Evaluating Weighted DFS Branch and Bound over Graphical Models

Natalia Flerova
University of California
Irvine

Radu Marinescu
IBM Research
Dublin, Ireland

Rina Dechter
University of California
Irvine

Abstract

Weighted search was explored significantly in recent years for path-finding problems, but until now was barely considered for optimization tasks such as MPE/MAP and Weighted CSPs. An important virtue of weighted search schemes, especially in the context of anytime search, is that they are w-optimal, i.e. when terminated, they return a weight w, and a solution cost C, such that $C \leq w \cdot C^*$, where C^* is the optimal cost.

In this paper we introduce Weighted Branch and Bound (WBB) for graphical models and provide a broad empirical evaluation of its performance compared with one of the best unweighted anytime search scheme, BRAOBB (won Pascal 2011 competition). We also compare against weighted best-first (WBF). Our results show that WBB can be superior to both un-weighted BB and to weighted BF on a significant number of instances. We also illustrate the benefit of weighted search in providing suboptimality relative error bounds.

Introduction

Depth-first Branch-and-Bound is the most commonly used scheme for combinatorial optimization tasks, such as MAP/MPE or Weighted CSP, and it was extensively studied in recent years (Kask and Dechter 2001; Marinescu and Dechter 2009b; Otten and Dechter 2011; de Givry, Schiex, and Verfaillie 2006). In contrast, in path-finding domains, e.g., planning, best-first search and, especially, its variant A* (Hart, Nilsson, and Raphael 1968) are far more popular, and for good, well-known reasons. Yet, A*'s exponential memory and lack of anytime performance lead to extensions into more flexible anytime schemes based on *Weighted* $A*$ (WA*) (Pohl 1970). The idea is to inflate the heuristic function guiding the search by a constant factor of $w > 1$, making the heuristic inadmissible, and the overall search more greedy-like (depending on the magnitude of w). It typically yields faster and less memory intensive search, while still guaranteeing a solution cost within a factor of w from the optimal. If the (non-optimal) solution is found quickly, the search for a better solution may continue with a smaller weight.

In our exploration of the weighted search approach for graphical models we focused first on weighted best-first search (Flerova, Marinescu, and Dechter 2014b). We adapted weighted best-first to graphical models and evaluated the resulting schemes' performance as anytime algorithms on some graphical models domains. We illustrated the effectiveness of these schemes on some benchmarks and, moreover, the value of their w-optimality. A summary of the extensive evaluation we carried out is presented here for completeness. Yet, the notorious memory issues of best-first search often persisted and debilitated its potential.

Therefore, our contribution in this paper is in extending weighted search to depth-first Branch and Bound, known for its memory efficiency, yielding yet another new anytime scheme which is accompanied with suboptimality guarantees. We provide an extensive comparative empirical evaluation demonstrating the power of this approach.

As a basis we use AND/OR Branch and Bound search (AOBB (Marinescu and Dechter 2009a)), a depth-first Branch and Bound algorithm developed for graphical models. AOBB explores the context minimal AND/OR graph and is guided by an admissible and consistent mini-bucket heuristic (Dechter and Rish 2003; Kask and Dechter 2001; Dechter and Mateescu 2007). Since AOBB does not have good anytime behaviour due to the underlying AND/OR search space, it was extended into an algorithm called Breadth-Rotating AND/OR Branch and Bound (BRAOBB (Otten and Dechter 2011)). BRAOBB won the 2011 Probabilistic Inference Challenge[1] in all optimization categories. We therefore extended both AOBB and BRAOBB into weighted anytime schemes.

The two resulting anytime algorithms: Weighted AOBB (wAOBB) and Weighted BRAOBB (wBRAOBB) run iteratively using weighted heuristic. After each iteration a (possibly suboptimal) solution and the weight is reported and the weight is decreased according to a fixed weight schedule. We report on a comprehensive empirical evaluation on over 100 instances from 4 benchmarks.

Our empirical analysis shows that the weighted Branch and Bound schemes often exhibit anytime performance superior to the state-of-the-art BRAOBB and anytime weighted best-first search. In particular, on two of the four

[1]http://www.cs.huji.ac.il/project/PASCAL/realBoard.php

benchmarks used (Pedigrees and Type4) for some i-bounds and time limits wAOBB and wBRAOBB report solutions of better quality than BRAOBB on more than half of the instances.

Overall, the results vary significantly across benchmarks, with no single algorithm being superior for all domains. However, combined in a portfolio this collection of schemes can potentially yield a powerful and efficient anytime solver (Huberman, Lukose, and Hogg 1997).

Background

Weighted best-first search exploits the idea of making an admissible evaluation function (i.e. one that never overestimates a cost to the goal, assuming minimization) inadmissible, by multiplying it by some weight $w > 1$. In particular, the most well-known variant of Weighted best-first search, *WA**, uses evaluation function $f(n) = g(n) + w \cdot h(n)$, where $w > 1$, $g(n)$ is the current minimal cost from the root to n, and $h(n)$ is the heuristic function that estimates the optimal cost to go. Higher values of w typically yield greedier behaviour, finding a solution earlier during search and with less memory. WA* is guaranteed to be w*-optimal*, namely to terminate with a solution cost C such that $C \leq w \cdot C^*$, where C^* is the optimal solution's cost (Pohl 1970). In the past decade, several anytime weighted Best-First search schemes were proposed in the context of path-finding problems (Hansen and Zhou 2007; Likhachev, Gordon, and Thrun 2003; van den Berg et al. 2011; Richter, Thayer, and Ruml 2010).

A *Graphical model* is a tuple $\mathcal{M} = (\mathbf{X}, \mathbf{D}, \mathbf{F}, \prod)$, where $\mathbf{F}$ is a set of real-valued local functions over subsets of discrete variables $\mathbf{X}$, called scopes, with finite domains $\mathbf{D}$. The scopes of $\mathbf{F}$ define a *primal graph* G with certain *induced width* and a *pseudo tree* $\mathcal{T}$ of G that guides an AND/OR search space (Dechter and Mateescu 2007). The common optimization task is to find $\max_{\mathbf{X}} \prod_i f_i$ called *MAP* or *MPE* or *Weighted CSP* problem: $\min_{\mathbf{X}} \sum_i f_i$. These two tasks are equivalent and are easily transformed into one another. In our discussion we assume the min-sum task.

A context minimal *AND/OR search graph* $\mathcal{S}_{\mathcal{T}}$, which is guided by a pseudo-tree $\mathcal{T}$ and whose identical subproblems are merged, has size $O(nk^{w^*})$, where w^* is the induced width of G along a depth-first order of $\mathcal{T}$, n is the number of variables and k bounds the domain sizes.

AND/OR Best First search (*AOBF*) (Marinescu and Dechter 2009b) and AND/OR Branch and Bound (*AOBB*) (Marinescu and Dechter 2009a) are search algorithms traversing AND/OR search space. They are guided by the mini-bucket heuristic which is admissible and consistent (Kask and Dechter 1999). The accuracy of the heuristic is controlled by a parameter *i-bound* (higher i-bounds typically yield more accurate heuristics and take more time and space ($exp(i)$) to compute).

AND/OR Best First Search (AOBF) (Marinescu and Dechter 2009a). AOBF is a state-of-the-art version of A* for the AND/OR search space for graphical models. AOBF is a variant of AO* [(Nillson 1980)] that explores the AND/OR context-minimal search graph. Unlike usual A*, AOBF does not have explicit OPEN and CLOSED lists, instead maintaining the explicated part of the context minimal AND/OR search graph and keeping track of the current best partial solution tree T^*. AOBF interleaves a top-down step of expanding the nodes in the best-first manner with bottom-up update of nodes' values and recaltulation of T^*.

Algorithm 1: wAOBF(w_0, h_i)

```
Input: A graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$; heuristic $h_i$
       calculated with i-bound $i$; initial weight $w_0$, weight
       update schedule $S$
Output: Set of suboptimal solutions $\mathcal{C}$
1 Initialize $w = w_0$ and let $\mathcal{C} \leftarrow \emptyset$;
2 while $w >= 1$ do
3     $\langle C_w, T^*_w \rangle \leftarrow$ AOBF($w \cdot h_i$);
4     $\mathcal{C} \leftarrow \mathcal{C} \cup \{\langle w, C_w, T^*_w \rangle\}$;
5     Decrease weight $w$ according to schedule $S$;
6 return $\mathcal{C}$;
```

AND/OR Branch and Bound (AOBB) (Marinescu and Dechter 2009a): AOBB traverses the weighted context-minimal AND/OR graph in a depth-first manner while keeping track of the current upper bound on the minimal solution cost. A node n will be pruned if this upper bound exceeds a heuristic lower bound on the solution to the subproblem below n.

Breadth-Rotating AND/OR Branch and Bound (BRAOBB) (Otten and Dechter 2011). Though Branch and Bound search is inherently anytime, AND/OR decomposition hinders anytime performance of AOBB, which has to solve completely all but one independent child subproblems at each AND node before the last one is even considered. BRAOBB is an anytime version of AOBB, which works by rotating through different subproblems in a breadth-first manner. It empirically proved to be a far more efficient anytime algorithm than plain AOBB.

Weighted Best-First Search: Summary of Previous Results

In this section we present an overview of our previously conducted investigation of anytime weighted best-first search (Flerova, Marinescu, and Dechter 2014b).

Tailoring Weighted BFS to Graphical Models

AND/OR Best-First search using the mini-bucket heuristic, multiplied by the weight $w > 1$, yields *Weighted AOBF*. Clearly, the cost of the solution found by this scheme is guaranteed to be w-optimal (see also (Chakrabarti, Ghose, and De Sarkar 1987)).

Using Weighted AOBF algorithm as a basis, we extended two anytime weighted best-first algorithms, popular in path-finding domain, to searching over AND/OR spaces, yielding schemes wAOBF and wR-AOBF.

Iterative Weighted AOBF (wAOBF) : (Algorithm 1) executes Weighted AOBF iteratively, decreasing the weight at each iteration. This approach, similar to the Restarting Weighted

Algorithm 2: wR-AOBF(w_0, h_i)

```
Input: A graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$; pseudo tree $\mathcal{T}$
       rooted at $X_1$; heuristic $h_i$ for i-bound=$i$; initial weight
       $w_0$, weight update schedule $S$
Output: Set of suboptimal solutions $\mathcal{C}$
1  Initialize $w = w_0$ and let $\mathcal{C} \leftarrow \emptyset$;
2  Create root OR node $s$ labeled by $X_1$ and let $\mathcal{G} = \{s\}$;
3  Initialize $v(s) = w \cdot h_i(s)$ and best partial solution tree $T^*$ to
   $\mathcal{G}$;
4  while $w >= 1$ do
5      Expand and update nodes in $\mathcal{G}$ using AOBF($w$,$h_i$) search
       with heuristic function $w \cdot h_i$;
6      If $T^*$ has no more tip nodes then
       $\mathcal{C} \leftarrow \mathcal{C} \cup \{\langle w, v(s), T^* \rangle\}$;
7      Decrease weight $w$ according to schedule $S$;
8      For all leaf nodes in $n \in \mathcal{G}$, update $v(n) = w \cdot h_i(n)$.
       Update the values of all nodes in $\mathcal{G}$ using the values of
       their successors. Mark best successor of each OR node.;
9      Recalculate $T^*$ following the marked arcs;
10 return $\mathcal{C}$;
```

A* (Richter, Thayer, and Ruml 2010), results in a series of solutions, each with a smaller suboptimality bound of the weight w.

Anytime Repairing AOBF (wR-AOBF). Running each search iteration from scratch and thus possibly exploring the same search subspace multiple times is redundant. To remedy this problem we introduced wR-AOBF (Algorithm 2), an extension of the *Anytime Repairing A* (ARA*)* algorithm (Likhachev, Gordon, and Thrun 2003) to the AND/OR search spaces over graphical models. wR-AOBF also runs iteratively, gradually decreasing the weight, but unlike wAOBF, it keeps track on the expanded nodes and updates their evaluation function whenever the weight changes between the iterations. wR-AOBF is also a w-optimal algorithm.

Evaluating Anytime Weighted Best-First

In (Flerova, Marinescu, and Dechter 2014b) the anytime performance of wAOBF and wR-AOBF was evaluated and compared with state-of-the-art anytime Branch and Bound scheme BRAOBB, exploring the same search space. The algorithms return improved solutions until the optimal solution is found or until either the time limit of 1 hour or memory limit of 4 GB is reached. We implemented our algorithms in C++ and ran all experiments on a 2.67GHz Intel Xeon X5650, running Linux, with 4 GB allocated for each job.

We used benchmarks from UAI 2008 competition [2] and 2011 Probabilistic Inference Challenge [3], which include probabilistic graphical models and weighted CSPs. For uniformity we assume that the optimization task is maximization throughout. The benchmark parameters are presented in Table 1. In random grid networks, the nodes are arranged in an NxN square and the functions are defined over pairs of

Benchmark	# inst	n	k	w^*	h_T
Pedigrees	11	581-1006	3-7	16-39	52-102
Grids	32	144-2500	2	15-90	48-283
Type4	10	3907-8186	5	21-32	319-625
WCSP	56	25-1057	2-100	5-287	11-337

Table 1: Benchmark parameters: # inst - number of instances, n - number of variables, k - domain size, w^* - induced width, h_T - pseudo-tree height.

variables and are generated uniformly randomly. The pedigree and type4 instances come from the domain of genetic linkage analysis and are associated with the task of haplotyping. The Weighted CSP networks includes graph colouring problems, SPOT5 networks and other Weighted CSP domains.

Each problem was solved with 10 i-bounds, ranging from 2 to 20. Both weighted schemes used the $sqrt$ weight policy: $w_j = \sqrt{w_{j-1}}$, where w_j is the weight at iteration j. This policy proved to be the most effective out of the five candidates studied in our preliminary evaluation, reported in (Flerova, Marinescu, and Dechter 2014a). The initial weight was set to 64.

Table 2 presents a summary of the results. For 3 time bounds we show the percentage of instances for which wAOBF and wR-AOBF respectively find better solutions than BRAOBB, the percentage for which they are tied with BRAOBB and the number of instances, for which at least one of the algorithms found a (possibly suboptimal) solution. We show the results for a relatively small and a large i-bounds.

Analyzing the results presented in Table 2 and based on the full data in (Flerova, Marinescu, and Dechter 2014b) we observe:

1. Anytime weighted best-first schemes are superior to BRAOBB in quite a few cases. However, their performance varies a lot across benchmarks. wAOBF and wR-AOBF yielded better solutions than BRAOBB on a large percentage of instances on Grids (up to 72.7%) and Type4 (up to 100%), but are less effective on Pedigrees (up to 55.6%) and clearly inferior on WCSPs (only better than BRAOBB on 6.4% for presented i-bounds).

2. Weighted best-first algorithms tend to be superior when the i-bound is small and there is a large gap between the i-bound and the problems instances (e.g. for i-bound=4). However, when the i-bound is large (i.e. strong heuristics) their dominance is usually less pronounced.

3. wAOBF and wR-AOBF often have better performance for short time limits (e.g. for Grids, i=4, wAOBF is superior to BRAOBB on 72.7% of problems for 60 seconds, but only 29.2% for 3600 seconds).

4. wAOBF seems somewhat superior to wR-AOBF, dominating BRAOBB more often.

Overall, we concluded that though anytime weigted best-first schemes are not universally superior to BRAOBB, they definitely are often competitive.

[2]http://graphmod.ics.uci.edu/group/Repository

[3]http://www.cs.huji.ac.il/project/PASCAL/archive/mpe.tgz

I-bound	Algorithm	Time bounds		
		60	600	3600
		X% / Y% / N	X% / Y% / N	X% / Y% / N
Grids (# inst=32, n=144-2500, k=2, w^*=15-90, h_T=48-283)				
i=4	wAOBF	72.7 / 9.1 / 22	41.7 / 25.0 / 24	29.2 / 29.2 / 24
	wR-AOBF	40.9 / 9.1 / 22	16.7 / 25.0 / 24	12.5 / 29.2 / 24
i=16	wAOBF	20.0 / 72.0 / 25	7.4 / 77.8 / 27	0.0 / 75.0 / 28
	wR-AOBF	4.0 / 72.0 / 25	0.0 / 77.8 / 27	3.6 / 75.0 / 28
Pedigrees (# inst=11, n=581-1006, k=3-7, w^*=16-39, h_T=52-104)				
i=4	wAOBF	55.6 / 0.0 / 9	33.3 / 11.1 / 9	11.1 / 22.2 / 9
	wR-AOBF	11.1 / 0.0 / 9	0.0 / 11.1 / 9	0.0 / 11.1 / 9
i=16	wAOBF	42.9 / 28.6 / 7	11.1 / 33.3 / 9	0.0 / 44.4 / 9
	wR-AOBF	28.6 / 14.3 / 7	0.0 / 22.2 / 9	0.0 / 22.2 / 9
WCSP (# inst=56, n=25-1057, k=2-100, w^*=5-287, h_T=11-337)				
i=4	wAOBF	4.3 / 21.3 / 47	2.0 / 26.5 / 49	5.5 / 23.6 / 55
	wR-AOBF	6.4 / 23.4 / 47	2.0 / 26.5 / 49	5.5 / 23.6 / 55
i=12	wAOBF	0.0 / 80.0 / 5	0.0 / 35.7 / 14	0.0 / 35.7 / 14
	wR-AOBF	0.0 / 80.0 / 5	0.0 / 35.7 / 14	0.0 / 35.7 / 14
Type4 (# inst=10, n=3907-8186, k=5, w^*=21-32, h_T=319-625)				
i=4	wAOBF	70.0 / 0.0 / 10	90.0 / 0.0 / 10	100.0 / 0.0 / 10
	wR-AOBF	20.0 / 0.0 / 10	30.0 / 0.0 / 10	30.0 / 0.0 / 10
i=16	wAOBF	100.0 / 0.0 / 6	100.0 / 0.0 / 10	100.0 / 0.0 / 10
	wR-AOBF	83.3 / 0.0 / 6	50.0 / 0.0 / 10	50.0 / 0.0 / 10

Table 2: X% - percentage of instances for which each algorithm is the better than BRAOBB at a specific time bound, Y% - percentage of instances for which algorithm ties with BRAOBB, N - number of instances for which at least one of algorithms found a solution. # inst - total number of instances in benchmark, n - number of variables, k - maximum domain size, w^* - induced width, h_T - pseudo-tree height. 4 GB memory, 1 hour time limit.

.

Weighted Branch and Bound Search

In the following section we present and explore the potential of weighted depth-first Branch and Bound.

We denote by AOBB(h_i,w_0,*UB*) the weighted version of AOBB that uses the mini-bucket heuristic h_i with i-bound=i, with the weight w_0 and where initial the upper bound used for pruning is *UB* (default *UB*$=\infty$). It is easy to show that:

Theorem 1. *If the heuristic $h(n)$ is admissible, the cost of the solution discovered by weighted AOBB is w-optimal.* □

Iterative Weighted AOBB (wAOBB): extends AOBB(h_i,w_0,*UB*) to an iterative scheme similar to wAOBF (Algorithm 3). During first iteration wAOBB executes AOBB(h_i,w_0,$UB=\infty$). When AOBB terminates, it reports its final solution cost C_1 and the value of the (initial) weight $w_1 = w_0$. At each subsequent iteration j, where $j \geq 2$ wAOBB executes AOBB(h_i,w_j,UB_j) to completion. The weight w_j is decreased according to some predetermined weight policy. The upper bound UB_j depends on the cost of the solution with which the previous iteration terminates, i.e. $UB_j = C_{j-1}$. This upper bound is

Algorithm 3: wAOBB(w_0, h_i)

```
Input: A graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$; heuristic $h_i$
       obtained with i-bound $i$; initial weight $w_0$
Output: $\mathcal{C}$ - a set of suboptimal solutions $C_w$, each with a
        bound $w$
1 Initialize $j = 1$, $UB_j = \infty$, $w_j = w_0$, weight update
  schedule $S$ and let $\mathcal{C} \leftarrow \emptyset$;
2 while $w_j >= 1$ do
3     while $T^*$ has more tip nodes, not yet expanded do
4         run AOBB($h_i$,$w_j$, $UB_j$); return the solution bounded
          by the weight of previous iteration:
          $\mathcal{C} \leftarrow \mathcal{C} \cup \{\langle w_{j-1}, C'_j \rangle\}$
5     return the solution with which AOBB terminated,
      bounded by the current weight: $\mathcal{C} \leftarrow \mathcal{C} \cup \{\langle w_j, C_j \rangle\}$
6     Decrease weight $w$ according to schedule $S$;
7     UB$\leftarrow C_j$
8     return $\mathcal{C}$
```

used by AOBB for pruning at the beginning of the iteration. During iterations $j \geq 2$ AOBB returns all the intermediate solutions, whose costs are denoted C'_j and are bounded by weight w_{j-1}, up until it terminates with the final solution, whose cost is denoted C_j and is w_j-optimal.

Iterative Weighted BRAOBB (wBRAOBB): extends BRAOBB(i,w_0,*UB*) to a weighted iterative scheme in the same manner as wAOBB. Clearly, for both schemes the sequence of the solution costs is non-increasing.

Experiments

We evaluate the performances of wAOBB and wBRAOBB and compare with wAOBF and BRAOBB using the same experimental settings and benchmarks as described above for evaluation of weighted best-first algorithms. All weighted schemes use starting weight 64 and the $sqrt$ weight policy. We solve the max-product task throughout.

In our empirical evaluation we will address:

1. The impact of the weight on the solution accuracy and runtime
2. The quality of schemes' anytime behaviour
3. The interaction between heuristic strength and the multiplicative weight

The Impact of Weights

In *Table 3* we report the time (sec) required for each weighted scheme (wAOBF, wAOBB, wBRAOBB) to produce a w-optimal solution, and the corresponding solution cost. For each algorithm we report the entire runtime required to reach the target weight from the starting weight of 64, according to the $sqrt$ policy, and produce a solution bounded by w.

We also report the runtime and the cost by BRAOBB at termination time. A time of 3600 seconds for BRAOBB signifies that it failed to return the optimal solution within the time limit and we report the best solution found. We show two select instances from each benchmark, for a relatively strong heuristic.

Instance	BRAOBB	Scheme	Weights			
			2.8284	1.2968	1.0330	1.000
	time / cost		time / cost	time / cost	time / cost	time / cost
Grids, I-bound=18						
75-22-5 (484, 2, 30, 107)	115.45 / -15.605	wAOBF	5.92 / -15.72	6.66 / -15.72	59.81 / -15.61	423.76 / -15.61
		wAOBB	5.87 / -19.08	6.1 / -17.55	52.46 / -15.65	— / —
		wBRAOBB	5.91 / -19.08	6.22 / -17.55	79.91 / -15.7	— / —
75-25-5 (625, 2, 34, 122)	3582.08 / -20.836	wAOBF	8.0 / -23.38	9.77 / -21.7	— / —	— / —
		wAOBB	8.08 / -31.0	8.34 / -22.55	— / —	— / —
		wBRAOBB	8.14 / -31.0	8.56 / -22.84	— / —	— / —
Pedigrees, I-bound=18						
pedigree9 (935, 7, 27, 100)	220.34 / -122.904	wAOBF	26.75 / -129.55	26.96 / -123.06	33.56 / -122.9	— / —
		wAOBB	12.44 / -129.76	12.47 / -128.56	13.63 / -123.2	— / —
		wBRAOBB	12.59 / -129.76	12.61 / -128.56	13.74 / -123.2	— / —
pedigree51 (871, 5, 39, 98)	3600 / -111.55	wAOBF	120.94 / -119.16	— / —	— / —	— / —
		wAOBB	29.01 / -121.77	31.15 / -121.77	— / —	— / —
		wBRAOBB	26.41 / -121.77	28.36 / -121.77	3035.48 / -109.83	— / —
WCSP, I-bound=6						
capmo2.wcsp (200, 100, 100, 100)	3600 / -0.28	wAOBF	26.81 / -0.31	— / —	— / —	— / —
		wAOBB	22.43 / -0.31	— / —	— / —	— / —
		wBRAOBB	22.47 / -0.31	— / —	— / —	— / —
myciel5g_3.wcsp (47, 3, 19, 24)	12.93 / -64.0	wAOBF	2.52 / -72.0	50.47 / -64.0	— / —	— / —
		wAOBB	0.96 / -72.0	7.8 / -64.0	37.63 / -64.0	— / —
		wBRAOBB	0.9 / -72.0	7.37 / -64.0	35.63 / -64.0	— / —
Type4, I-bound=18						
type4b_120_17 (4072, 5, 24, 319)	3600 / -1332.18	wAOBF	80.49 / -1354.93	81.24 / -1329.59	84.98 / -1327.6	— / —
		wAOBB	49.79 / -1353.83	49.97 / -1337.37	50.25 / -1334.85	— / —
		wBRAOBB	54.91 / -1353.83	55.12 / -1337.37	55.44 / -1334.85	— / —
type4b_130_21 (4874, 5, 29, 416)	3600 / -1383.74	wAOBF	86.21 / -1438.24	88.89 / -1386.34	— / —	— / —
		wAOBB	58.12 / -1414.3	97.66 / -1489.57	— / —	— / —
		wBRAOBB	96.61 / -1512.32	96.93 / -1489.57	— / —	— / —

Table 3: Runtime (sec) and cost obtained by wAOBF, wAOBB and wBRAOBB for selected w, and by BRAOBB (that finds optimal cost). Instance parameters: n - number of variables, k - max domain size, w^* - induced width, h_T - pseudo tree height. "time out" - running out of time, "—/—" - running out of memory. 4 GB memory limit, 1 hour time limit.

Time saving for w-bounded suboptimality. Comparing pairs of columns, in particular column 2 (exact results for BRAOBB) and columns 5-6 (1.2968- and 1.0330-optimal solutions), we observe that the weighted schemes can yield remarkable time savings compared to BRAOBB.

Overall, from all the experiments we observed that the weighted schemes can often provide good approximate solutions with tight suboptimality bounds, yielding significant time savings compared to finding optimal solutions by the competing BRAOBB. The weighted Branch and Bound schemes are more memory efficient than wAOBF. However, on the instances feasible for all three weighted schemes there is no clear dominance.

Anytime Performance

Figure 1 (for Grids and Pedigrees) and *Figure 2* (for Type4 and WCSPs) display the anytime behaviour of the schemes for typical instances from each benchmark. For each instance we show the ratio between the cost returned at a particular time limit (at 10, 60, 600 and 1800 sec) and the optimal (if known) or best cost found. The closer the ratio is to 1, the better. For clarity, we display the interval between 0.7 and 1.0 only. Each row shows a particular i-bound, two per benchmark. The four leftmost bars (for 4 different time points) correspond to wAOBF, next ones, left to right, to BRAOBB, wAOBB and wBRAOBB. We additionally display the weight w for a time bound above the respective bar for the weighted schemes. The cases where BRAOBB proved solution optimality is indicated by '***' above bars.

Grids (Figure 1): we observe that wAOBF is typically superior to Branch and Bound algorithms, returning solutions earlier and of better accuracy. For example, wAOBF is the only scheme to return a solution within 10 seconds on grid 75-23-5, i=6. However, there are exceptions where wAOBB and wBRAOBB are superior to wAOBF, (e.g., grid 50-16-5, i=6). BRAOBB often tends to find the least accurate solutions, providing guarantees only when it proves solution optimality, (e.g., 90-20-5, i=60, 600 and 1800 seconds).

Pedigrees (Figure 1): Weighted Branch and Bound schemes are often superior (e.g. pedigree13, i=6). For some problems they even provide solutions with accuracy approaching 1.0 while the other schemes fail to find any solution, e.g., pedigree7, i=6.

WCSPs (Figure 2): wAOBF is mostly inferior. The wAOBB and wBRAOBB perform better than BRAOBB on many instances (e.g., capmo2.wcsp for all i-bounds), except for a number of problems (not shown), for which only BRAOBB is the only scheme to return solutions.

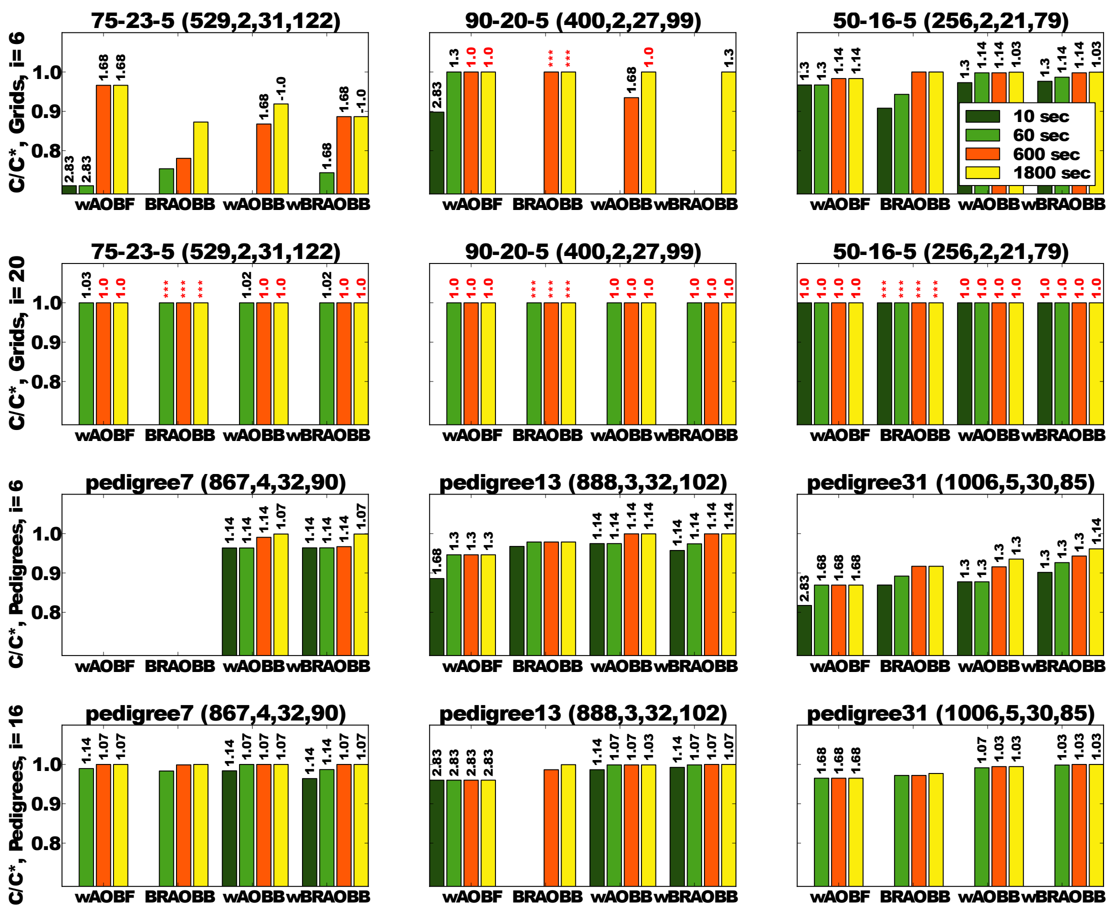

Figure 1: Ratio of the cost obtained by some time point (10, 60, 600 and 1800 sec) and max cost. Max. cost = optimal, if known, otherwise = best cost found for the problem. Corresponding weight - above the bars. The cases where BRAOBB proved solution optimality is indicated by '***' above bars. In red - optimal solutions. Instance parameters are in format (n,k,w^*,h_T), where n - number of variables, k - max. domain size, w^* - induced width, h_T - pseudo-tree height. Grids and Pedigrees. Memory limit 4 GB, time limit 1 hour.

Type4 (Figure 2): wAOBF mostly dominates over all Branch and Bound schemes. However, for larger i-bounds wAOBB and wBRAOBB perform well, sometimes providing tighter suboptimality guarantees or report solutions in less time than wAOBF. For example, for type4b_170_23, i=12, wAOBB and wBRAOBB return solutions within 10 seconds, while wAOBF does not.

We conclude that weighted Branch and Bound schemes are competitive and on 2 out of 4 benchmarks (Pedigrees and WCSPs) they are often superior.

The Impact of Heuristic Strength

Figures 1 and 2 illustrate that in many cases the weighted schemes tend to be superior to BRAOBB when there is a significant distance between the i-bound which characterizes the heuristic strength and the problem's induced width. For example, compare pedigree7, i=6 and i=16.

One explanation is that the weight may make the weak admissible heuristic more accurate (a weak lower bound becomes stronger lower bound, closer to the actual optimal cost, when multiplied by a constant).

For higher i-bounds there is typically little difference between the costs of the solutions, which are often exact (e.g. all pedigree instances, i=20).

When both of the presented i-bounds values are far from the induced width (e.g. type4_140_20, $w^* = 29$, i-bounds 6 and 12), the weighted Branch and Bound schemes benefit from the improved heuristics, as do the other schemes.

Summary

Figure 3 presents a different, more inclusive view of the data. It shows the scatterplots comparing wAOBF and wAOBB schemes. Each scatterplot corresponds to a single benchmark, for time 600 seconds and for a particular typical i-bound. Since we are only interested in relative performance comparison, we do not take into account the heuristic computation time, which is the same for all schemes. Values equal to 1.0 depict better performance. Each marker corresponds to a single instance.

In Figure 3 shows that on Grid and Type4 benchmarks

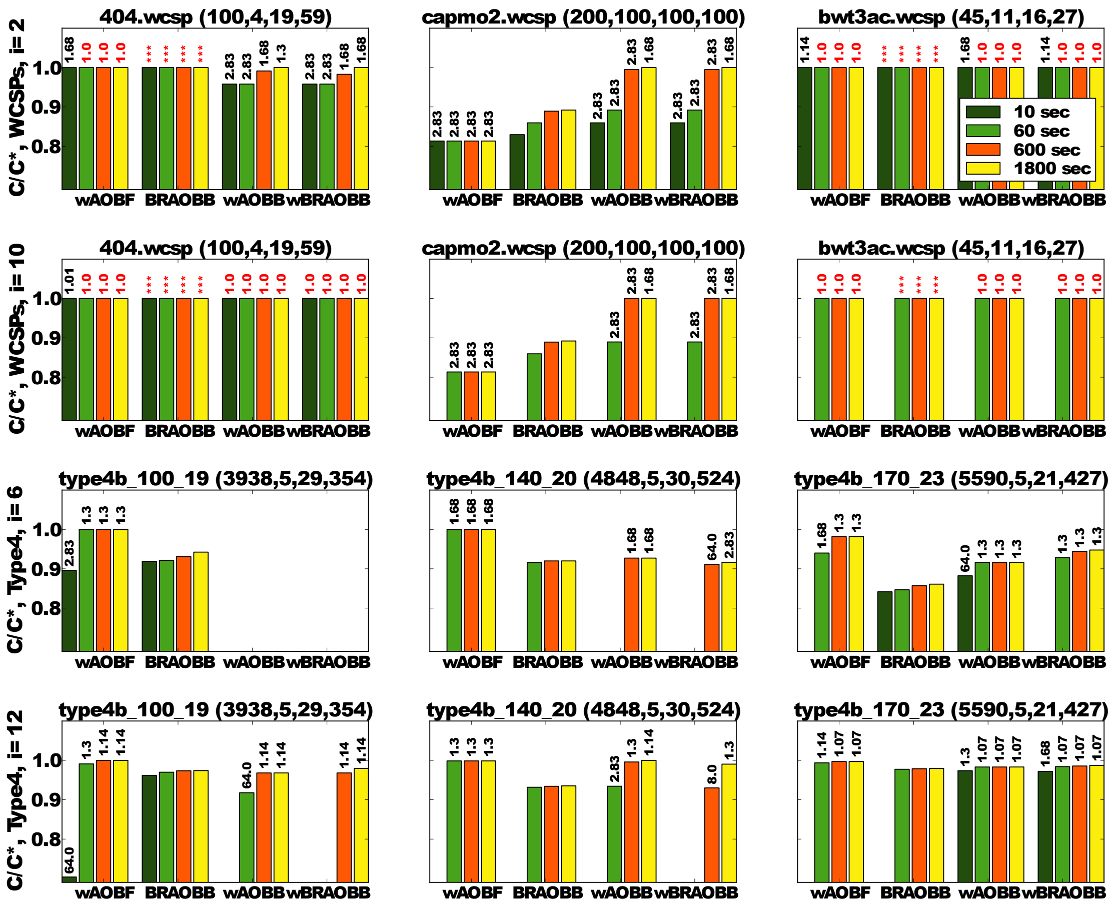

Figure 2: Ratio of the cost obtained by some time point (10, 60, 600 and 1800 sec) and max cost. Max. cost = optimal, if known, otherwise = best cost found for the problem. Corresponding weight - above the bars. The cases where BRAOBB proved solution optimality is indicated by '***' above bars. In red - optimal solutions. Instance parameters are in format (n,k,w^*,h_T), where n - number of variables, k - max. domain size, w^* - induced width, h_T - pseudo-tree height. Type4 and WCSPs. Memory limit 4 GB, time limit 1 hour.

wAOBF is often superior, as indicated by the many markers laying above the red diagonal. On Pedigrees wAOBB is slightly better, and on WCSP wAOBB really shines, finding visibly better solutions for the majority of instances.

The comparison between both of the weighted Branch and Bound schemes and BRAOBB is presented in a summarizing Table 4. Similar to Table 2 it shows the percentage of problems for which wAOBB and wBRAOBB are superior to BRAOBB, percentage, for which they are tied, and the total number of problems for which a solution was found. We also show results for wAOBF for comparison.

The results in Table 4 agree with observations we made earlier:

- The weighted BB schemes are sometimes considerably superior to BRAOBB (e.g., wAOBB better on 44% of problems on Grids, i=6, 60 sec; wBRAOBB is better on 70% on Pedigrees, i=6, 3600 sec).
- Unlike wAOBF, there is no obvious relation between the time limit and the quality of weighted Branch and Bound schemes' performance. Sometimes they are more sucessful for short time bounds (e.g. for WCSPs, i=6, wAOBB is better than BRAOBB on 22.9% of problems for 60 sec, but only on 16.3% for 3600 sec), but it is not always the case.
- On Grids and Pedigrees wAOBB and wBRAOBB clearly are better than BRAOBB on more instances when the heuristic is weak. However, for Type4 and WCSPs larger i-bounds yield better performance for most time limits. Strong heuristics might be more crucial for these benchmarks, since they are known to be harder than Grids and Pedigrees.

Conclusion

The paper provides the first study of weighted depth-first search over graphical models. We compared the new algorithms with one of the most competitive Branch and Bound schemes and with weighted best-first search. Our results demonstrated that the weighted Branch and Bound algorithms can be effective as anytime schemes, having the

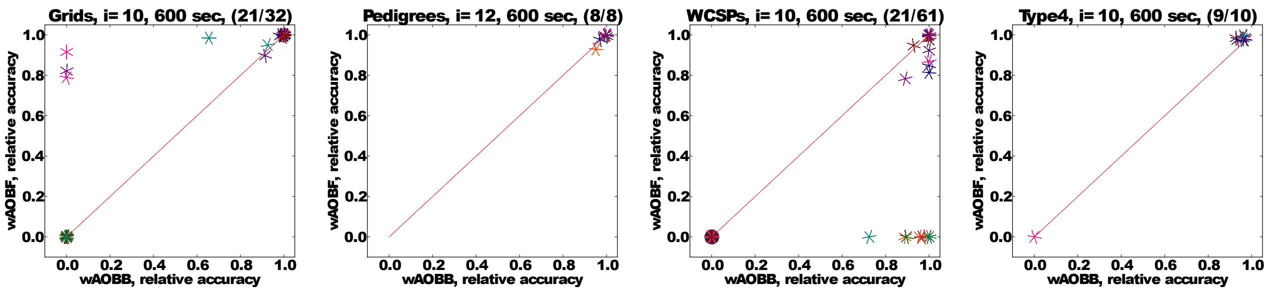

Figure 3: wAOBB vs wAOBF: comparison of relative accuracy for 600 sec. Each marker is a single instance. Memory limit 4 GB, time limit 1 hour. In parenthesis (X/Y): X - # instances, for which at least one algorithm found a solution, Y - total # instances.

I-bound	Algorithm	**Time bounds**		
		60	600	3600
		X% / Y% / N	X% / Y% / N	X% / Y% / N
Grids (# inst=32, n=144-2500, k=2, w^*=15-90, h_T=48-283)				
i=6	wAOBF	68.0 / 20.0 / 25	40.0 / 40.0 / 25	16.0 / 44.0 / 25
	wAOBB	44.0 / 12.0 / 25	32.0 / 32.0 / 25	28.0 / 44.0 / 25
	wBRAOBB	36.0 / 12.0 / 25	28.0 / 28.0 / 25	24.0 / 44.0 / 25
i=18	wAOBF	16.7 / 66.7 / 24	3.8 / 80.8 / 26	0.0 / 75.0 / 28
	wAOBB	16.7 / 62.5 / 24	7.4 / 77.8 / 27	3.6 / 82.1 / 28
	wBRAOBB	12.5 / 62.5 / 24	7.4 / 77.8 / 27	0.0 / 82.1 / 28
Pedigrees (# inst=11, n=581-1006, k=3-7, w^*=16-39, h_T=52-104)				
i=6	wAOBF	44.4 / 33.3 / 9	22.2 / 33.3 / 9	22.2 / 22.2 / 9
	wAOBB	30.0 / 10.0 / 10	40.0 / 10.0 / 10	60.0 / 20.0 / 10
	wBRAOBB	30.0 / 20.0 / 10	60.0 / 30.0 / 10	70.0 / 30.0 / 10
i=20	wAOBF	0.0 / 50.0 / 2	20.0 / 20.0 / 5	0.0 / 40.0 / 5
	wAOBB	50.0 / 0.0 / 2	20.0 / 80.0 / 5	0.0 / 80.0 / 5
	wBRAOBB	50.0 / 50.0 / 2	20.0 / 80.0 / 5	0.0 / 80.0 / 5
WCSP (# inst=56, n=25-1057, k=2-100, w^*=5-287, h_T=11-337)				
i=6	wAOBF	2.9 / 20.0 / 35	0.0 / 20.0 / 40	0.0 / 19.0 / 42
	wAOBB	17.1 / 25.7 / 35	14.6 / 36.6 / 41	14.0 / 34.9 / 43
	wBRAOBB	22.9 / 28.6 / 35	22.0 / 36.6 / 41	16.3 / 37.2 / 43
i=14	wAOBF	0.0 / 66.7 / 3	0.0 / 42.9 / 7	0.0 / 50.0 / 8
	wAOBB	0.0 / 66.7 / 3	42.9 / 42.9 / 7	37.5 / 50.0 / 8
	wBRAOBB	0.0 / 66.7 / 3	42.9 / 42.9 / 7	37.5 / 50.0 / 8
Type4 (# inst=10, n=3907-8186, k=5, w^*=21-32, h_T=319-625)				
i=6	wAOBF	80.0 / 0.0 / 10	90.0 / 0.0 / 10	90.0 / 0.0 / 10
	wAOBB	20.0 / 0.0 / 10	50.0 / 0.0 / 10	50.0 / 0.0 / 10
	wBRAOBB	10.0 / 0.0 / 10	20.0 / 0.0 / 10	50.0 / 0.0 / 10
i=18	wAOBF	100.0 / 0.0 / 1	100.0 / 0.0 / 9	88.9 / 11.1 / 9
	wAOBB	100.0 / 0.0 / 1	88.9 / 0.0 / 9	88.9 / 0.0 / 9
	wBRAOBB	100.0 / 0.0 / 1	88.9 / 0.0 / 9	88.9 / 0.0 / 9

Table 4: X% - percentage of instances for which each algorithm is the better than BRAOBB at a specific time bound, Y% - percentage of instances for which algorithm ties with BRAOBB, N - number of instances for which at least one of algorithms found a solution. # inst - total number of instances in benchmark, n - number of variables, k - maximum domain size, w^* - induced width, h_T - pseudo-tree height. 4 GB memory, 1 hour time limit.

.

added benefit of w-optimal solution, which are sometime tighter than those by weighted best-first search.

Clearly, however, due to the fact that no algorithm is always superior, the question of algorithm selection requires further investigation. We aim to identify problem features that could be used to predict which scheme is best suited for solving a particular instance. We also plan to automate the algorithm parameter selection based on benchmarks or problems, as well as enrich our set of schemes by considering, for example, dynamic weights.

An alternative to selecting a single scheme for a specific problem is combining our algorithms within a portfolio framework, similar to e.g., SATzilla [(Xu et al. 2008)] and PbP [(Gerevini, Saetti, and Vallati 2009)]. The question of portfolio building and scheduling also will be studied in the future.

Acknowledgement

This work was sponsored in part by NSF grants IIS-1065618 and IIS-1254071, and by the United States Air Force under Contract No. FA8750-14-C-0011 under the DARPA PPAML program.

References

Chakrabarti, P.; Ghose, S.; and De Sarkar, S. 1987. Admissibility of AO* when heuristics overestimate. *Artificial Intelligence* 34(1):97–113.

de Givry, S.; Schiex, T.; and Verfaillie, G. 2006. Exploiting tree decomposition and soft local consistency in weighted csp. In *AAAI*, 22–27.

Dechter, R., and Mateescu, R. 2007. AND/OR search spaces for graphical models. *Artificial Intelligence* 171(2-3):73–106.

Dechter, R., and Rish, I. 2003. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM* 50(2):107–153.

Flerova, N.; Marinescu, R.; and Dechter, R. 2014a. Weighted anytime search: new schemes for optimization over graphical models. Report R210:http://www.ics.uci.edu/ dechter/publications/r210.pdf.

Flerova, N.; Marinescu, R.; and Dechter, R. 2014b. Weighted best first search for map. In *The International Symposium on Artificial Intelligence and Mathematics 2014*.

Gerevini, A.; Saetti, A.; and Vallati, M. 2009. An automatically configurable portfolio-based planner with macro-actions: Pbp. In *ICAPS*.

Hansen, E., and Zhou, R. 2007. Anytime heuristic search. *Journal of Artificial Intelligence Research* 28(1):267–297.

Hart, P.; Nilsson, N.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans on Systems Science and Cybernetics* 4(2):100–107.

Huberman, B. A.; Lukose, R. M.; and Hogg, T. 1997. An economics approach to hard computational problems. *Science* 275(5296):51–54.

Kask, K., and Dechter, R. 1999. Mini-bucket heuristics for improved search. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, 314–323. Morgan Kaufmann Publishers Inc.

Kask, K., and Dechter, R. 2001. A general scheme for automatic search heuristics from specification dependencies. *Artificial Intelligence* 129(1–2):91–131.

Likhachev, M.; Gordon, G.; and Thrun, S. 2003. ARA*: Anytime A* with provable bounds on sub-optimality. *NIPS* 16.

Marinescu, R., and Dechter, R. 2009a. AND/OR Branch-and-Bound search for combinatorial optimization in graphical models. *Artificial Intelligence* 173(16-17):1457–1491.

Marinescu, R., and Dechter, R. 2009b. Memory intensive AND/OR search for combinatorial optimization in graphical models. *Artificial Intelligence* 173(16-17):1492–1524.

Nillson, N. J. 1980. *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA.

Otten, L., and Dechter, R. 2011. Anytime AND/OR depth first search for combinatorial optimization. In *SOCS*.

Pohl, I. 1970. Heuristic search viewed as path finding in a graph. *Artif. Intell.* 1(3-4):193–204.

Richter, S.; Thayer, J.; and Ruml, W. 2010. The joy of forgetting: Faster anytime search via restarting. In *ICAPS*, 137–144.

van den Berg, J.; Shah, R.; Huang, A.; and Goldberg, K. 2011. Anytime nonparametric A*. In *AAAI*.

Xu, L.; Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2008. Satzilla: Portfolio-based algorithm selection for sat. *J. Artif. Intell. Res.(JAIR)* 32:565–606.