Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

From Exact to Anytime Solutions for Marginal MAP

Junkyu Lee
University of California, Irvine
Irvine, CA 92697, USA
junkyul@ics.uci.edu

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

Rina Dechter
University of California, Irvine
Irvine, CA 92697, USA
dechter@ics.uci.edu

Alexander Ihler
University Irvine, CA 92697, USA
ihler@ics.uci.edu

Abstract

This paper explores the anytime performance of search-based algorithms for solving the Marginal MAP task over graphical models. The current state-of-the-art for solving this challenging task is based on best-first search exploring the AND/OR graph with the guidance of heuristics based on mini-bucket and variational cost-shifting principles. Yet, those schemes are uncompromising in that they solve the problem exactly, or not at all, and often suffer from memory problems. In this work, we explore the well known principle of weighted search for converting best-first search solvers into anytime schemes. The weighted best-first search schemes report a solution early in the process by using inadmissible heuristics, and subsequently improve the solution. While it was demonstrated recently that weighted schemes can yield effective anytime behavior for pure MAP tasks, Marginal MAP is far more challenging (e.g., a conditional sum must be evaluated for every solution). Yet, in an extensive empirical analysis we show that weighted schemes are indeed highly effective anytime solvers for Marginal MAP yielding the most competitive schemes to date for this task.

Introduction

Graphical models provide a powerful framework for reasoning about conditional dependency structures over many variables. The Maximum a Posteriori (MAP) task asks for the mode of the joint probability distribution, while the Marginal MAP (MMAP) task generalizes MAP by allowing a subset of the variables to be marginalized.

MMAP is known to be a very challenging task. The difficulty rises from not only the exponential size of the search space but also the hardness of evaluating a full instantiation of MAP variables, i.e., computing the marginal probability given evidence. Furthermore, variable elimination orders of MMAP are constrained and often more costly because the maximization and the summation do not commute, i.e., sum variables are eliminated before any MAP variable.

Early work for solving MMAP exactly was based on the depth-first branch and bound search guided by a heuristic based on evaluating (incrementally) a join-tree built along an unconstrained variable ordering (Park and Darwiche 2003; Yuan and Hansen 2009). The depth-first branch and bound search improves its performance when it traverses an AND/OR search graph, and when it is guided by heuristics based on weighted mini-bucket elimination with cost shifting schemes, as shown in (Marinescu, Dechter, and Ihler 2014). More recently, several best-first AND/OR search algorithms for MMAP were also proposed, and were shown to outperform considerably depth-first search if memory is available (Marinescu, Dechter, and Ihler 2015). They include the best-first AND/OR search algorithm which belongs to the AO* family, as well as a recursive best-first AND/OR search algorithm that operates within limited memory. In addition, variational algorithms for MMAP (Liu and Ihler 2013), and an anytime MMAP algorithm based on factor-set-elimination (Maua and De Campos 2012) have also been developed.

Best-first search, which typically expands fewer nodes than depth-first search, is a particularly favorable choice for MMAP because it can save on the number of evaluations of the marginal probability conditioned on the MAP variables. However, best-first search faces two drawbacks: (1) it may require enormous amounts of memory, (2) it only returns the optimal solution at the end of processing. Recursive best-first search (Korf 1993) remedies the memory issue, but still lacks anytime behavior.

Contributions: Here, we focus on providing good anytime solutions, and develop the anytime version of the exact AND/OR search algorithms. We use the principle of weighted search that was shown to provide a useful tool for converting best first algorithms into anytime scheme (Pohl 1970). Unlike local search based methods (Park and Darwiche 2004), our proposed algorithms come with solution quality guarantees; unlike variational schemes (Liu and Ihler 2013) our algorithms will prove optimality if given enough time.

We discuss the effectiveness of the newly introduced anytime algorithms against state-of-the-art exact search algorithms over varying strength of heuristics and time bounds. Through extensive empirical evaluations, we show that the proposed anytime algorithms dominate exact search algorithms and provide anytime solutions to the problems that were beyond the reach of exact algorithms.

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. $\mathbf{F} = \{\psi_\alpha : \alpha \in F\}$ is a set of discrete positive real-valued local functions defined on subsets of variables, where we use $\alpha \subseteq V$ and $\mathbf{X}_\alpha \subseteq \mathbf{X}$ to indicate the *scope* of function ψ_α, i.e., $\mathbf{X}_\alpha = var(\psi_\alpha) = \{X_i : i \in \alpha\}$. The function scopes yield a *primal graph* whose vertices are the variables and whose edges connect any two variables that appear in the scope of the same function. The graphical 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 $\mathbf{X}_S$:

$$\mathbf{x}^*_M = \underset{\mathbf{x}_M}{\operatorname{argmax}} \sum_{\mathbf{x}_S} \prod_{\alpha \in F} \psi_\alpha(\mathbf{x}_\alpha) \tag{1}$$

AND/OR Search Spaces

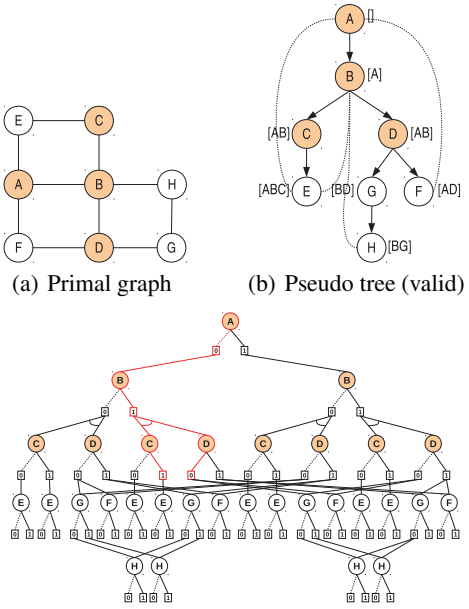
Significant recent improvements in search for MMAP inference have been achieved by using AND/OR search spaces, which often capture problem structure far better than standard OR search methods (Marinescu, Dechter, and Ihler 2014; Dechter and Mateescu 2007). The AND/OR search space is defined relative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

DEFINITION **1 (pseudo tree)** *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 included 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.*

We say that a pseudo tree $\mathcal{T}$ of G is *valid for MAP variables* $\mathbf{X}_M$ if $\mathcal{T}$ restricted to $\mathbf{X}_M$, denoted by $\mathcal{T}' = (V', E'')$ where $V' \subseteq V$, $E'' \subseteq E'$ and V' correspond to $\mathbf{X}_M$, forms a connected subgraph of $\mathcal{T}$ that has the same root as $\mathcal{T}$ and is called a *start pseudo tree*.

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 $\mathcal{T}$ (i.e., the constrained induced width).

DEFINITION **2 (MMAP solution tree)** *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 $\hat{x}_M$.*



(a) Primal graph (b) Pseudo tree (valid)

(c) Context minimal AND/OR search graph

Figure 1: A simple graphical model.

Each node n in $C_\mathcal{T}$ can be associated with a *value* $v(n)$; for MAP variables $v(n)$ is 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 an example of simple graphical model. Figure 1(a) is the primal graph of 8 bi-valued variables and 10 binary functions, where the MAP and sum variables are $\mathbf{X}_M = \{A, B, C, D\}$ and $\mathbf{X}_S = \{E, F, G, H\}$. Figure 1(b) is a valid pseudo tree whose MAP variables form a start pseudo tree. Figure 1(c) displays the context minimal AND/OR search graph based on the valid pseudo tree. (the contexts are shown next to the pseudo tree nodes). A MMAP solution subtree corresponding to the MAP assignment $(A = 0, B = 1, C = 1, D = 0)$ is shown in red.*

Earlier Work: Exact AND/OR Search for MMAP

Current state-of-the-art approaches for exact MMAP solving are based on either depth-first or best-first AND/OR search guided by mini-bucket heuristics enhanced with variational cost-shifting ideas (Marinescu, Dechter, and Ihler 2014; 2015).

AND/OR Branch and Bound (AOBB-MMAP) (Marinescu, Dechter, and Ihler 2014) explores in a *depth-first*

Algorithm 1: AOBF-MMAP

Input: Graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, pseudo tree $\mathcal{T}$, $\mathbf{X}_M = \mathbf{X} \setminus \mathbf{X}_S$, heuristic $h(\cdot)$
Output: Optimal MMAP value (and assignment)

Insert root n_0 in $C_\mathcal{T}$, where n_0 is labeled by the root of $\mathcal{T}$
Initialize $q(n_0) \leftarrow h(n_0)$ and let $T' = \{n_0\}$
while *true* **do**
  Select non-terminal tip node n in the best partial tree T'.
  if $tips(T') = \emptyset$ **then return** $(q(n_0), T')$
  foreach *successor* n' *of* n **do** // Expand
    if $n' \notin C_\mathcal{T}$ **then**
      Add n' as child of n in $C_\mathcal{T}$
    if n' *is OR node labeled by* $X \in \mathbf{X}_S$ **then**
      $q(n') \leftarrow eval(\mathcal{M}|_{T'})$
    else $q(n') \leftarrow h(n')$
  foreach *ancestor* m *of* n *in* $C_\mathcal{T}$ **do** // Update
    if m *is OR node* **then**
      $q(m) \leftarrow \max_{n' \in succ(m)} w(m, n') \cdot q(n')$
      Mark best successor n' of m as $n' = \text{argmax}_{n' \in succ(m)} w(m, n') \cdot q(n')$, maintaining marked successor if still best
    else $q(m) \leftarrow \prod_{n' \in succ(m)} q(n')$
  Recompute best partial tree T' by following marked arcs from the root n_0

manner the context minimal AND/OR search graph and therefore takes advantage of problem decomposition. During search, AOBB-MMAP keeps track of the value of the best solution found so far (a lower bound on the optimal MMAP cost) 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 in a typical branch and bound manner.

Best-First AND/OR Search (AOBF-MMAP) (Marinescu, Dechter, and Ihler 2015) is a variant of AO* (Nilsson 1980) applicable to graphical models that explores the context minimal AND/OR search graph in a *best-first* rather than depth-first manner. This enables AOBF-MMAP to visit a significantly smaller search space than AOBB-MMAP which sometimes translates into important time savings, as well as considerably fewer conditional likelihood evaluations. Extensive empirical evaluations (Marinescu, Dechter, and Ihler 2015) showed that when given enough memory AOBF-MMAP is often superior to AOBB-MMAP. For completeness, we provide the pseudo code of AOBF-MMAP in Algorithm 1.

Recursive Best-First AND/OR Search (RBFAOO-MMAP) was recently introduced to address the memory issues of AOBF-MMAP. RBFAOO-MMAP (described by Algorithm 2) extends Recursive Best-First Search (RBFS) (Korf 1993) to MMAP queries over graphical models and thus it uses a threshold controlling mechanism to drive the search in a depth-first like manner. Specifically, let $q(n)$, called the q-value, be an upper bound of the solution cost at node n, and $\theta(n)$ be the threshold at n indicating the availability of a second best solution besides $q(n)$. RBFAOO-MMAP keeps examining the subtree rooted at n

Algorithm 2: RBFAOO-MMAP

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

Procedure RBFAOO()
  Insert root n_0 in $C_\mathcal{T}$, where n_0 is labeled by the root of $\mathcal{T}$
  OrNode(n_0, 0)
  return $q(n_0)$
Function OrNode(n, θ)
  while *true* **do**
    foreach *AND child* c *of* n **do**
      if c *is in cache* **then** $q(c) \leftarrow ReadCache(c)$
      else $q(c) \leftarrow h(c)$
      $q(c) = w(n, c) \cdot q(c)$
    Update $q(n) \leftarrow \max_{c \in succ(n)} q(c)$ and mark n as solved if the child with the highest q-value is solved
    if $q(n) < \theta$ *or* n *is solved* **then**
      break
    Identify two children (n_1, n_2) with the two highest q values s.t. $q(n_1) \geq q(n_2) \geq q(others)$ and update threshold as $\theta' = max(\theta, q(n_2))/w(n, n_1)$
    AndNode(n_1, θ')
  $WriteCache(n, q(n))$
Function AndNode(n, θ)
  while *true* **do**
    foreach *OR child* c *of* n **do**
      if c *is in cache* **then** $q(c) \leftarrow ReadCache(c)$
      else if c *is labeled by* $X \in \mathbf{X}_S$ **then**
        $q(c) \leftarrow eval(\mathcal{M}|_{\bar{x}})$
      else $q(c) \leftarrow h(c)$
    Update $q(n) \leftarrow \prod_{c \in succ(n)} q(c)$, and mark n as solved if all its children are solved
    if $q(n) < \theta$ *or* n *is solved* **then**
      break
    Identify an unsolved OR child n_1 and update threshold $\theta' = \theta \cdot q(n_1)/q(n)$
    OrNode(n_1, θ')
  $WriteCache(n, q(n))$

until either $q(n) < \theta(n)$ or the subtree is solved optimally. Therefore, it gradually grows the search space by updating the q-values of the internal nodes and re-expanding them, unlike AOBF-MMAP. The algorithm may operate in linear space (no caching); however, for efficiency, it may use a fixed size cache table to store some of the nodes (based on contexts). In practice, RBFAOO-MMAP outperformed dramatically both AOBB-MMAP and AOBF-MMAP on a variety of benchmarks (Marinescu, Dechter, and Ihler 2015).

Anytime AND/OR Search for MMAP

A serious drawback of these highly competitive best-first search schemes is that they lack anytime behavior, namely they either solve the problem exactly, or not at all. In many situations, it is desirable to obtain a good solution relatively quickly and spend the remaining time improving it. Therefore, in this paper we explore the potential of the well known principle of weighted search and convert the best-first search algorithms into anytime search schemes for Marginal MAP.

Weighted best-first AND/OR search was introduced re-

cently as an effective alternative to anytime depth-first search schemes for pure MAP tasks (Flerova, Marinescu, and Dechter 2014). These algorithms were evaluated extensively on a wide range of benchmark problems and were shown to be very powerful and highly competitive with Breadth Rotating AOBB search (Otten and Dechter 2011), one of the best performing anytime MAP solvers. The anytime best-first AND/OR search algorithms for MMAP presented in this paper are direct extensions of the weighted best-first search algorithms for pure MAP (Flerova, Marinescu, and Dechter 2014).

Weighted AOBF-MMAP and RBFAOO-MMAP The fixed-weighted version of the AOBF-MMAP and RBFAOO-MMAP algorithms can be obtained easily by multiplying[1] the heuristic function $h(n)$ of a node n in the AND/OR search graph by a weight $w > 1$ (i.e., substituting $h(n)$ by $w \cdot h(n)$). Notice that only the portion of the search space that corresponds to the MAP variables is considered. If $h(n)$ is admissible, which is the case for mini-bucket heuristics, then the cost of the solution discovered by weighted AOBF-MMAP (or weighted RBFAOO-MMAP) is w-optimal, namely it is guaranteed to be within a factor w from the optimal one. These schemes extend well known approaches such as WA* (Pohl 1970) and WAO* (P. Chakrabarti 1987) to MMAP queries.

Algorithm 3: WAOBF-MMAP

Input: Graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, pseudo tree $\mathcal{T}$, $\mathbf{X}_M = \mathbf{X} \setminus \mathbf{X}_S$, heuristic $h(\cdot)$, initial weight w_0
Output: **w-optimal** MMAP value $\mathcal{V}(T)$, and assignment T

1 Initialize $w = w_0$, $\mathcal{V}(T) = -\infty$, $T = \emptyset$
2 **while** $w >= 1$ **do**
3 $\quad (\mathcal{V}', T') \leftarrow$ **AOBF-MMAP**$(\mathcal{M}, w \cdot h)$
4 $\quad$ Maintain and report current best solution as $(\mathcal{V}(T), T)$
5 $\quad$ Decrease weight according to schedule policy
6 **return** $(\mathcal{V}(T), T)$

Iterative Weighted AOBF-MMAP Since weighted AOBF-MMAP yields w-optimal solutions, it can then be extended into an iterative anytime scheme called WAOBF-MMAP (Algorithm 3), by decreasing the weight from one iteration to the next (until it becomes 1).

Specifically, WAOBF-MMAP starts by running AOBF-MMAP with the initial weight w_0 until it finds a suboptimal solution, and restarts the search with a decreased weight following a weight update schedule policy such as $w_{i+1} = \sqrt{w_i}$. Clearly, different schedules are also possible, however, the latter proved quite effective in practice (see also (Flerova, Marinescu, and Dechter 2013) for additional details). Notice that the algorithm also output a w-optimality guaranty.

Iterative Weighted RBFAOO-MMAP The anytime weighted RBFAOO-MMAP, called here WRBFAOO-MMAP, is obtained by replacing the call to AOBF-MMAP in line 3 of Algorithm 3 with a call to RBFAOO-MMAP.

[1]For numerical stability all algorithms presented solve Eq. 1 in log space, as an energy minimization problem.

Algorithm 4: WRAOBF-MMAP

Input: Graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, pseudo tree $\mathcal{T}$, $\mathbf{X}_M = \mathbf{X} \setminus \mathbf{X}_S$, heuristic $h(\cdot)$, initial weight w_0
Output: **w-optimal** MMAP value $\mathcal{V}(T)$, and assignment T

1 Initialize $w = w_0$, $C'_{\mathcal{T}} = \{s\}$, $\mathcal{V}(T) = -\infty$, $T = \emptyset$
2 **while** $w >= 1$ **do**
3 $\quad (\mathcal{V}', T', C'_{\mathcal{T}}) \leftarrow$ **AOBF-MMAP**$(\mathcal{M}, w \cdot h, C'_{\mathcal{T}}, T)$
4 $\quad$ Maintain and report current best solution as $(\mathcal{V}(T), T)$
5 $\quad$ Decrease weight according to schedule policy
6 $\quad$ Revise $C'_{\mathcal{T}}$ according to $w \cdot h$ from leaves to the root
7 $\quad$ Update the best partial solution tree T from revised $C'_{\mathcal{T}}$
8 **return** $(\mathcal{V}(T), T)$

Anytime Repairing AOBF-MMAP Running WAOBF-MMAP from scratch at each iteration seems redundant. Therefore, we introduce another anytime scheme, called WRAOBF-MMAP, that is based on the anytime repairing AOBF approach for pure MAP (Flerova, Marinescu, and Dechter 2014). Algorithm 4 describes WRAOBF-MMAP. As the name suggests, it does not discard the explicated AND/OR search graph $C'_{\mathcal{T}}$ after finishing the i-th iteration but rather it revises the node values in $C'_{\mathcal{T}}$ from leaves to the root node according to the newly inflated heuristic $w_{i+1} \cdot h(n)$. The search continues with the revised best partial solution tree.

Anytime AND/OR Branch and Bound Search Depth-first AND/OR Branch and Bound often lacks good anytime behavior because during search AOBB will solve to completion all but one independent subproblems rooted at an AND node. This behavior was first observed by (Otten and Dechter 2011) in the context of pure MAP inference. Therefore, Breadth Rotating AOBB (BRAOBB) was introduced as an anytime depth-first AND/OR search scheme that rotates through different subproblems in a round-robin manner. Extensive empirical evaluations showed that BRAOBB improves considerably AOBB's anytime behavior (Otten and Dechter 2011). We describe next algorithm BRAOBB-MMAP which is a direct application of the technique to MMAP queries.

Algorithm 5 summarizes BRAOBB-MMAP, omitting the details about AOBB-MMAP (i.e., pruning, caching and propagation). For these details, see (Marinescu, Dechter, and Ihler 2014).

The algorithm maintains not only the $LOCAL$ stack but also a $GLOBAL$ queue to store and rotate the subproblems. Notice that rotation applies only to MAP variables. A separate stack called $LOCAL_S$ is used to perform an exhaustive depth-first exploration of the summation subproblems (lines 3-4). BRAOBB-MMAP expands the chain portions of the pseudo tree corresponding to the MAP variables until the rotation limit R is reached or the current subproblem is fully solved (line 9). If the rotation limit R is reached before solving the current subproblem, the nodes in $LOCAL$ are moved to the $GLOBAL$ queue, and visited later. The partial solution tree T' as well as the best solutions found so far need to be maintained according to the node expansion (lines 6-8). The algorithm returns the optimal solution upon

Algorithm 5: BRAOBB-MMAP

Input: Graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, pseudo tree $\mathcal{T}$, $\mathbf{X}_M = \mathbf{X} \setminus \mathbf{X}_S$, heuristic $h(\cdot)$, rotation limit R
Output: the anytime MMAP value $\mathcal{V}(T)$, and assignment T

Insert $root$ to $GLOBAL$ queue (FIFO)
while $GLOBAL \neq \emptyset$ **do**
  if $LOCAL_S \neq \emptyset$ **then**
    DFS traversal below n using stack $LOCAL_S$
  $LOCAL_M \leftarrow front(GLOBAL)$
  Maintain partial MMAP assignment T'
  if T' *is a complete MMAP assignment* **then**
    Maintain and report best solution so far as $(\mathcal{V}(T), T)$
  for $r \leftarrow 1$ *to* R *or* ***until*** $LOCAL = \emptyset$ *or* ***until*** $childSubprob(LOCAL) \neq \emptyset$ **do**
    Pop the top node n in $LOCAL$
    if $n = \langle X_i \rangle$ *is OR node* **then**
      foreach $x_j \in D_i$ **do**
        Create AND node $n' = \langle X_i, x_j \rangle$
        Push n' to top of $LOCAL$
    else if $n = \langle X_i, x_j \rangle$ *is AND node* **then**
      $Y_1 \dots Y_m \leftarrow children_{\mathcal{T}}(X_i)$
      Create OR nodes $\langle Y_1 \rangle \dots \langle Y_m \rangle$
      if $m = 1$ **then**
        Push $\langle Y_1 \rangle$ to top of $LOCAL$
      else if $m > 1$ **then**
        for $j \leftarrow 1$ *to* m **do**
          if $Y_j \in \mathbf{X}_M$ **then**
            $NEW \leftarrow \{\langle Y_j \rangle\}$
            Push NEW to back of $GLOBAL$
          else if $Y_j \in \mathbf{X}_S$ **then**
            Push $\langle Y_j \rangle$ to top of $LOCAL_S$
            break
    if $succ(n) = \emptyset$ **then** Propagate values upward
  if $LOCAL \neq \emptyset$ **then** Push $LOCAL$ to back of $GLOBAL$
return $(\mathcal{V}(T), T)$

completion or an anytime solution if interrupted.

Empirical Evaluation

We evaluate empirically the newly introduced anytime AND/OR search algorithms and compare them with the exact algorithms on problem instances generated from the PASCAL2 Inference Challenge benchmarks (Elidan, Globerson, and Heinemann 2012). All competing algorithms use the static weighted mini-bucket heuristic with moment matching, WMB-MM(i), whose strength can be controlled by a parameter i-bound (Dechter and Rish 2003; Liu and Ihler 2011). The experiment supports different levels of strength for the guiding heuristics, from weak to strong, and time bounds up to 2 hours with 24 GB memory. The starting weight was 64 for WAOBF, WRAOBF, and WRBFAOO, and the overestimation parameter was 1 for RBFAOO and WRBFAOO. For the 5 best-first algorithms, the size of the cache table was 4 GB, whereas for the 2 depth-first branch and bound algorithms the size of the cache table was only limited by the total memory limit of 24 GB for a fair comparison.

Our benchmark is a subset of PASCAL2 benchmark on 3 domains: `grid` with 15 networks; `pedigree` and `promedas` with 10 networks. For each network, we generated 5 MMAP problem instances where half of the variables are MAP variables: 1 easy instance such that the MAP variables were selected from a breadth-first traversal of a pseudo tree obtained from an unconstrained min-fill ordering; and 4 hard instances where the MAP variables were selected uniformly at random.

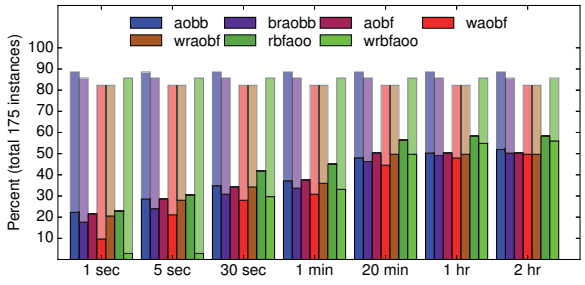


Figure 2: Percentage of instances solved (i=18). The height of the bottom bar represents the percent of instances solved optimally, and the height of the stacked bar represents the percent of instances with any solution.

To compare anytime performance, we consider responsiveness and the solution quality: how quickly the algorithm can generate a solution (responsiveness), and how good is the solution provided by a given time bound (quality). The responsiveness was measured by the coverage of each algorithm in terms of optimal and anytime solutions at varying time bounds. Specifically, anytime coverage refers to the percentage of problem instances with a solution within the time bound, whereas optimal coverage only accounts for the optimally solved instances. The quality of anytime solution for each algorithm was evaluated by counting the number of problem instances with better anytime solutions. The metric for evaluating the score of each algorithm will be given later in this section.

Performance Regimes of MMAP Algorithms

In Table 1, we summarize the anytime performance regimes of 7 MMAP algorithms with respect to the responsiveness at 1 hour time bound with WMB-MM($i = 18$) heuristics, and the quality of anytime solutions at 1 hour time bound with WMB-MM($i = 12$) heuristics on 3 benchmark domains.

RBFAOO and AOBF are the two worst performing algorithms in terms of both anytime responsiveness and quality of solution. Algorithms WRBFAOO and WAOBF are the best performing algorithms in terms of solution quality, and the anytime responsiveness of both algorithms was close to that of depth-first search based algorithms, showing the benefit of weighted best-first search.

Responsiveness of MMAP Algorithms

In Figure 2, we present both optimal and anytime coverage results for 7 MMAP algorithms over varying time bounds from 1 second to 2 hours. Note that, all algorithms shown in Figure 2 used the WMB-MM($i = 18$) heuristic. Considering the optimality of the solution at 2 hour time bound,

		grid (75 instances)		pedigree (50 instances)		promedas (50 instances)		overall (175 instances)	
MMAP algorithms		responsiveness	quality score	responsiveness	quality score	responsiveness	quality score	responsiveness	quality score
exact	AOBB	93%	293%	84%	342.00%	**86%**	**405%**	**89%**	339%
	AOBF	69%	209%	30%	158.00%	42%	258%	50%	208%
	RBFAOO	79%	58%	44%	95.00%	42%	132%	58%	90%
anytime	WAOBF	97%	**379%**	**88%**	**442.00%**	54%	266%	82%	**365%**
	WRBFAOO	**100%**	**422%**	**90%**	**440.00%**	60%	305%	**86%**	**394%**
	WRAOBF	97%	374%	88%	364.00%	54%	261%	82%	339%
	BRAOBB	**99%**	365%	58%	259.00%	**94%**	**473%**	**86%**	**365%**

Table 1: Anytime performance regimes for each of 7 evaluted algorithms. 3 exact MMAP algorithms (AOBB-MMAP, AOBF-MMAP, RBFAOO-MMAP) and 4 new anytime algorithms (BRAOBB-MMAP, WAOBF-MMAP, WRBFAOO-MMAP, WRAOBF-MMAP) are evaluated in grid, pedigree, and promedas domain. Responsiveness shows the percentage of instances solved at 1 hour time bound with WMB-MM($i = 18$) heuristic. Quality score is the sum of individual quality scores against other 6 algorithms that compare the quality of anytime solutions at 1 hour time bound with WMB-MM($i = 12$) heuristic. The metric for the score is defined in the following section.

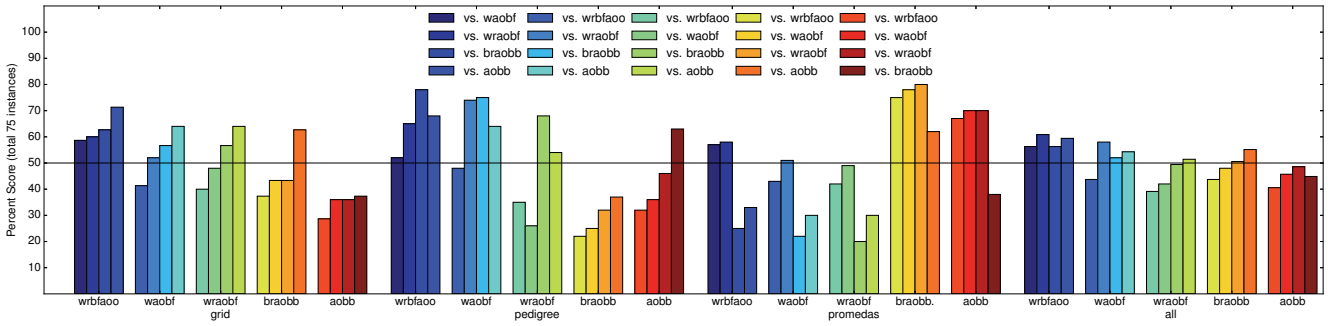


Figure 3: The score of each algorithm against others (t=1 hr, i=12). There are 4 groups correspond to the benchmark domain: `grid`, `pedigree`, `promedas`, and `all`. In each group, there are 5 subgroups, each of them represents an algorithm score against the other 4 algorithms. The winner has a score higher than 50. For example, the scores of the WRBFAOO in `grid` against WAOBF, WRAOBF, BRAOBB, and AOBB are displayed at 4 left most bars, and it outperformed all other 4 algorithms.

RBFAOO found the optimal solution on the largest number of instances (59%), WRBFAOO followed it by 56%, while the remaining algorithms proved optimality for about 50%. Clearly, the newly introduced WRBFAOO, WAOBF, and WRAOBF provide a significant improvement in terms of responsiveness as they readily reported a solution on more than 80% of the instances at 1 second time bound, which is close to AOBB and BRAOBB, respectively.

Solution Quality of MMAP Algorithms

Figure 3 reports the solution quality obtained by 5 MMAP algorithms (WRBFAOO, WAOBF, WRAOBF, BRAOBB, and AOBB) at 1 hour time bound. The i-bound was $i = 12$, which corresponds to guiding heuristics with moderate strength. The score is computed as follows. We record the solutions for each pair of algorithms A and B at a fixed time bound t. If A won against B (i.e., A's solution was better than B's) on n instances and tied on m instances, the score of A against B is $\frac{n+\frac{m}{2}}{N}$, where N is the total number of instances; the score of B against A is $\frac{N-n-\frac{m}{2}}{N}$. In `grid` and `pedigree`, WRBFAOO is the winner, WAOBF is second, and WRAOBF is third. On the other hand, BRAOBB is the best algorithm in `promedas`, and AOBB is second. The overall score across all domains shows that WRBFAOO is the winner, WAOBF is second, while AOBB is the worst among the 5 competing algorithms.

The ranking between algorithms presented above remains the same on different settings with varying time bounds and i-bounds, which is not shown in this paper due to the space limit. BRAOBB is the best performer only in the extreme cases, typically characterized by very short time bounds ($t <$ 1 sec) or uninformative heuristics ($i < 6$),

Example: `grid` instance `75-20-5-I4` in Table 2 is a case where weighted best-first search algorithms find a good solution in a short time bound. All three algorithms found the optimal solution in less than 1 minute, although they failed to prove optimality when the guiding heuristic was weak.

Comparing WRBFAOO and BRAOBB

We investigate further the anytime performance of WRBFAOO and BRAOBB based on two problem intrinsic parameters, the constrained induced width w_c and the conditioned induced width w_s of the summation problem only. Figure 4 visualizes the difficulty of individual problem instances and the winner at each point.

		i=12			i=18		
instance	algorithm	60 sec	600 sec	status	60 sec	600 sec	status
(n,m,k,w_c,w_s)		sol | and map | and (wt)	sol | and map | and (wt)	(time)	sol | and map | and (wt)	sol | and map | and (wt)	(time)
grid	braobb	-48.4052 | 2278014 | 2737734	-35.233 | 21747732 | 28300934	t(7200)	-22.243 | 2708481 | 2984371	-21.7217 | 13098051 | 13978657	o(230)
75-20-5-I4	waobf	-21.7217 | 623084 | 1.13879	-21.7217 | 7812572 | 1.13879	m(999)	-21.7217 | 22642 | 1.01638	-21.7217 | 7949237 | 1.00000	o(581)
(400,200,2,123,6)	wraobf	-21.7217 | 923220 | 1.13879	-21.7217 | 11970690 | 1.13879	m(934)	-21.7217 | 721493 | 1.01638	-21.7217 | 1396170 | 1.00000	o(104)
	wrbfaoo	-21.7217 | 3874836 | 1.13879	-21.7217 | 46299595 | 1.06714	m(711)	-21.7217 | 4430190 | 1.01638	-21.7217 | 17433182 | 1.00000	o(218)
grid	braobb		-37.864 | 146573 | 106869149	t(7200)		-21.0226 | 12683535 | 92577919	o(2479)
90-30-5-I0	waobf	| 484071 | 64.00000	| 8883291 | 64.00000	m(1351)	-78.7471 | 366757 | 64.00000	-21.1774 | 101028 | 1.13879	t(7200)
(900,450,2,45,19)	wraobf	| 482878 | 64.00000	| 8788862 | 64.00000	m(1279)	| 366901 | 64.00000	-21.3243 | 8521297 | 1.06714	o(917)
	wrbfaoo	| 6468703 | 64.00000	| 78908803 | 64.00000	t(7200)	| 1729 | 64.00000	-21.351 | 850834 | 1.29684	o(2537)
pedigree	braobb	-349.143 | 1628499 | 3623860	-347.953 | 19467699 | 33360344	t(7200)	-332.024 | 2127583 | 3547236	-332.024 | 21130781 | 30105390	t(7200)
pedigree39-I4	waobf	-337.491 | 1044653 | 2.82843	-335.863 | 5115966 | 1.29684	m(1115)	-323.707 | 768400 | 1.29684	-323.707 | 8180376 | 1.13879	m(930)
(1272,636,4,75,4)	wraobf	-323.651 | 768237 | 1.06714	-323.651 | 9644430 | 1.06714	m(845)	-332.04 | 745269 | 1.13879	-312.892 | 9861013 | 1.00816	m(819)
	wrbfaoo	-319.804 | 3541822 | 1.06714	-319.804 | 45251925 | 1.06714	t(7200)	-319.282 | 4529426 | 1.03302	-312.671 | 64996296 | 1.00203	m(931)
promedas	braobb	-77.5966 | 2176769 | 2208387	-71.5164 | 25231278 | 25278598	t(7200)	-30.0158 | 2035020 | 2075806	-30.0158 | 32982167 | 33102113	t(7200)
orchain22fg-I3	waobf	| 651909 | 64.00000	| 6917951 | 64.00000	m(911)	-84.3913 | 606502 | 64.00000	-84.3913 | 6474805 | 8.00000	m(992)
(1044,522,2,196,5)	wraobf	| 651897 | 64.00000	| 6884019 | 64.00000	m(876)	-84.3913 | 603704 | 64.00000	-84.3913 | 9318074 | 8.00000	m(743)
	wrbfaoo	| 4765920 | 64.00000	| 50921981 | 64.00000	t(7200)	-84.3913 | 4387947 | 64.00000	-84.3913 | 50995351 | 8.00000	t(7200)
promedas	braobb	-25.3431 | 3690522 | 3706122	-25.3431 | 42605032 | 42637522	t(7200)	-25.3147 | 3732389 | 3749229	-19.5736 | 42385217 | 42421823	t(7200)
orchain8fg-I2	waobf	| 1209866 | 64.00000	| 12649848 | 64.00000	m(811)	-48.7187 | 1114238 | 64.00000	-45.0764 | 11186511 | 8.00000	m(1525)
(1195,597,2,53,4)	wraobf	| 1193825 |64.00000	| 12605159 | 64.00000	m(769)	| 1093969 | 64.00000	-48.7187 | 13252856 | 2.82843	m(686)
	wrbfaoo	| 6506800 | 64.00000	| 69469031 | 64.00000	t(7200)	-86.1035 | 6584968 | 8.00000	-31.2978 | 67551872 | 2.82843	t(7200)

Table 2: Search statistics of anytime algorithms for `grid`, `pedigree`, and `promedas` instances. n is the number of variables, m is the number of MAP variables, k is the maximum domain size, w_c is the constrained induced width, and w_s is the conditional induced width of the summation problem only. The current best solution in log scale, sol, the number of AND map nodes, $and\ map$, are displayed at two time bounds, 60 seconds and 10 minutes, and two i-bounds, 12 and 18. In addition, the number of AND nodes, and, is shown for BRAOBB, while the weight, wt, is shown for the weighted best-first algorithms. The $t(7200)$ denotes time out at 7200 seconds, $m(t)$ denotes out of memory at time t, and $o(t)$ indicates optimality found at time t.

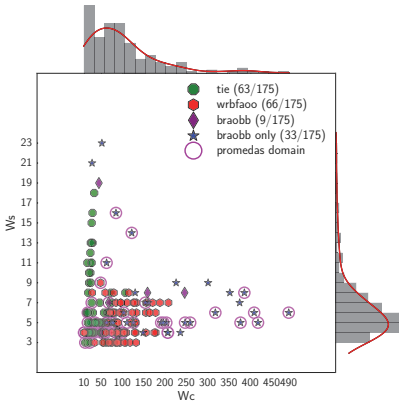


Figure 4: Distribution of w_c and w_s over our benchmark instances, and comparison between WRBFAOO and BRAOBB (t =1 hr, i=12). Each point's location (w_c, w_s) indicates the instance's difficulty. The circled points are `promedas`, where 23 instances out of 50 were solved by BRAOBB only.

Tied Region ($\mathbf{w_c} \in [10, 58]$): WRBFAOO and BRAOBB are tied in this region. WRBFAOO failed to report a single solution on 2 instances with $w_s > 20$ due to the overhead of computing the exact solution to the summation subproblem.

WRBFAOO Region ($\mathbf{w_c} \in [60, 200]$): WRBFAOO dominates BRAOBB, consistent with the ranking in Figure 3. WRBFAOO failed to report any solution when $w_s > 10$.

BRAOBB Region ($\mathbf{w_c} \in [202, 490]$): Only BRAOBB reported a solution in this region, except for 2 instances. Such extremely hard instances are mostly `promedas`, consistent with BRAOBB being a winner on `promedas` in Figure 3.

Note that, the time bound for determining the winner was 1 hour and the i-bound of the WMB-MM heuristic was 12. The area of each region changes over different combinations of the time bounds and the i-bounds, but the trend remains the same, which is not presented here due to space reasons.

Example: In Table 2, WRBFAOO showed the best anytime performance in `pedigree39-I3`. However, best-first algorithms couldn't find the first solution for two `promedas` instances when the guiding heuristic was relatively weak.

Comparing Weighted Best-First Algorithms

We next explore the anytime behavior of the weighted best-first search algorithms based on the average weight at different time bounds. All algorithms used the $sqrt$ policy with the initial weight 64 (i.e., $w_{i+1} = \sqrt{w_i}$).

WRBFAOO: Figure 5 shows that WRBFAOO has the lowest average weight after 5 second on both strong and relatively weak heuristics, consistent with the ranking in Figure 3. This implies that the depth-first like node expansion strategy of WRBFAOO can not only avoid the memory issues but also allows the algorithm to report a tighter w-optimal solution.

WAOBF vs. WRAOBF: For shorter time bounds ($t < 30$), the weight of WRABOF decreased faster than WAOBF and stronger heuristics foster the speed. This implies that repairing schemes save on the number of node expansions and find tighter w-optimal solutions when w is high ($w \geq 8$).

On the other hand, the average weight of WAOBF and WRAOBF are indistinguishable at longer time bounds. This implies that repairing the entire explicated search space incurs severe overhead, losing its advantage against expanding the search space afresh. Furthermore, the solution quality

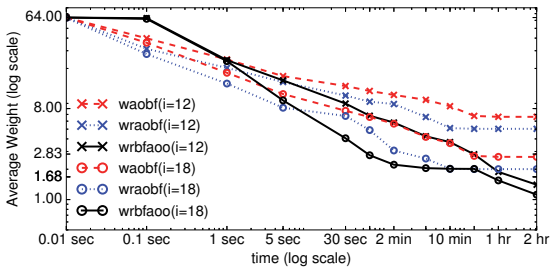


Figure 5: Average weights (i =12, i =18). There are two groups of curves with respect to the i bound. Each curve shows the average weight over *instances with a solution at 2 hours*.

of WAOBF is superior to WRAOBF in Figure 3. This implies that restarting scheme effectively escapes from unfruitful search space by discarding the explicated search space.
Example: In Table 2, WRAOBF found the optimal solution for `90-30-5-I0` more than 2 times faster than other algorithms. We can see that the weight of WRAOBF at 10 minute was the smallest compared to the other two algorithms.

Conclusion

We introduced new anytime search algorithms for MMAP that explore the context-minimal AND/OR search space for graphical models, either depth-first or best-first, and are guided by static weighted mini-bucket heuristics with variational cost-shifting. Our extensive empirical evaluation showed that weighted best-first AND/OR search algorithms have a particularly effective anytime performance for MMAP in terms of both responsiveness and the quality of solutions. In addition, these schemes were superior to anytime depth-first search for relatively weak heuristics. Nevertheless, depth-first search is the only possibility when the heuristic is very weak, or the conditional likelihood evaluation is hard. For future work, we plan to explore approximations for the summation sub-task and hybrid search schemes that combine the various anytime algorithms we introduced.

Acknowledgments

We thank the reviewers for their valuable feedback. This work was supported in part by NSF grants IIS-1065618, IIS-1526842 and IIS-1254071, and by the US Air Force under Contract No. FA8750-14-C-0011 under the DARPA PPAML program.

References

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.

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

Flerova, N.; Marinescu, R.; and Dechter, R. 2013. Anytime AND/OR best-first search for optimization in graphical models. In *ICML 2013 Workshop on Inferning: Interactions between Inference and Learning*.

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

Korf, R. 1993. Linear-space best-first search. *Artificial Intelligence* 62:41–78.

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 De Campos, C. 2012. Anytime marginal MAP inference. In *International Conference on Machine Learning*, 1471–1478.

Nilsson, 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 *International Symposium on Combinatorial Search*, 117–702.

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

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

Park, J., and Darwiche, A. 2004. Complexity results and approximation strategies for MAP explanations. *Journal of Artificial Intelligence Research* 21(1):101–133.

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

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.