Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Anytime Best+Depth-First Search for Bounding Marginal MAP

Radu Marinescu
IBM Research – Ireland
tradu.marinescu@ie.ibm.com

Junkyu Lee, Alexander Ihler, Rina Dechter
University of California, Irvine
Irvine, CA 92697, USA
{junkyul,ihler,dechter}@ics.uci.edu

Abstract

We introduce new anytime search algorithms that combine best-first with depth-first search into hybrid schemes for Marginal MAP inference in graphical models. The main goal is to facilitate the generation of upper bounds (via the best-first part) alongside the lower bounds of solutions (via the depth-first part) in an anytime fashion. We compare against two of the best current state-of-the-art schemes and show that our best+depth search scheme produces higher quality solutions faster while also producing a bound on their accuracy, which can be used to measure solution quality during search. An extensive empirical evaluation demonstrates the effectiveness of our new methods which enjoy the strength of best-first (optimality of search) and of depth-first (memory robustness), leading to solutions for difficult instances where previous solvers were unable to find even a single solution.

Introduction

Graphical models provide a powerful framework for reasoning about conditional dependency structures over many variables. The Maximum a Posteriori Probability (MAP) query over probabilistic graphical models, asks for the mode of the joint probability distribution, while the Marginal MAP (MMAP) generalizes MAP by allowing a subset of the variables to be marginalized. MMAP is known to be notoriously challenging, having complexity NP^{PP}-complete (Park 2002). MMAP distinguishes between maximization variables (called MAP variables) and summation variables, and it is more difficult than either max- or sum- inference tasks alone primarily because summation and maximization operations do not commute, forcing processing along constrained variable orderings that may have significantly higher induced widths (Dechter 1999; 2013). This implies larger search spaces (when using search algorithms) or larger messages (when using message-passing schemes). In particular, MMAP can be NP-hard even on tree structured graphs (Park 2002). Still, MMAP is often the appropriate task where there exist hidden variables or uncertain parameters. It can be also treated as a special case of the more complicated frameworks of decision networks (Howard and Matheson 2005; Liu and Ihler 2013).

Current state-of-the-art schemes focus on solvers that are based on either depth-first or best-first search over AND/OR search spaces (Marinescu, Dechter, and Ihler 2014; 2015; Lee et al. 2016) and were shown to dominate the earlier generation of search-based MMAP algorithms (Park and Darwiche 2003; Yuan and Hansen 2009). Since the problem is inherently hard, recent focus has been on anytime schemes that can hopefully find a suboptimal solution quickly and then improve it if more time/memory is available.

Our focus here is on advancing anytime schemes so that they will produce, not only a solution, but also an upper and a lower bound on the optimal MMAP value that get tighter when provided more time. The idea is that the depth-first branch and bound scheme is inherently anytime and provides a *lower bound* on the optimal solution. Best-first schemes, on the other hand, can be viewed as anytime methods for generating an *upper bound*. Therefore, a hybrid of best+depth-first search can facilitate both. In addition, depth-first search is memory efficient, while best-first is memory expensive but has superior performance, therefore a best+depth hybrid may yield a cost-effective tradeoff between time, memory and accuracy, improving with more time.

One of the most effective anytime MMAP search schemes to date, which we will compare against, is based on the well known idea of weighted-heuristic in best-first search (Pohl 1970), which transforms best-first search into an anytime scheme. This class of best-first search algorithms were shown to be particularly attractive for MMAP (because it minimizes the number of conditional summations during search), yielding an algorithm called WRBFAOO (Lee et al. 2016). Most significantly, they yield a bound on the solution accuracy using the weight value. Another, successful anytime AND/OR depth-first search scheme which was shown to particularly handle memory effectively for MMAP is BRAOBB (Otten and Dechter 2011; Lee et al. 2016), against which we will also compare. The only other competing approach to anytime MMAP that produces both upper and lower bounds is the factor set elimination introduced recently by (Maua and Campos 2012).

Contribution In this paper we will present several best+depth-first search hybrids for MMAP and show that they can be more competitive against the current state-of-the-art in yielding both superior anytime performance and

effective anytime accuracy bounds. Specifically, we propose three best+depth-first variants for exploring the search space. LAOBF (best-first AND/OR search with depth-first lookaheads) traverses the search space in a best-first manner and performs explicit depth-first dives (or lookaheads) below the leaf nodes of the best partial solution tree. AAOBF (alternating best-first with depth-first AND/OR search) is a parameter-free scheme that interleaves an outer best-first loop with an aggressive depth-first loop that aims at finding improved sub-optimal solutions as quickly as possible. LnDFS (learning depth-first AND/OR search) consists of a sequence of depth-first search iterations to find improved feasible solutions that yield tighter lower bounds. Upon encountering a solution, the heuristic node values in the explicated search graph are updated in a best-first manner thus providing improved upper bounds.

Our empirical evaluation on various difficult benchmarks demonstrates the effectiveness of the new schemes compared with state-of-the-art anytime pure depth-first and with the current best version of weighted best-first search solvers as well as the factor set elimination method (Maua and Campos 2012). Notably, our proposed search-based approach can potentially exploit and improve over *any* approximate bounding algorithm for MMAP, by using it as a heuristic to guide search. In our work we illustrate this methodology when using the weighted mini-bucket bounding scheme (Marinescu, Dechter, and Ihler 2014).

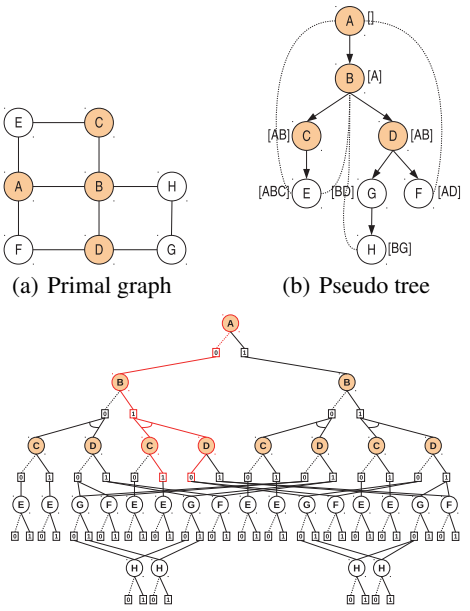
Background

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. $\mathbf{F} = \{\psi_\alpha(\mathbf{X}_\alpha) : \alpha \in F\}$ is a set of discrete positive real-valued factors indexed by set F, where each ψ_α is defined on a subset of variables $\mathbf{X}_\alpha \subseteq \mathbf{X}$, called its scope. Specifically, $\psi_\alpha : \Omega_\alpha \rightarrow \mathbb{R}_+$, where Ω_α is the Cartesian product of the domains of each variable in $\mathbf{X}_\alpha$. The scopes of the factors yield a *primal graph* whose vertices are the variables and whose edges connect any two variables that appear in the scope of the same factor. The model $\mathcal{M}$ defines a factorized probability distribution on $\mathbf{X}$, $P(\mathbf{x}) = \frac{1}{Z}\prod_{\alpha \in F}\psi_\alpha(\mathbf{x}_\alpha)$. The *partition function*, Z, normalizes the probability.

Let $\mathbf{X}_M = \{X_1, ..., X_m\}$ be a subset of $\mathbf{X}$ called MAP variables and $\mathbf{X}_S = \mathbf{X} \setminus \mathbf{X}_M$ be the complement of $\mathbf{X}_M$, called sum variables. The Marginal MAP (MMAP) task seeks an assignment $\mathbf{x}^*_M$ to variables $\mathbf{X}_M$ having maximum probability. This requires access to the marginal distribution over $\mathbf{X}_M$, which is obtained by summing out variables $\mathbf{X}_S$: Therefore, $\mathbf{x}^*_M = \text{argmax}_{\mathbf{x}_M} \sum_{\mathbf{x}_S} \prod_{\alpha \in F} \psi_\alpha(\mathbf{x}_\alpha)$.

AND/OR Search Spaces

A significant recent improvement in search for MMAP inference have been achieved by using AND/OR search spaces, which often capture problem structure far better than standard OR search methods (Marinescu, Dechter, and Ihler 2014; Dechter and Mateescu 2007). The AND/OR search space is defined relative to a *pseudo tree* of the primal graph, which captures problem decomposition.



(a) Primal graph (b) Pseudo tree

(c) Context minimal AND/OR search graph

Figure 1: A simple graphical model.

Definition 1 (pseudo tree, start)**.** *A* pseudo tree *of an undirected graph $G = (V, E)$ is a directed rooted tree $\mathcal{T} = (V, E')$ such that every arc of G not in E' is a back-arc in $\mathcal{T}$ connecting a node in $\mathcal{T}$ to one of its ancestors. The arcs in E' may not all be included in E. A directed rooted tree $\mathcal{T}_s = (V', E'')$, $V' \subseteq V$, is called a* start pseudo tree *of $\mathcal{T}$ if it has the same root and is a connected subgraph of $\mathcal{T}$.*

For MMAP we need to restrict the collection of pseudo trees to *valid* ones only.

Definition 2 (valid pseudo tree)**.** *Let $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ be a graphical model with primal graph G and $\mathbf{X}_M \subseteq \mathbf{X}$. A pseudo tree $\mathcal{T}$ of G is* valid for the MAP variables $\mathbf{X}_M$ *if $\mathcal{T}$ restricted to $\mathbf{X}_M$ forms a connected* start pseudo tree *having the same root as $\mathcal{T}$.*

Given a graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ with primal graph G and valid pseudo tree $\mathcal{T}$ of G, the *AND/OR search tree* $S_\mathcal{T}$ based on $\mathcal{T}$ has alternating levels of OR nodes corresponding to the variables, and AND nodes corresponding to the values of the OR parent's variable, with edge weights extracted from the original functions $\mathbf{F}$ (for details see (Dechter and Mateescu 2007)). Identical sub-problems, identified by their *context* (the partial instantiation that separates the sub-problem from the rest of the problem graph), can be merged, yielding an *AND/OR search graph*. Merging all context-mergeable nodes yields the *context minimal AND/OR search graph*, denoted $C_\mathcal{T}$. The size of $C_\mathcal{T}$ is exponential in the induced width of G along a depth-first traversal of $\mathcal{T}$ (also known as the *constrained* induced width).

Definition 3 (solution subtree)**.** *A* solution subtree $\hat{x}_M$ *of $C_\mathcal{T}$ relative to the MAP variables $\mathbf{X}_M$ is a subtree of $C_\mathcal{T}$ restricted to $\mathbf{X}_M$ that: (1) contains the root of $C_\mathcal{T}$; (2) if an internal OR node $n \in C_\mathcal{T}$ is in $\hat{x}_M$, then n is labeled with a*

MAP variable and exactly one of its children is in $\hat{x}_M$; (3) if an internal AND node $n \in C_{\mathcal{T}}$ is in $\hat{x}_M$ then all its OR children which denote MAP variables are in $\hat{x}_M$.

Each node n in $C_{\mathcal{T}}$ can be associated with a *value* $v(n)$ capturing the optimal marginal MAP value of the conditioned sub-problem rooted at n, while for a sum variable it is the likelihood of the partial assignment denoted by n. Clearly, $v(n)$ can be computed recursively based on the values of n's successors: OR nodes by maximization or summation (for MAP or sum variables, respectively), and AND nodes by multiplication.

Example 1. *Figure 1 shows a simple graphical model. Figure 1(a) is the primal graph of 8 bi-valued variables and 10 binary factors, where the MAP and sum variables are $\mathbf{X}_M = \{A, B, C, D\}$ and $\mathbf{X}_S = \{E, F, G, H\}$, respectively. Figure 1(b) is a valid pseudo tree whose MAP variables form a start pseudo tree (dashed lines denote back-arcs). Figure 1(c) displays the context minimal AND/OR search graph based on the pseudo tree (the contexts are shown next to the pseudo tree nodes). A solution subtree corresponding to the MAP assignment (A=0,B=1,C=1,D=0) is shown in red.*

Current Anytime Search for MMAP

Current state-of-the-art anytime search methods for MMAP are based on either *depth-first* or *best-first* AND/OR search (Marinescu, Dechter, and Ihler 2014; 2015; Lee et al. 2016). We provide below details of the best current schemes.

Breadth Rotating Depth-First AND/OR Branch and Bound (BRAOBB-MMAP) is a recent anytime search algorithm that explores in a *depth-first* manner the context minimal AND/OR search graph (Lee et al. 2016; Otten and Dechter 2011). During search, BRAOBB-MMAP keeps track of the value of the best solution found so far (a lower bound on the optimal MMAP value) and uses this value and the heuristic function to prune away portions of the search space that are guaranteed not to contain the optimal solution. The algorithm also rotates through the independent subproblems rooted at AND nodes in a round-robin manner to generate anytime solutions quickly.

Weighted Recursive Best-First AND/OR Search (WRBFAOO-MMAP), transforms the recursive best-first AND/OR search algorithm RBFAOO-MMAP (Marinescu, Dechter, and Ihler 2015) into an anytime scheme by using the principle of weighted search (Pohl 1970). Specifically, the method generates suboptimal solutions by employing an inadmissible heuristic which is obtained by inflating the original heuristic with a weight (≥ 1). The weight is then decreased by a fixed amount at the subsequent iterations in the attempt to improve the best solution found so far. WRBFAOO operates within restricted memory and restarts the search after each weight update. Our recent extensive empirical evaluations on various benchmarks showed that BRAOBB-MMAP and WRBFAOO-MMAP are the overall best performing anytime search algorithms both in terms of solution quality as well as responsiveness (Lee et al. 2016).

Weighted Mini-Bucket Heuristics The effectiveness of all the above search algorithms greatly depends on the quality of the upper bound heuristic function that guides the search. The MMAP search algorithms use the weighted mini-bucket (WMB) based heuristic (Dechter and Rish 2003; Liu and Ihler 2011) which can be pre-compiled along the reverse order of the pseudo tree.

The weighted mini-bucket improves the naïve mini-bucket bound (Dechter and Rish 2003) with Hölder's inequality. For a given variable X_k, the mini-buckets Q_{kr} associated with X_k are assigned a non-negative *weight* $w_{kr} \geq 0$, such that $\sum_r w_{kr} = 1$. Then, each mini-bucket r is processed using a powered sum, $(\sum_{X_k} f(X)^{1/w_{kr}})^{w_{kr}}$. It is useful to note that w_{kr} can be interpreted as a "temperature"; if $w_{kr} = 1$, it corresponds to a standard summation, while if $w_{kr} \to 0$, it instead corresponds to a maximization over X_k. In addition, a cost shifting scheme (or reparameterization) can be applied across mini-buckets to match the marginal beliefs (or "moments") to further tighten the bound (Ihler et al. 2012). The single-pass message passing algorithm that combines weighted mini-bucket and moment matching reparameterization yields a scheme denoted by WMB-MM(i), where i is called the i-bound and controls the accuracy of mini-bucket approximation (Dechter and Rish 2003; Liu and Ihler 2011; Marinescu, Dechter, and Ihler 2014).

Anytime Best+Depth-First AND/OR Search

While weighted best-first search schemes are favorable for MMAP because they can save on the number of conditional likelihood summation problems compared with depth-first search (Lee et al. 2016), their anytime performance still depends on the initial weight value as well as on the weight update schedule. Some of their drawbacks are: (i) the algorithms sometime spend significant computational resources (time, memory) struggling to find even the first suboptimal solution; (ii) depending on the weight update schedule either very few solutions are generated or there is no improvement in the solution quality; (iii) most of all, the suboptimality bound produced by the weight is often very loose.

In this paper, we take a different approach and introduce new schemes that combine best-first and depth-first search into several best+depth hybrid anytime search algorithms that not only produce superior anytime solutions (better quality solutions and significantly faster) than state-of-the-art, but they also provide *anytime upper and lower bounds* on the optimal MMAP value that improve and can be used to better gauge the solution quality during search.

Notations Let $G'_{\mathcal{T}}$ denote the explicated context minimal AND/OR search graph relative to pseudo tree $\mathcal{T}$. Each node $n \in G'_{\mathcal{T}}$ maintains two values $q(n)$ and $l(n)$. The quantity $q(n)$ represents an upper bound provided by the heuristic evaluation function at the node n (i.e., the weighted mini-bucket value), while $l(n)$ is the cost of the current best solution found below the node n, and is therefore yielding also a lower bound on the best solution in the search space below n (we solve a maximization task). We use $\mathcal{U}$ and $\mathcal{L}$ to denote the current best global upper and lower bounds on the optimal MMAP value. For node $n \in G'_{\mathcal{T}}$, $ch(n)$ denote its children in $G'_{\mathcal{T}}$, while $w_{(n,m)}$ is the weight labeling the arc $n \to m$

Algorithm 1 Expanding a node by generating its successors

Require: node n, pseudo tree $\mathcal{T}$
function EXPAND(n)
  if n is OR node labeled by $\langle X_i \rangle$ and $X_i \in \mathbf{X}_M$ **then**
    for all values $x_i \in D_i$ **do**
      Create AND child c labeled by $\langle X_i, x_i \rangle$
      if c is terminal **then**
        $q(c) \leftarrow 1$ and $l(c) \leftarrow 1$
      else
        $q(c) \leftarrow h(c)$ and $l(c) \leftarrow -\infty$
      end if
    end for
  else if n is AND labeled by $\langle X_i, x_i \rangle$ and $X_i \in \mathbf{X}_M$ **then**
    for all successors X_j of X_i in $\mathcal{T}$ **do**
      Create OR child c labeled by $\langle X_j \rangle$
      if $X_j \in \mathbf{X}_M$ **then**
        $q(c) \leftarrow h(c)$ and $l(c) \leftarrow -\infty$
      else if $X_j \in \mathbf{X}_S$ **then**
        $q(c) = l(c) \leftarrow eval(\mathcal{M}|\bar{\mathbf{x}})$
      end if
    end for
  end if
end function

Algorithm 2 Updating the node q- and l-values

Require: node n, explicated search graph $G'_{\mathcal{T}}$, pseudo tree $\mathcal{T}$
function UPDATE(n)
  for all ancestor p of n in $G'_{\mathcal{T}}$, including n **do**
    if p is OR node **then**
      $l(p) \leftarrow \max_{m \in ch(p)}(w_{(p,m)} \cdot l(m))$
      $q(p) \leftarrow \max_{m \in ch(p)}(w_{(p,m)} \cdot q(m))$
      $m' \leftarrow \text{argmax}_{m \in ch(p)}(w_{(p,m)} \cdot q(p))$
      Mark with symbol $\star$ the arc $p \rightarrow m'$
    else if p is AND node **then**
      $l(p) \leftarrow \prod_{m \in ch(p)} l(m)$
      $q(p) \leftarrow \prod_{m \in ch(p)} q(m)$
    end if
  end for
end function

in $G'_{\mathcal{T}}$. Algorithm 1 describes the node expansion procedure during search, while Algorithm 2 shows how the q- and l-values are updated bottom-up based on the corresponding values of their children in the search graph. Note that during the update of q-values, we also mark with a $\star$ symbol the arc corresponding to the best child m' of an OR node n.

LAOBF: Best-First AND/OR Search with Depth-First Lookaheads

As noted before, the rationale behind best+depth based search is to alternate in some manner between the two styles of search space exploration, where the best-first component progresses towards improved upper bounds and depth-first progresses towards improved potential solutions and their accompanied lower bounds.

A simple way to augment best-first search with a mechanism that generates solutions is to do explicit depth-first lookahead dives under some nodes in the best-first search frontier (Stern et al. 2010). Our first best+depth hybrid,

Algorithm 3 LAOBF

Require: Graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, pseudo tree $\mathcal{T}$, heuristic function $h(\cdot)$, $\mathbf{X}_M = \mathbf{X} \setminus \mathbf{X}_S$, cutoff θ
Create an OR node s labeled by the root of $\mathcal{T}$ and
Initialize $\mathcal{U} = q(s) = h(s)$, $\mathcal{L} = l(s) = -\infty$, $G_{\mathcal{T}} = \{s\}$, $T_b = \{s\}$
$counter \leftarrow 0$
while $\mathcal{U} \neq \mathcal{L}$ **do**
  Let $cost(T_b) = \prod_{(n,m) \in arcs(T_b)} w_{(n,m)}$
  if $counter$ **mod** $\theta = 0$ **then**
    for all node m in $tips(T_b)$ **do**
      $l(m) \leftarrow$ `dfs-lookahead(`m`)`
    end for
  end if
  if $cost(T_b) \cdot \prod_{m \in tips(T_b)} l(m)) > \mathcal{L}$ **then**
    $\mathcal{L} \leftarrow cost(T_b) \cdot \prod_{m \in tips(T_b)} l(m))$
    $\mathcal{U} \leftarrow q(s)$ and **print**$\langle \mathcal{U}, \mathcal{L} \rangle$
  end if
  Select non-terminal tip node n in T_b
  EXPAND(n); UPDATE(n)
  Select new T_b by tracing down $\star$-marked arcs from root s
  $counter \leftarrow counter + 1$
end while
return $\mathcal{U}$

called *Best-First AND/OR Search with Depth-First Lookaheads* (LAOBF), is described in Algorithm 3. Specifically, it generates anytime solutions (lower bounds) by having depth-first dives below the current best partial solution tree to find a (suboptimal) solution. We elaborate on these dives next.

DFS and BFS Iterations Let T_b be the current best partial solution tree determined by the best-first search scheme, $tips(T_b)$ be the set of tip nodes of T_b, and m be one of these tips. The algorithm performs a depth-first dive at the subtree rooted at m, recording the conditioned lower bound found as $l(m)$. Once all the tips of T_b are explored in such a depth-first manner, a global lower bound $\mathcal{L}$ of T_b can be obtained by multiplying the generated lower bounds $l(m)$ by their corresponding weights of the arcs in T_b. Once such a full dive is accomplished, best-first search takes over as usual, expanding a tip node n from the current T_b, updating the q-values of n's ancestors, and selecting the next best partial solution tree. Then, a new depth-first dive can be performed, and so on. The updated q-value of the root node provides an anytime (and often improved) upper bound $\mathcal{U}$ on the optimal MMAP value.

Performing the depth-first lookaheads at every single iteration can often incur significant overhead. To bound this overhead, we use a *cutoff* parameter θ to trigger the depth-first lookaheads every θ best-first iterations. It is also important to note that the cache information is shared between the depth-first and best-first iterations, so that the solutions to the summation subproblems could be reused.

AAOBF: Alternating Best-First with Depth-First AND/OR Search

LAOBF restricts the depth-first exploration to the subspaces below the tip nodes of the current best partial solution tree

Algorithm 4 AAOBF

Require: Graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, pseudo tree $\mathcal{T}$, heuristic function $h(\cdot)$, $\mathbf{X}_M = \mathbf{X} \setminus \mathbf{X}_S$

1: Create an OR node s labeled by the root of $\mathcal{T}$
2: Initialize $\mathcal{U} = q(s) = h(s)$, $\mathcal{L} = l(s) = -\infty$, $G'_{\mathcal{T}} = \{s\}$, $T_l = \{s\}$, $T_b = \{s\}$, $flag = false$
3: **while** $\mathcal{U} \neq \mathcal{L}$ **do**
4: **if** $tips(T_l) \neq \emptyset$ **then**
5: Select non-terminal tip node n in T_l
6: EXPAND(n); UPDATE(n)
7: **if** n is OR node **then**
8: $m' \leftarrow \text{argmax}_{m \in ch(n)}(w_{(n,m)} \cdot q(m))$
9: Mark with symbol $\diamond$ the arc $n \rightarrow m'$
10: **end if**
11: Select new T_l using $\diamond$-marked arcs from s
12: $flag \leftarrow true$
13: **else**
14: **if** $l(s) > \mathcal{L}$ **then**
15: $\mathcal{U} \leftarrow q(s)$, $\mathcal{L} \leftarrow f(s)$ and **print**$\langle \mathcal{U}, \mathcal{L} \rangle$
16: **end if**
17: **if** $flag = true$ **then**
18: **for all** OR nodes n in $G'_{\mathcal{T}}$ **do**
19: **if** $l(n) \neq -\infty$ **then**
20: $m' \leftarrow \text{argmin}_{m \in ch(n)} \frac{l(n)}{w_{(n,m)} \cdot q(m)}$
21: **else**
22: $m' \leftarrow \text{argmax}_{m \in ch(n)}(w_{(n,m)} \cdot q(m))$
23: **end if**
24: Mark with symbol $\diamond$ the arc $n \rightarrow m'$
25: **end for**
26: Select new T_b using $\star$-marked arcs from s
27: $flag \leftarrow false$
28: **end if**
29: Select non-terminal tip node n in T_b
30: EXPAND(n); UPDATE(n)
31: Select new T_b using $\star$-marked arcs from s
32: Select new T_l using $\diamond$-marked arcs from s
33: **end if**
34: **end while**
35: **return** $\mathcal{U}$

T_b, and, as we will show in the experimental section, this may not always lead to improved lower bounds. Moreover, LAOBF requires tuning the cutoff parameter, and finding a good value for it can be challenging. Therefore, our second approach called *Alternating Depth-First and Best-First AND/OR Search* (AAOBF) is a parameter-free best+depth hybrid that aims at diversifying more the depth-first exploration (thus allowing it to explore feasible solutions in a greedier manner rather than restricting it to specific regions dictated by best-first search).

Specifically, AAOBF (Algorithm 4) alternates between a depth-first and a best-first stage and maintains two independent partial solution trees for each. In its depth-first stage it expands, depth-first, a current *feasible* partial solution tree T_l (lines 4–12), where T_l is a partial assignment that can be extended to a feasible (often suboptimal) solution, to improve the global lower bound. In its best-first stage it expands, best-first, a current *best* partial solution tree T_b (lines 14–32) to improve the global solution and an associated upper bound and guide the exploration closer to the region containing the optimal solution. At any stage during search, the partial solution trees T_l and T_b can be identified by tracing down $\diamond$- and respectively $\star$-marked arcs from the root s of $G'_{\mathcal{T}}$.

The algorithm begins by expanding non-terminal tip nodes from the current T_l. Each node expansion is followed by a bottom-up revision of the q- and l-values (line 6). An OR node which was just expanded also marks with a $\diamond$ symbol the arc to its best AND child (lines 7–9). Notice that the $\diamond$-markings do not change during this stage, and therefore T_l is extended depth-first to a solution tree.

When a solution is found (i.e., T_l has no tips), AAOBF attempts to give control to best-first search. Before turning to best-first search, it revises the $\diamond$-markings along the arcs in $G'_{\mathcal{T}}$ so that a new feasible partial solution tree could be selected later. Specifically, if there is a lower bound $l(n)$ at node n (i.e., $l(n) \neq -\infty$) then AAOBF marks with a $\diamond$ symbol the arc to the AND child that minimizes the ratio between the current $l(n)$ and the successor's updated q-value, and therefore is the most likely to increase $l(n)$. Otherwise, it selects the best AND child having the largest heuristic (upper bounding) value.

AAOBF continues by expanding nodes from T_b in the usual best-first manner. However, if during this stage a new feasible partial solution tree T_l is identified (line 32), the algorithm switches immediately to the depth-first expansion of the new T_l. Search terminates with the optimal MMAP value when the global lower and upper bounds are equal.

LnDFS: Learning Depth-First AND/OR Search

Algorithm 5 describes a different approach for anytime MMAP which belongs to the family of learning depth-first search algorithms (Bonet and Geffner 2005). It is also a parameter-free scheme, like AAOBF, however it maintains a single best partial solution tree T_b.

Specifically, LnDFS expands the current best partial solution tree T_b in a depth-first manner while selecting the best child of an OR node by sorting the up-to-date q- values backed up from the previous iterations. Unlike best-first expansion, it postpones updating node values until either T_b extended to a solution tree (lines 20–24) or the current T_b cannot be extended to a better solution than the one given by the current best lower bound $\mathcal{L}$, namely, $ub(T_b) \leq \mathcal{L}$ (lines 15–19). In either case, LnDFS backtracks to the root and updates the q- and l-values of the nodes that are ancestors in $G'_{\mathcal{T}}$ of the leaf nodes of T_b. This step corresponds to learning better heuristic upper bounds of the nodes in $G'_{\mathcal{T}}$. Therefore, LnDFS improves the current best upper bound (given by the q-value of the root node) in an anytime fashion, and produces (often improved) anytime lower bounds (given by the l-value of the root node) at the same time.

Notice that during the node expansion process (lines 7–13) the $\star$-markings along the arc of T_b do not change and therefore T_b is expanded depth-first until it switches to the next iteration after revising the $\star$-markings during the updates. Therefore, LnDFS interleaves smoothly best-first and depth-first search.

Since the proposed best+depth-first search algorithms for MMAP traverse the context minimal AND/OR search graph, we have that:

Algorithm 5 LnDFS

Require: Graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, pseudo tree $\mathcal{T}$, heuristic function $h(\cdot)$, $\mathbf{X}_M = \mathbf{X} \setminus \mathbf{X}_S$

Create an OR node s labeled by the root of $\mathcal{T}$
Initialize $\mathcal{U} = q(s) = h(s)$, $\mathcal{L} = l(s) = -\infty$, $G_{\mathcal{T}} = \{s\}$, $T_b = \{s\}$
while $\mathcal{U} \neq \mathcal{L}$ **do**
  if $tips(T_b) \neq \emptyset$ **then**
    Select non-terminal tip node n in T_b
    EXPAND(n)
    if n is OR node **then**
      $q(n) \leftarrow \max_{m \in ch(n)}(w_{(n,m)} \cdot q(m))$
      $m' \leftarrow \text{argmax}_{m \in ch(n)}(w_{(n,m)} \cdot q(m))$
      Mark with symbol $\star$ the arc $n \rightarrow m'$
    else if n is AND node **then**
      $q(n) \leftarrow \prod_{m \in ch(n)} q(m)$
    end if
    $ub(T_b) \leftarrow \prod_{(n,m) \in arcs(T_b)} w_{(n,m)} \cdot \prod_{m \in leaves(T_b)} q(m)$
    if $ub(T_b) \leq \mathcal{L}$ **then**
      for all nodes $m \in leaves(T_b)$ **do**
        UPDATE(m)
      end for
    end if
  else
    for all nodes $m \in leaves(T_b)$ **do**
      UPDATE(m)
    end for
  end if
  Let $\mathcal{U} = q(s)$, $\mathcal{L} = \max(\mathcal{L}, l(s))$ and **print** $\langle \mathcal{U}, \mathcal{L} \rangle$
  Select new T_b by tracing down $\star$-marked arcs from root s
end while
return $\mathcal{U}$

Theorem 1 (complexity)**.** *Algorithms LAOBF, AAOBF, and LnDFS are sound and complete (namely, they will find an optimal solution if given enough time and space). Their complexity is time and space $O(n \cdot k^{w_c^*})$, where n is the number of variables, k bounds the domain size, and w_c^* it the induced width of the valid pseudo tree (i.e., constrained induced width).*

Finally, we note that the hybrids of depth-first and best-first search can switch to using only depth-first search as soon as memory fills up, thus continuing to improve the lower bound. For example, LnDFS can be turned into an iterative deepening A* search over the AND/OR search tree as it discards updated nodes and re-expands nodes following the standard depth-first search with the updated global upperbound $\mathcal{U}$.

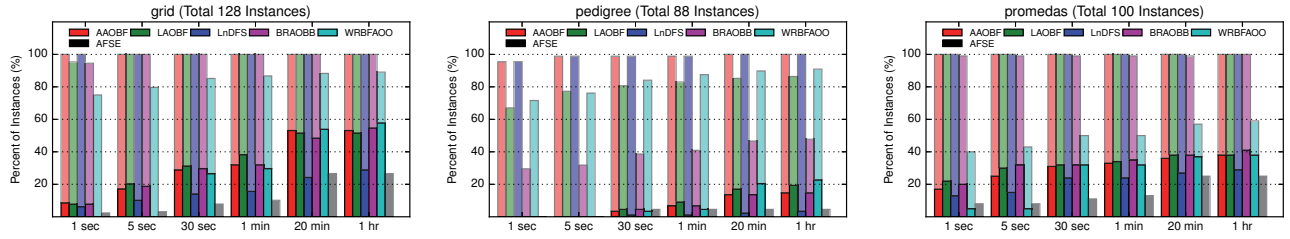
Experiments

We compare empirically the anytime performance of our proposed best+depth-first search algorithms (LAOBF, AAOBF, and LnDFS) with the best-first (WRBFAOO) and the depth-first search algorithm (BRAOBB) on benchmark problems generated from the PASCAL2 Inference Challenge (Elidan, Globerson, and Heinemann 2012). For reference, we also compare with the recent factor-set elimination scheme of (Maua and Campos 2012) which we denote by AFSE. The competing MMAP search algorithms are provided with the same heuristic function, WMB-MM(i) where we used the i-bound of 20 or the largest i-bound that fits 4 GB of memory. Four of the algorithms require tuning parameters to optimize the anytime performance, as follows: the cutoff parameter θ of LAOBF that triggers the depth-first lookahead was set to 1000 (this value was determined experimentally), the subproblem rotation parameter of BRAOBB was set to 1000, and the initial weight and overestimation parameter of WRBFAOO were set to 64 and 1, respectively. The weight reduction schedule of WRBFAOO was $w_{i+1} = \sqrt{w_i}$. The step size used by AFSE to control the partitioning of the factor sets propagated was set to 1 (we tried larger values without any significant difference in performance). The algorithms were allotted up to 1 hour time limit within 24 GB of memory, where all but BRAOBB had an additional 4 GB memory limit on the size of the cache table that primarily stores the exact solutions of conditional likelihoods under summation variables.

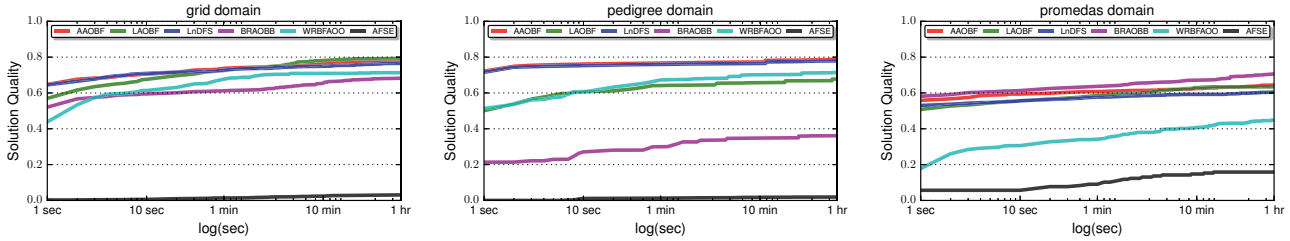
Our benchmark set includes 3 problem domains from grid network (`grid`), genetic linkage analysis (`pedigree`), and medical diagnosis expert systems (`promedas`). Since the original problems are pure MAP tasks, we generated 4 synthetic MMAP instances from each of the pure MAP tasks by randomly selecting 50% of the variables as MAP variable. We created 128 `grid` , 88 `pedigree` , and 100 `promedas` MMAP instances. Their parameters ranged as follows: n number of variables (144-2500), k maximum domain size (2-7), w_c constrained induced width (11-834). Note that, this paper illustrates much harder MMAP problem instances compared to previous papers (Yuan and Hansen 2009; Maua and Campos 2012; Lee et al. 2016).

Responsiveness The responsiveness characterizes how fast an algorithm can discover any solution. Figure 2(a) compares the responsiveness from 1 second to 1 hour. For each time bound, the plot shows a group of 6 vertical bars. Within each group, the stacked bars from left to right correspond to algorithms AAOBF, LAOBF, LnDFS, BRAOBB, WRBFAOO and AFSE, respectively. For each algorithm, the height of the bottom bar (darker color) shows the percentage of instances solved optimally within the corresponding time bound, while the height of the stacked bar (lighter color) shows the percentage of instances with any solution. We can see that AAOBF and LnDFS significantly outperform AFSE, BRAOBB, and WRBFAOO as they discover at least one solution for all problem instances in 5 seconds. Notice the poor performance of AFSE which couldn't solve optimally any of the instances, and ran out of memory on 73%, 81%, and 75% of the `grid`, `pedigree`, and `promedas` instances, respectively.

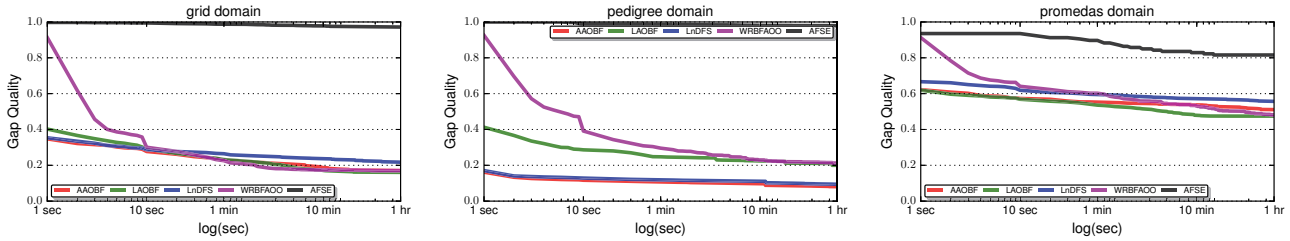
Average Solution Quality Figure 2(b) plots the average solution quality for each domain as a function of time. The solution quality of a problem instance is the best cost discovered within the time bound divided by the optimal cost of the problem instance. When the optimal cost is unknown, the best-known cost was used instead. Clearly, for any given time bound, larger values of the solution quality indicate superior solutions (i.e., larger lower bounds). We can observe that AAOBF and LnDFS produced high-quality anytime solutions



(a) Responsiveness



(b) Average solution quality



(c) Average gap between upper and lower bounds

Figure 2: Anytime performance measures for `grid`, `pedigree`, and `promedas` benchmarks.

across all different domains, yet LnDFS performed slightly worse than AAOBF on the `promedas` domain. On the other hand, AFSE completely failed on all domains, and BRAOBB or WRBFAOO showed worse quality on either `pedigree` or `promedas` domain. Notably, BRAOBB was the best on the `promedas` domain.

Average Gap between Upper and Lower Bounds Figure 2(c) plots the average gap between the upper and lower bounds. The gap of a problem instance is computed in log space by normalizing the difference between the upper and the lower bound by the upper bound, where the upper bound is the best known anytime solution in log space. When the gap becomes 0, it guarantees optimality. If no solution is available at all, we define the gap to be 1. Clearly, both AAOBF and LAOBF improve significantly the gap of WRBFAOO at all time bounds, especially at the smaller ones. LnDFS produced a slightly larger gap than the other search-based algorithms on the `grid` and `promedas` domain. Again, AFSE hardly improved the gap over time.

Related Work

The idea of combining depth-first and best-first search was first described in (Pearl 1984). A similar idea was also employed in mixed integer programming algorithms as a plunging strategy in (Achterberg 2007). The work that is closest in spirit to our LAOBF is that by (Stern et al. 2010) who developed an A* with depth-first lookaheads in the context of generic path-finding. More recently, (Allouche et al. 2015) introduced a hybrid best-first search algorithm for solving Weighted CSPs that can be guided by a tree decomposition. It selects a node in the best-first manner and performs standard depth-first search with an adaptive number of backtracks to expand the frontier. LnDFS is closely related to the bounded learning depth-first search (Bonet and Geffner 2005) which was originally developed for non-deterministic planning and MDP planning.

Approximation algorithms for MMAP, including variational methods, have also been introduced (Jiang, Rai, and Hal 2011; Cheng et al. 2012; Liu and Ihler 2013). Upper and lower bound versions of these methods are closely related to the WMB heuristic we use to guide the search. However, unlike our search based algorithms, these schemes do

not guarantee eventual optimality of their solutions without significant memory requirements.

The anytime MMAP algorithm introduced recently by (Maua and Campos 2012) is another approach that can provide upper and lower bounds. It is an iterative join-tree based algorithm that propagates sets of messages (called factor sets) between the clusters of the join tree. These messages, however, tend to grow very large and therefore the method is limited to solving relatively easy problems with small induced widths.

Conclusions

We proposed new hybrid best+depth-first search algorithms for anytime MMAP. These schemes are able to compute not only anytime lower bounds, but also anytime upper bounds on the optimal MMAP value, and the gap between the two can be used to better gauge the solution quality during search. Our extensive empirical evaluation on various benchmarks demonstrates the effectiveness of the new algorithms compared with existing state-of-the-art depth-first and weighted best-first search as well as anytime factor set elimination. Our new best+depth-first search approach produces superior quality solutions more quickly than both best-first and depth-first search, and it also guarantees tighter gap compared to the weighted best-first and factor set elimination schemes.

Acknowledgments

This work was supported in part by NSF grants IIS-1526842 and IIS-1254071, and by the US Air Force under Contract No. FA8750-14-C-0011 under the DARPA PPAML program. We are grateful to the reviewers for their helpful comments.

References

Achterberg, T. 2007. *Constraint Integer Programming*. Ph.D. Dissertation, Zuse Institute Berlin.

Allouche, D.; de Givry, S.; Katsileros, G.; Schiex, T.; and Zytnicki, M. 2015. Anytime hybrid best-first search with tree decomposition for weighted csp. In *International Conference on Principles and Practice of Constraint Programming (CP)*, 12–29.

Bonet, B., and Geffner, H. 2005. An algorithm better than AO*? In *AAAI Conference on Artificial Intelligence*, 1343–1347.

Cheng, Q.; Chen, F.; Dong, J.; Xu, W.; and Ihler, A. 2012. Approximating the sum operation for marginal MAP inference. In *AAAI Conference on Artificial Intelligence*, 1882–1887.

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 of approximating inference. *Journal of ACM* 50(2):107–153.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence* 113(1-2):41–85.

Dechter, R. 2013. *Reasoning with Probabilistic and Deterministic Graphical Models: Exact Algorithms*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Elidan, G.; Globerson, A.; and Heinemann, U. 2012. PASCAL 2011 probabilistic inference challenge. http://www.cs.huji.ac.il/project/PASCAL/.

Howard, R., and Matheson, J. 2005. Influence diagrams. *Decision Analysis* 2(3):127–143.

Ihler, A.; Flerova, N.; Dechter, R.; and Otten, L. 2012. Join-graph based cost-shifting schemes. In *Uncertainty in Artificial Intelligence (UAI)*, 397–406.

Jiang, J.; Rai, P.; and Hal, D. 2011. Message-passing for approximate MAP inference with latent variables. In *Advances in Neural Information Processing Systems (NIPS)*, 1197–1205.

Lee, J.; Marinescu, R.; Dechter, R.; and Ihler, A. 2016. From exact to anytime solutions for marginal MAP. In *AAAI Conference on Artificial Intelligence*, 1749–1755.

Liu, Q., and Ihler, A. 2011. Bounding the partition function using Hölder's inequality. In *International Conference on Machine Learning (ICML)*, 849–856.

Liu, Q., and Ihler, A. 2013. Variational algorithms for marginal MAP. *Journal of Machine Learning Research* 14:3165–3200.

Marinescu, R.; Dechter, R.; and Ihler, A. 2014. AND/OR search for marginal MAP. In *Uncertainty in Artificial Intelligence (UAI)*, 563–572.

Marinescu, R.; Dechter, R.; and Ihler, A. 2015. Pushing forward marginal MAP with best-first search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 696–702.

Maua, D., and Campos, C. D. 2012. Anytime marginal MAP inference. In *International Conference on Machine Learning (ICML)*, 1471–1478.

Otten, L., and Dechter, R. 2011. Anytime AND/OR depth-first search for combinatorial optimization. In *International Symposium on Combinatorial Search*, 117–702.

Park, J., and Darwiche, A. 2003. Solving MAP exactly using systematic search. In *Uncertainty in Artificial Intelligence (UAI)*, 459–468.

Park, J. 2002. MAP complexity results and approximation methods. In *Uncertainty in Artificial Intelligence (UAI)*, 388–396.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies*. Addison-Wesley.

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

Stern, R.; Kulberis, T.; Felner, A.; and Holte, R. 2010. Using lookahead with optimal best-first search. In *AAAI Conference on Artificial Intelligence*, 185–190.

Yuan, C., and Hansen, E. 2009. Efficient computation of jointree bounds for systematic MAP search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 1982–1989.