d2e	3	-754.99	-754.99	-753.57	26321	554	-	-	-
(1328,5220,81,2,22,136)	433.64	Set 3	0	0	29818	557	-	Mout	-
pdb1i24	3	-190.83	-190.83	-190.83	0	361	78.54	-190.83	9487
(337,1360,81,2,33,58)	60.15	Set 3	0	0	1	361	0.22	opt	0

Table 5: Detailed benchmark results for selected instances from Figure 1 and hard domains.

Acknowledgement

This work was supported by NSF grant IIS-1065618.

References

[1] L. Otten, A. Ihler, K. Kask, and R. Dechter. Winning the pascal 2011 map challenge with enhanced and/or branch-and-bound. In *NIPS Workshop DISCML*, 2012.

[2] Z. Gu, E. Rothberg, and R. Bixby. Gurobi optimizer v5.5. `http://www.gurobi.com`.

[3] The 2011 PASCAL2 Probabilistic Inference Challenge. `http://www.cs.huji.ac.il/project/PASCAL/`.

[4] R. Marinescu. *AND/OR Search Strategies for Combinatorial Optimization in Graphical Models*. PhD thesis, University of California, Irvine, 2008.

[5] L. Otten. *Extending the Reach of AND/OR Search for Optimization in Graphical Models*. PhD thesis, University of California, Irvine, 2013.

[6] A. Globerson and T. S. Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In *NIPS*, pages 553–560, 2007.

[7] R. Dechter A. Ihler, N. Flerova and L. Otten. Join-graph based cost-shifting schemes. 2012.

[8] R. Dechter N. Flerova, A. Ihler and L. Otten. Mini-bucket elimination with moment matching. In *NIPS Workshop DISCML*, 2011.

[9] Rina Dechter and Irina Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM (JACM)*, 50(2):107–153, 2003.

[10] L. Otten K. Kask, A. Gelfand and R. Dechter. Pushing the power of stochastic greedy ordering schemes for inference in graphical models. In *AAAI*, 2011.

[11] Frank Hutter. SLS4MPE. `http://www.cs.ubc.ca/labs/beta/Projects/SLS4MPE/`.

[12] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*, pages 577–579. The MIT Press, 2009.