Weighted Best-First Search for W-Optimal Solutions over Graphical Models

Natalia Flerova
University of California
Irvine, USA

Radu Marinescu
IBM Research
Dublin, Ireland

Pratyaksh Sharma
Indian Institute of Technology
Bombay, India

Rina Dechter
University of California
Irvine, USA

Abstract

The paper explores the potential of weighted best-first search schemes as anytime optimization algorithms for solving graphical models tasks such as MPE (Most Probable Explanation) or MAP (Maximum a Posteriori) and WCSP (Weighted Constraint Satisfaction Problem). While such schemes were widely investigated for path-finding tasks, their application for graphical models was largely ignored, possibly due to their memory requirements. Compared to the depth-first branch and bound, which has long been the algorithm of choice for optimization in graphical models, a valuable virtue of weighted best-first search is that they are w-optimal, i.e. when terminated, they return a solution cost C and a weight w, such that $C \leq w \cdot C^*$, where C^* is the optimal cost. We report on a significant empirical evaluation, demonstrating the usefulness of weighted best-first search as approximation anytime schemes (that have suboptimality bounds) and compare against one of the best depth-first branch and bound solver to date. We also investigate the impact of different heuristic functions on the behaviour of the algorithms.

Introduction

The most common search scheme for combinatorial optimization tasks over graphical models, such as MAP/MPE or Weighted CSP, is depth-first branch and bound, extensively studied in recent years (Kask and Dechter 2001; Marinescu and Dechter 2009b; Otten and Dechter 2011; de Givry, Schiex, and Verfaillie 2006). Meanwhile, best-first search algorithms, though known to bound the search space more effectively (Dechter and Pearl 1985), are seldom considered for this domain due to their inability to provide any solution before termination and inherently large memory requirements. Furthermore, one of best-first's most attractive features, avoiding the exploration of unbounded paths, seems irrelevant since solutions are of equal depth (i.e., the number of variables).

In contrast, in path-finding domains, where solution length varies (e.g., planning), best-first search and especially its popular variant A* (Hart, Nilsson, and Raphael 1968) is clearly favoured. However, A*'s exponential memory needs, coupled with lack of anytime behaviour, lead to extension into more flexible anytime schemes based on the *Weighted A** (WA*) (Pohl 1970). The main idea lies in inflating the heuristic function guiding the search by a factor of $w > 1$, which makes the heuristic inadmissible and typically yields faster search, while still guaranteeing a solution cost within a factor w from the optimal. If the (non-optimal) solution is found before reaching the time limit, the search for a better solution may resume. In the context of path-finding in the past decade several anytime weighted best-first search schemes were proposed (Likhachev, Gordon, and Thrun 2003; Hansen and Zhou 2007; Richter, Thayer, and Ruml 2010; van den Berg et al. 2011). These algorithms are able to output a suboptimal solution fairly quickly and, given additional time, improve the accuracy of the solution, ultimately finding the optimal one.

Our contribution is in extending the above methods to graphical models and in investigating their potential empirically. As a basis we used AND/OR best-first search (AOBF) (Marinescu and Dechter 2009b), a best-first algorithm developed for AND/OR search spaces over graphical models. AOBF explores the context minimal AND/OR graph in a best-first manner, guided by admissible and consistent heuristic (Dechter and Mateescu 2007). We considered heuristics obtained by three algorithms: Mini-Bucket Elimination (Kask and Dechter 2001), Mini-Bucket Elimination with Moment-Matching and Joint Graph Linear Programming (Ihler et al. 2012).

After exploring a variety of approaches and following extensive empirical analysis, including two non-parametric algorithms that interleave depth- and best-first exploration, the two schemes that emerged as most promising were wAOBF and wR-AOBF. Both are running Weighted A* iteratively while decreasing w. Algorithm wAOBF starts from scratch at each iteration, while wR-AOBF reuses search efforts from previous iterations, extending ideas of Anytime Repairing A* (ARA*) (Likhachev, Gordon, and Thrun 2003).

We report on a comprehensive empirical evaluation of the two candidate algorithms on around 100 instances from 4 different benchmarks, evaluating their performance both as approximation and anytime schemes with multiple heuristic strengths. We compared against Breadth-Rotating AND/OR Branch-and-Bound (BRAOBB) (Otten and Dechter 2011), a state-of-the-art anytime depth-first branch and bound which

Copyright © 2015, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

won the 2011 Probabilistic Inference Challenge[1] in all optimization categories.

Our empirical analysis revealed that weighted best-first search algorithms outperform BRAOBB on many instances from 2 of our benchmarks and are comparable on the majority of problems from another one. Most importantly, they provide suboptimality guarantees and, overall, should be considered as candidate algorithms for solving optimization over graphical models alongside BRAOBB, in example, in a portfolio scheme (Huberman, Lukose, and Hogg 1997).

Background

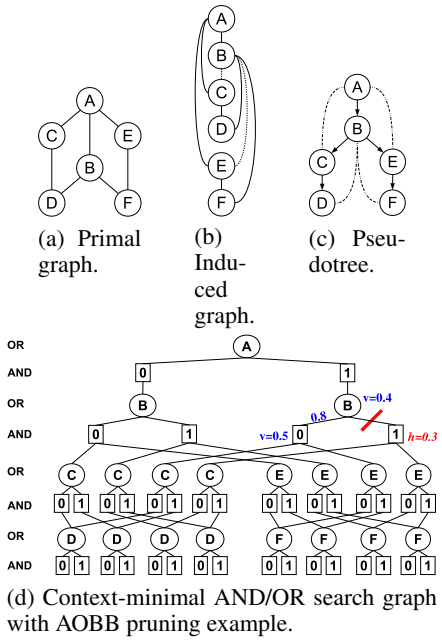
Weighted search exploits the idea of making an admissible evaluation function (i.e. one that never overestimates a cost to the goal, assuming minimization) inadmissible, by multiplying it by some weight $w > 1$. In particular, the most well-known variant of weighted best-first search, *WA**, uses evaluation 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 $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, where $\mathbf{X} = \{X_i : i \in V\}$ is a set of variables indexed by set V and $\mathbf{D} = \{D_i : i \in V\}$ is the set of their finite domains of values. $\mathbf{F} = \{f_{\mathbf{S}_1}, \ldots, f_{\mathbf{S}_r}\}$ is a set of non-negative real-valued functions defined on subsets of variables $\mathbf{x}_{\mathbf{S}_i} \subseteq \mathbf{X}$, called *scopes* (i.e., $\forall i \; f_i : \mathbf{x}_{\mathbf{S}_i} \rightarrow \mathbb{R}^+$) The set of function scopes imply a *primal graph* (e.g., Figure 1a), whose vertices are the variables and which includes an edge connecting any two variables that appear in the scope of the same function. Given an ordering of the variables, the set of function scopes yields an *induced graph*, where each node's earlier neighbours are connected from last to first, (e.g., Figure 1b) with a certain *induced width* w^*. For detail see, e.g. (Kask et al. 2005).

The common optimization task is to find $\max_{\mathbf{X}} \prod_i \psi_i$ called **MAP** (Maximum a Posteriori) or **MPE** (Most Probable Explanation) and $\min_{\mathbf{X}} \sum_i \psi_i$, called **WCSP** (Weighted Constraint Satisfaction Problem). These two tasks are equivalent and are easily transformed into one another. In our discussion of algorithms we assume the min-sum task, as is traditional in heuristic search literature.

The AND/OR search space is defined using a *pseudotree* of the primal graph $G = (X, E)$ (e.g., Figure 1c), which captures problem decomposition. A *pseudo-tree* of an undirected graph G is a directed rooted tree $\mathcal{T} = (X, E')$, such that every arc of G not included in E' is a back-arc in $\mathcal{T}$, namely it connects a node in $\mathcal{T}$ to its ancestor. The arcs in

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



(a) Primal graph. (b) Induced graph. (c) Pseudotree.

(d) Context-minimal AND/OR search graph with AOBB pruning example.

Figure 1: Example problem with six variables, induced graph along ordering A, B, C, D, E, F, corresponding pseudotree, and resulting AND/OR search graph with AOBB pruning example.

E' might not necessarily all be included in the set of original edges E of graph G.

AND/OR search tree associated with a graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ with a primal graph G and pseudo tree $\mathcal{T}$ of G, consists of alternating levels of OR and AND nodes, guided by the pseudotree structure. OR nodes correspond to the variables, AND nodes correspond to the values of the OR parent's variable, rooting conditionally independent subproblems. The edges of the AND/OR tree are annotated by values derived from the input functions $\mathbf{F}$.

Graph-based merging of identical subproblems yields the *context minimal AND/OR search graph* $C_{\mathcal{T}}$ (e.g. Figure 1d) which has size $O(N \cdot k^{w^*})$, where w^* is the induced width of G along a depth first order traversal of $\mathcal{T}$, N is the number of variables and k bounds the domain sizes (Dechter and Mateescu 2007).

A *solution tree* T^* of $C_{\mathcal{T}}$ is a subtree that: (1) contains the root of $C_{\mathcal{T}}$; (2) if an internal OR node $n \in C_{\mathcal{T}}$ is in T^*, then n is labelled by a variable and exactly one of its children is in T^*; (3) if an internal AND node $n \in C_{\mathcal{T}}$ is in T^*, then all its OR children labelled by variables are in T^*.

AND/OR Best First Search (AOBF) (Marinescu and Dechter 2009a) 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 the usual A*, AOBF does not have explicit OPEN and CLOSED lists, instead maintaining a graph denoted $\mathcal{G}$, which is the explicated part of the context minimal AND/OR search graph $\mathcal{S}_{\mathcal{T}}$, and keeping track of the current best partial solution tree T^*. AOBF

Algorithm 1: wAOBF(w_0, 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 obtained with i-bound i; initial weight w_0

Output: Set of suboptimal solutions $\mathcal{C}$

1 Initialize $w = w_0$, weight update schedule S and let $\mathcal{C} \leftarrow \emptyset$;
2 **while** $w >= 1$ **do**
3 $\quad \langle C_w, T^*_w \rangle \leftarrow$ AOBF($w \cdot h$), where T^*_w - full solution tree;
4 $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{\langle w, C_w, T^*_w \rangle\}$;
5 $\quad$ Decrease weight w according to schedule S;
6 **return** $\mathcal{C}$;

interleaves a top-down step of expanding the nodes in the best-first manner with bottom-up update of nodes' values and recalculation of T^*.

AND/OR Branch and Bound (AOBB) (Marinescu and Dechter 2009a): AOBB traverses the 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 is 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) was developed to remedy poor anytime performance of AOBB due to AND/OR decomposition. At each AND node all but one independent child subproblems have to be solved completely, before the last one is even considered by AOBB. BRAOBB remedies this deficiency, rotating through different subproblems in a breadth-first manner. It empirically proved to be a far more efficient anytime algorithm than plain AOBB.

Typically all the AND/OR search algorithms utilize the mini-bucket heuristic, which is admissible and consistent (Kask and Dechter 1999). *Mini-Bucket Elimination* (MBE) (Dechter and Rish 2003) is an approximation version of an exact variable elimination algorithm called bucket elimination (Dechter 1999). MBE outputs an upper bound on the optimal MPE value (lower bound on the optimal WCSP value). The complexity of the algorithm, which is parametrized by the i-bound i, is time and space exponential in i. Larger i-bound typically yields tighter bound, and thus allows to flexibly trade complexity and accuracy.

Weighted Best-First Search for Graphical Models

Iterative Weighted AOBF (wAOBF): AND/OR Best-First search using the 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)). A natural extension of the algorithm to an anytime scheme (**wAOBF** in Algorithm 1), executes Weighted AOBF iteratively, decreasing the weight at each iteration. This approach, similar to the Restarting Weighted A* by (Richter, Thayer, and Ruml 2010), results in a series of solutions, each with a smaller suboptimality factor of the weight w.

Algorithm 2: wR-AOBF(w_0, 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 obtained with i-bound i; initial weight w_0

Output: A set of suboptimal solutions $\mathcal{C}$

1 Initialize $w = w_0$, weight update schedule S and let $\mathcal{C} \leftarrow \emptyset$;
2 Create root OR node s labelled by X_1 and let $\mathcal{G} = \{s\}$ (currently explored part of the search space);
3 Initialize value $v(s) = w \cdot h(s)$ and current best partial solution tree T^* to $\mathcal{G}$;
4 **while** $w \geq 1$ *and time* $\leq$ *time bound* **do**
5 $\quad$ **while** T^* *has more tip nodes* **do**
6 $\quad\quad$ Expand and update nodes in $\mathcal{G}$ using AOBF search with heuristic function ($w \cdot h$);
7 $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{\langle w, v(s), T^* \rangle\}$;
8 $\quad$ Decrease weight w according to schedule S;
9 $\quad$ For all leaf nodes in $n \in \mathcal{G}$, update $v(n) = w \cdot h(n)$. Update the values of all nodes in $\mathcal{G}$ using the values of their successors. Mark best successor of each OR node.;
10 $\quad$ Recalculate T^* following the marked arcs;
11 **return** $\mathcal{C}$;

Anytime Repairing AOBF (wR-AOBF): Running each search iteration from scratch, as wAOBF does, is wasteful. Anytime Repairing AOBF (**wR-AOBF** in Algorithm 2) records and re-uses information between iterations. It adapts *Anytime Repairing A* (ARA*)* algorithm (Likhachev, Gordon, and Thrun 2003) to AND/OR search spaces over graphical models. The original ARA* algorithm utilizes the results of previous steps by recomputing the evaluation functions of the nodes with each weight change, re-using inherited OPEN and CLOSED lists and by keeping track of already expanded nodes, whose evaluation function changed between iterations and re-inserting them back to OPEN list before starting a new iteration.

Since AOBF does not maintain explicit OPEN and CLOSED lists, *wR-AOBF* keeps track of $\mathcal{G}$, the partially explored AND/OR search graph. At each iteration, until a new complete solution is found (line 5) the algorithm expands the graph $\mathcal{G}$ in the usual best-first manner (line 6): the tip node of the current best solution tree T^* having the smallest value of the evaluation function f is expanded and its children are added to $\mathcal{G}$. Once a solution is found, it is reported along with the corresponding weight (line 7) and the weight is decreased (line 8). After each weight update wR-AOBF performs a bottom-up update of all the node values starting from the leaf nodes (whose h-values are multiplied by the new weight) and continuing towards the root node (line 9). During this phase, the algorithm also marks the best AND successor of each OR node in the search graph. These markings are used to recompute the best partial solution tree T^* (line 10). Then, the search resumes in the usual manner by expanding a tip node of T^*.

Like ARA*, wR-AOBF is guaranteed to terminate with a solution cost C such that $C \leq w \cdot C^*$, where C^* is the optimal solutions cost.

Benchmark	# inst	N	k	w^*	h_T
Pedigrees	11	581-1006	3-7	16-39	52-102
Grids	32	144-2500	2-2	15-90	48-283
Type4	10	3907-8186	5-5	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.

Instance	BRAOBB	Weighted AOBF weights 2.00	1.20	1.10	1.00
	time log-cost	time log-cost	time log-cost	time log-cost	time log-cost
Grids	I-bound=18				
75-26-5 (676, 2, 36, 129)	23.55 -50.404	0.19 -53.707	0.23 -53.485	1.0 -50.654	10.04 -50.404
90-34-5 (1156, 2, 51, 175)	79.39 -30.598	1.0 -33.375	1.54 -30.636	9.74 -30.636	— —
Pedigrees	I-bound=14				
pedigree19 (693, 5, 21, 107)	3841.07 -223.56	0.04 -235.061	0.05 -225.397	0.39 -223.59	— —
pedigree7 (867, 4, 28, 140)	3289.57 -262.238	0.07 -275.693	0.14 -266.909	0.24 -264.712	— —
WCSP	I-bound=10				
404.wcsp (100, 4, 19, 59)	6.78 -6.402	0.01 -6.627	0.03 -6.402	0.18 -6.402	1.92 -6.402
myciel5g_3.wcsp (47, 3, 19, 24)	15.76 -147.365	33.85 -156.576	— —	— —	— —
Type4	I-bound=16				
type4b_100_19 (3938, 5, 29, 354)	OOT -2598.003	0.95 -2638.441	0.65 -2587.936	0.85 -2587.936	— —
type4b_120_17 (4072, 5, 24, 319)	OOT -3061.944	0.41 -3100.952	0.46 -3067.778	0.45 -3062.157	— —

Table 2: Runtime (sec) and log-cost obtained by weighted AOBF(w, i) for selected w, and by BRAOBB, both with MBE-MM heuristic. OOT for BRAOBB indicates terminating without proving solution optimality. Instance parameters (N,k,w^*,h_T): N - number of variables, k - max domain size, w^* - induced width, h_T - pseudo tree height. 4 GB memory, 1 hour time limit.

Experiments

We evaluate the performances of wAOBF, wR-AOBF and compare with BRAOBB exploring the same context minimal AND/OR graph. The search space is determined by a common variable ordering that determines the pseudo-tree. The algorithms output solutions at different times until either the optimal solution is found or the time limit of 1 hour is reached, or the scheme runs out of memory (4 GB).

The experimental results reported were a final phase of an extensive evaluation that we carried out in the past 2 years. These include hundreds of instances from 8 benchmarks from UAI 2008[2] and Pascal 2011[3] competitions, currently the main source of benchmarks in the community.

We chose to present results on 4 benchmarks due to space considerations and because the results were more meaningful and illustrative. These instances include probabilistic graphical models and weighted CSPs. Most instances that we do not report were either trivially easy (solved exactly under 5 seconds by all schemes) or too hard (no solutions or only a single suboptimal solution by any scheme within the time bound).

In random grid networks the nodes are arranged in an N by N square and each conditional probability table is generated uniformly randomly. The pedigree and type4 instances come from the domain of genetic linkage analysis, more specifically haplotyping. The Weighted CSP networks benchmark comprises of graph colouring problems, SPOT5 networks and other domains. Table 1 shows the following benchmark parameters: # inst - number of instances, N - number of variables, k - domain size, w^* - induced width, h_T - pseudotree height. The pseudo trees that guide the AND/OR search algorithms were obtained using the depth-first traversal along the variable ordering generated using MinFill heuristic (Kjaerulff 1990).

For uniformity we assume that the optimization task is maximization throughout. After considering 5 different weight policies with various parameters, we settled for both wAOBF and wR-AOBF on the the following: $w_i = k \cdot \sqrt{w_{i-1}}$, where w_i is the weight at iteration i and k is a constant factor empirically chosen to be equal to 4. The starting weight w_0 was selected to be 64: a) to explore the schemes' behaviour on a large range of weights; b) to make the search greedy enough initially to solve harder instances known to be infeasible for regular AOBF within the memory limit.

All algorithms used admissible and consistent heuristic functions obtained by three schemes: Mini-Bucket Elimination (MBE) (Kask and Dechter 1999) and its two advanced versions: MBE with Moment-Matching (MBE-MM) and Joint Graph Linear Programming (JGLP) (Ihler et al. 2012). MBE and MBE-MM are single-pass algorithms, while JGLP is an iterative scheme that we chose to run for 60 seconds, based on preliminary experiments. The accuracy of all heuristics 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). We solved each problem with 8 values of i-bound, ranging from 6 to 20.

In our empirical evaluation we address:

1. The effectiveness of weighted best-first schemes as approximations
2. The quality of schemes' anytime behaviour
3. The interaction between heuristic strength and the multiplicative weight

Effectiveness of Weighted AOBF as Approximation

We first illustrate the impact of the weighted heuristic on the performance of best-first search. We focus on the results obtained using MBE-MM heuristic, since it is the most versatile of the three and proved to provide reasonably accurate bounds while keeping pre-processing time short.

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

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

In **Table 2** we report runtime (sec) and logarithm of solution cost for Weighted AOBF, for 2 instances from each benchmark. We also report the run time and the log-cost by BRAOBB at time of termination. Log-cost closer to 0 is better. OOT signifies that BRAOBB terminated due to timeout without proving solution optimality. We defer discussing BRAOBB as an anytime scheme till the next section. Interesting results are framed. We report results for relatively high i-bounds.

Time saving for w-bounded suboptimality. It is interesting to compare first the results by BRAOBB (column 2) and by Weighted AOBF (column 6, when $w = 1$) against any of the other columns. Each weighted column represents a particular level of guaranteed suboptimality, corresponding to the values of the weights that we used when running a single iteration of Weighted AOBF. For example, consider the column of weight $w = 1.10$. These results are guaranteed to be just a factor of 1.1 away from optimal, yet the time savings compared with BRAOBB are significant (e.g., pedigree19: 0.39 vs 3841.07 sec or type4b_100_19: Weighted AOBF terminated in 0.85 sec, while BRAOBB finished without proving solution optimality and giving any guarantees within the time limit). The solution cost found by Weighted BF is often close to or (for Type4) better than one by BRAOBB.

Effectiveness of Weighted AND/OR Best-First Algorithms as Anytime Schemes

Figure 2 displays the anytime behaviour of the schemes for typical instances, chosen to best illustrate predominant trends for each benchmark. For each instance we show the ratio between the cost available at a particular time point (at 5, 10, 60 and 600 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 1.1 only. Each row corresponds to a particular benchmark, for a medium strength i-bound. The four leftmost bars (for 4 different time points) correspond to wAOBF, the central ones - to wR-AOBF and the four rightmost - to BRAOBB.

We display the weight at the time bound above the respective bar for wAOBF and wR-AOBF. For BRAOBB '***' indicates that solution optimality has been proven. MBE-MM heuristic was used.

Figure 2 demonstrates that weighted best-first schemes are often superior on Grid instances. They often find solutions of high accuracy faster than BRAOBB, as, for example, in case of instance 90-24-5, where both wAOBF and wR-AOBF report solutions with accuracy greater than 0.9 within 5 seconds, while BRAOBB provides none until 600 seconds.

On certain Pedigree problems weighted best-first schemes also perform well, e.g. pedigree7, where both wAOBF and wR-AOBF, unlike BRAOBB, manage to find a solution of accuracy higher than 0.7 within 5 seconds. However, on most Pedigrees BRAOBB is superior, e.g. pedigree30.

On WCSPs BRAOBB clearly dominates, since on many problems (e.g. 1403.wcsp and 1504.wcsp) neither wAOBF nor wR-AOBF report within the time limit any solutions with accuracy greater than our threshold of 0.7. They are only competitive on some easier instances, such as 404.wcsp or 505.wcsp.

On Type4 benchmark weighted best-first algorithms typically find reasonably accurate solutions much faster than BRAOBB. For example, on type4b_30_21 wR-AOBF and wAOBF report solutions with accuracy greater than 0.9 within 60 seconds and BRAOBB does not, up until 600 seconds.

Significantly, for most benchmarks the weighted BF schemes often provide tight w-optimality bounds. It is especially valuable whenever BRAOBB does not prove solution optimality within the time limit, as is the case for all Type4 instances shown. Then the accuracy of the BRAOBB costs remains unknown, unless we obtain the ground truth, i.e. exact solution, by other means. At the same time, weighted BF schemes provide w-optimality guarantees. For example, on the four Type4 instances presented wR-AOBF guarantees that the costs reported by 600 seconds are just within factor of 1.1 from the optimal.

Tables 3 and 4 present a summary of the results. For 2

I-bound	Heuristic	Algorithm	Time bounds	
			120	3600
			X%(Y%) / W / #	X%(Y%) / W / #
Grids (# inst=15, N=144-2500, k=2, w^*=15-90, h_T=48-283)				
i=6	MBE	wAOBF	50.0 (25.0) / 4.0 / 12	25.0 (50.0) / 1.0 / 12
		wR-AOBF	58.3 (16.7) / 1.19 / 12	0.0 (41.7) / 1.19 / 12
	MBE-MM	wAOBF	33.3 (58.3) / 1.0 / 12	0.0 (66.7) / 1.0 / 12
		wR-AOBF	33.3 (58.3) / 1.07 / 12	8.3 (66.7) / 1.06 / 12
	JGLP	wAOBF	7.1 (85.7) / 3.0 / 14	0.0 (92.9) / 1.0 / 14
		wR-AOBF	0.0 (85.7) / 1.01 / 14	0.0 (92.9) / 1.01 / 14
i=16	MBE	wAOBF	0.0 (81.8) / 2.0 / 11	0.0 (75.0) / 1.0 / 12
		wR-AOBF	0.0 (81.8) / 1.05 / 11	0.0 (75.0) / 1.05 / 12
	MBE-MM	wAOBF	11.1 (88.9) / 3.0 / 9	11.1 (88.9) / 1.0 / 9
		wR-AOBF	0.0 (88.9) / 1.03 / 9	0.0 (88.9) / 1.03 / 9
	JGLP	wAOBF	0.0 (77.8) / 2.0 / 9	0.0 (88.9) / 1.0 / 9
		wR-AOBF	0.0 (88.9) / 1.01 / 9	0.0 (88.9) / 1.01 / 9
Pedigrees (# inst=16, N=581-1006, k=3-7, w^*=16-39, h_T=52-104)				
i=6	MBE	wAOBF	33.3 (46.7) / 3.0 / 15	18.8 (43.8) / 1.0 / 16
		wR-AOBF	20.0 (33.3) / 1.11 / 15	6.3 (31.3) / 1.11 / 16
	MBE-MM	wAOBF	31.3 (31.3) / 5.0 / 16	31.3 (43.8) / 1.0 / 16
		wR-AOBF	25.0 (37.5) / 1.07 / 16	18.8 (43.8) / 1.07 / 16
	JGLP	wAOBF	6.3 (50.0) / 1.0 / 16	0.0 (56.3) / 1.0 / 16
		wR-AOBF	6.3 (43.8) / 1.06 / 16	0.0 (50.0) / 1.06 / 16
i=14	MBE	wAOBF	28.6 (42.9) / 1.0 / 14	7.1 (50.0) / 1.0 / 14
		wR-AOBF	21.4 (42.9) / 1.0 / 14	0.0 (50.0) / 1.0 / 14
	MBE-MM	wAOBF	14.3 (64.3) / 1.0 / 14	0.0 (64.3) / 1.0 / 14
		wR-AOBF	0.0 (64.3) / 1.04 / 14	0.0 (57.1) / 1.04 / 14
	JGLP	wAOBF	7.1 (57.1) / 1.0 / 14	0.0 (57.1) / 1.0 / 14
		wR-AOBF	0.0 (57.1) / 1.04 / 14	0.0 (57.1) / 1.04 / 14

Table 3: 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, W - average weight, # - 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.

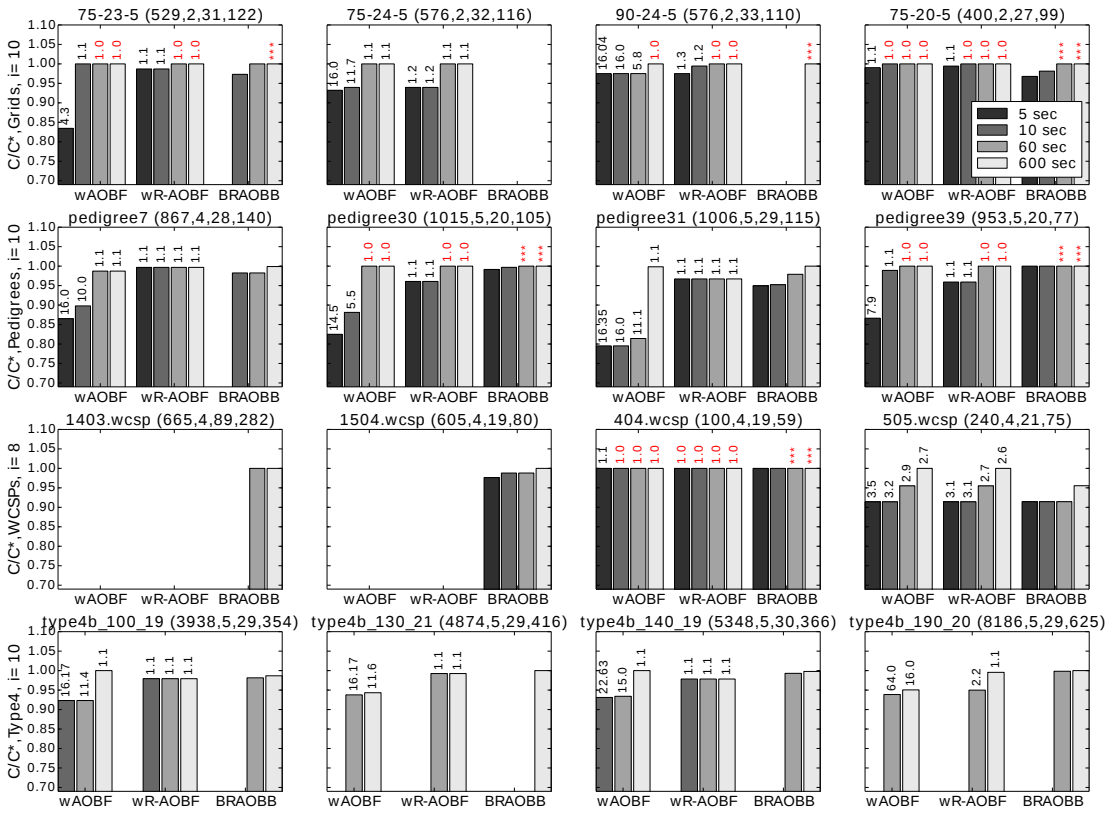


Figure 2: Ratio of the cost obtained by some time point (5, 10, 60, 600 sec) and max cost. Max. cost = optimal, if known, otherwise = best cost found for the problem. Corresponding weight - above the bars. '***' - BRAOBB proved solution optimality. Instance parameters are in format (N, k, w^*, h_T), where N - number of variables, k - max. domain size, w^* - induced width, h_T - pseudo-tree height. MBE-MM. Memory limit 4 GB, time limit 1 hour.

time bounds we show the percentage of instances for which wAOBF and wR-AOBF find better solutions than BRAOBB (X%), the percentage for which they are tied with BRAOBB (Y%), the weight averaged over the instances for which the weighted scheme has found any (possibly suboptimal) solution (W), and the number of instances, for which at least one of the algorithms found a solution (#). We average over the final weights w reported by an algorithm for each instance by the time bound shown. This is done in order to assess how tight, on average, is the w-optimality bound. We show the results for a relatively small and large i-bounds for all three heuristic algorithms considered: MBE, MBE-MM and JGLP.

From the results presented in Tables 3 and 4 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 yield better solutions than BRAOBB on a large percentage of instances on Grids (e.g. 58.3% at 120 sec for wR-AOBF with MBE, i=6) and Type4 (e.g. 80.0% for wAOBF for both MBE and MBE-MM, 3600 sec). They are less effective on Pedigrees (only better than BRAOBB on about 20% to 30% instances). On WCSP weighted BF schemes are mostly inferior. One outlier is i=6 for JGLP, where wAOBF and wR-AOBF are better than BRAOBB in up to 50% cases. However, these results pertain to just the 4 easiest instances of the benchmark and thus are not very conclusive.
2. Weighted best-first algorithms tend to be superior when the i-bound is small (e.g. i=6). When it is large (i.e. strong heuristics) their dominance is usually less pronounced. This ability to produce good solutions when there is a large gap between the i-bound and the problems induced width should be especially beneficial when solving hard problems, for which calculation of accurate heuristics is infeasible.
3. Same trends are seen when comparing different heuristics schemes. Both Weighted BF algorithms are more successful than BRAOBB when heuristic is relatively weaker, as in case of using MBE algorithm, as opposed to stronger heuristic yielded by JGLP.
4. Though all our heuristic algorithms are exponential in i, JGLP is known to have higher time and space requirements (Ihler et al. 2012), because unlike MBE and MBE-MM, which are single-pass, JGLP can have many iterations which improve the bound over time. That explains why on some benchmarks, e.g. WCSPs and Type4, JGLP can be infeasible for higher i-bounds and yields less

I-bound	Heuristic	Algorithm	Time bounds	
			120	3600
			X%(Y%) / W / #	X%(Y%) / W / #
WCSP (# inst=56, N=25-1057, k=2-100, w^*=5-287, h_T=11-337)				
i=6	MBE	wAOBF	0.0 (20.0) / 2.0 / 5	0.0 (33.3) / 1.0 / 6
		wR-AOBF	0.0 (20.0) / 1.45 / 5	0.0 (33.3) / 1.3 / 6
	MBE-MM	wAOBF	20.0 (20.0) / 3.0 / 5	0.0 (20.0) / 2.0 / 5
		wR-AOBF	20.0 (20.0) / 2.45 / 5	0.0 (20.0) / 2.35 / 5
	JGLP	wAOBF	25.0 (25.0) / 2.0 / 4	50.0 (50.0) / 2.0 / 4
		wR-AOBF	50.0 (25.0) / 1.7 / 4	25.0 (25.0) / 1.6 / 4
i=10	MBE	wAOBF	0.0 (50.0) / 1.0 / 2	0.0 (33.3) / 1.0 / 3
		wR-AOBF	0.0 (50.0) / 1.3 / 2	0.0 (33.3) / 1.3 / 3
	MBE-MM	wAOBF	0.0 (50.0) / 2.0 / 2	0.0 (33.3) / 2.0 / 3
		wR-AOBF	0.0 (50.0) / 1.55 / 2	0.0 (33.3) / 1.5 / 3
	JGLP	wAOBF	0.0 (100.0) / 1.0 / 1	0.0 (100.0) / 1.0 / 1
		wR-AOBF	0.0 (100.0) / 1.0 / 1	0.0 (100.0) / 1.0 / 1
Type4 (# inst=10, N=3907-8186, k=5, w^*=21-32, h_T=319-625)				
i=6	MBE	wAOBF	88.9 (0.0) / 44.0 / 9	80.0 (0.0) / 11.0 / 10
		wR-AOBF	11.1 (0.0) / 9.15 / 9	30.0 (0.0) / 1.31 / 10
	MBE-MM	wAOBF	75.0 (0.0) / 34.0 / 4	80.0 (0.0) / 11.0 / 5
		wR-AOBF	0.0 (0.0) / 4.9 / 4	0.0 (0.0) / 1.18 / 5
	JGLP	wAOBF	66.7 (0.0) / 32.0 / 3	83.3 (0.0) / 9.0 / 6
		wR-AOBF	33.3 (0.0) / 5.93 / 3	50.0 (0.0) / 1.12 / 6
i=18	MBE	wAOBF	0.0 (0.0) / 13.0 / 5	77.8 (11.1) / 1.0 / 9
		wR-AOBF	20.0 (0.0) / 1.08 / 5	0.0 (11.1) / 1.09 / 9
	MBE-MM	wAOBF	0.0 (7.7) / 20.0 / 13	30.8 (23.1) / 4.0 / 13
		wR-AOBF	0.0 (23.1) / 1.08 / 13	0.0 (23.1) / 1.08 / 13
	JGLP	wAOBF	0 (0) / 0.0 / 0	0.0 (100.0) / 1.0 / 1
		wR-AOBF	0 (0) / 0.0 / 0	0.0 (100.0) / 1.0 / 1

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, W - average weight, # - 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.

.

solved instances, even though given time it generally provides more accurate heuristics.

5. wAOBF and wR-AOBF often have better performance for short time limits. For example, for Grids, i=6, MBE-MM, wR-AOBF is superior to BRAOBB on 33.3% of problems for 120 seconds, but only on 8.3% for 3600 seconds.

6. Even on benchmarks where weighted BF schemes are inferior to BRAOBB, e.g. WCSPs, they often provide tight w-optimality bounds, as indicated by the average weights being close to 1.0. For example, the average weight for wR-AOBF on Type4, i=18, 120 sec is just 1.08. Such bounds are especially valuable for instances where optimal solution can not be obtained within the time and memory limit, as is the case for majority of Type4 problems.

7. wAOBF seems somewhat superior to wR-AOBF in terms of accuracy, dominating BRAOBB more often. wR-AOBF typically provides tighter w-optimality bounds.

Conclusion

In this paper we present the study of weighted best-first search for graphical models. We extended an advanced best-first search algorithm into several weighted schemes and evaluated their performance, comparing with one of the most competitive branch and bound schemes.

Our results show that on some domains the weighted algorithms are effective as anytime schemes. Crucially, they provide valuable suboptimality guarantees. Therefore weighted schemes should definitely be included as candidate solvers for optimization tasks over graphical models.

As we saw, no scheme always dominates, and so 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.

Alternatively, weighted best-first search together with depth-first branch and bound can be included within a portfolio framework, which was shown to be successful for combinatorial problems and planning, yielding such solvers as, e.g., SATzilla (Xu et al. 2008) and PBP (Gerevini, Saetti, and Vallati 2009). The issue of portfolio building and scheduling remains for future work.

Acknowledgements

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 Pearl, J. 1985. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM* 32:506–536.

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

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence* 113(1):41–85.

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.

Ihler, A. T.; Flerova, N.; Dechter, R.; and Otten, L. 2012. Join-graph based cost-shifting schemes. *arXiv preprint arXiv:1210.4878*.

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.

Kask, K.; Dechter, R.; Larrosa, J.; and Dechter, A. 2005. Unifying cluster-tree decompositions for automated reasoning. *Artificial Intelligence Journal*.

Kjaerulff, U. 1990. Triangulation of graph-based algorithms giving small total space. *Technical Report, University of Aalborg, Denmark*.

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.