AND/OR Search Spaces for Graphical Models

Rina Dechter
Department of Information and Computer Science
University of California, Irvine, CA 92697-3425
{dechter }@ics.uci.edu

Abstract

The paper introduces an AND/OR search space perspective for graphical models that include probabilistic networks (directed or undirected) and constraint networks. In contrast to the traditional (OR) search space view, the AND/OR search tree displays some of the independencies present in the graphical model explicitly and may sometime reduce the search space exponentially. Indeed, most algorithmic advances in search-based constraint processing and probabilistic inference can be viewed as searching an AND/OR search tree or graph. Familiar parameters such as the depth of a spanning tree, tree-width and path-width are shown to play a key role in characterizing the effect of AND/OR search graphs vs the traditional OR search graphs.

1 Introduction

Graphical models such as constraint networks, Bayes networks, Markov Random fields and influence diagrams are knowledge representation languages that capture independencies in the knowledge and allow both concise representation and efficient graph-based algorithms for query processing. Algorithms for processing graphical models are of two types: inference-based or search-based. The latter class typically traverses the problem's search space, where each path represents a partial or a full solution. In contrast to inference algorithms which exploit the independencies in the underlying graphical model effectively (e.g. variable elimination, tree-clustering), pure search is at risk of loosing this information because it is hidden in the linear structure of the search space. Advanced search algorithms developed for constraint satisfaction, and more recently for probabilistic reasoning can be viewed as attempting to overcome this difficulty.

Overall, Inference algorithms provide the best worst-case time guarantees. They are time exponential in the tree-width of the graph model. This guarantee comes with a memory price. Any method that is bounded exponential in the tree-width time-wise, requires in the worst-case, space exponential in the tree-width as well. This limitation is severe, making inference-based algorithms infeasible for large tree-width (above 20). Search algorithms on the other hand accommodates a spectrum of bounded memory algorithms, from linear space to tree-width bounded space. By their nature they are likely to take more time then inference because they are restricted in memory.

This paper is focused on search. We propose to harness the well known notion of *AND/OR search spaces* developed for heuristic search [Nillson1980], for processing queries over graphical models. We demonstrate how this principle can exploit independencies in the graph model to yield AND/OR search trees that are exponentially smaller than the corresponding traditional *OR* search tree. In particular, we show that the size of an AND/OR search tree is bounded exponentially by the depth of a tree spanning the graph-model. This implies that any *linear space* search algorithm which traverses the AND/OR space, is bounded exponentially by the tree-depth, saving exponential time relative to OR search trees. We subsequently extend the AND/OR search space from *search trees* to *search graphs*. Algorithms exploring the search space graph can engage in controlled memory management and cease to be linear space. Yet, their time performance can be reduced substantially and monotonically with their memory use.

The AND/OR search spaces view, provides a coherent account of various advanced search methods for graphical model, and has a potential to yield similar advances in any new graphical model and its queries. Known algorithms for probabilistic inference (e.g., recursive conditioning [Darwiche1999, F. Bacchus & Piassi2003]) for constraint satisfaction (caching goods and no-goods and backjumping [Dechter1990, Bayardo & Miranker1996]), for optimization [Terrioux & Jegou2003] can be placed within the AND/OR search space context. Finally, relationship to variable-elimination and OBDD compilation are discussed.

Following some preliminaries (Section 2) we present the notion of AND/OR search tree for graphical models and (section 3). Section 4 introduces minimal AND/OR search graphs and provide analysis. Typical algorithms exploring the AND/OR search tree and graph for probabilistic inference, are introduced (section 5). Section 6 provides discussion and related work.

2 Preliminaries and Background

A **Reasoning graphical-model is a triplet** $\mathcal{R} = (X, D, F)$ where X a set of variables $X = \{X_1, ..., X_n\}$, D is their respective domains of values $D = \{D_1, ..., D_n\}$ and F is a set of real-valued functions $F = \{F_1, ..., F_t\}$. Each function F_i is defined over a subset of variables S_i, called its scope, $S_i \subseteq X$. The primal graph of a reasoning problem, has a node for each variable, and any two variables appearing in the same function's scope are connected.

In **constraint network** $\mathcal{R} = (X, D, C)$ the functions F are constraints C. Each constraint is a pair $C_i = (S_i, R_i)$, where $S_i \subseteq X$ is the scope of the relation R_i defined over S_i, denoting the allowed combination of values. The primary queries over constraint network is to determine if the network is consistent, and if it is, to find, one or all solutions. A related task is to compute the number of solutions.

A **belief network** is a graphical model defined over a directed acyclic graph G over the variables X, and the the functions F are $P = \{P_i\}$, denoting conditional probability tables (CPTs) $P_i = \{P(X_i|pa_i)\}$, and pa_i is the set of *parent* nodes pointing to X_i in G. The belief network represents a probability distribution over X having the product form $P(\bar{x}) = P(x_1, ..., x_n) = \Pi_{i=1}^{n} P(x_i|x_{pa_i})$. $\bar{x}[S]$ denotes the restriction of a tuple x over a subset of variables S. The primary query over belief networks is *belief updating* namely determining the posterior marginals of subsets of variables given evidence.

Flat CPTs. Each CPT that has some zero probability entries expresses implicitly a constraint. The *flat* constraint of CPT P_i is a constraint R_i over its scope s.t (X_i, pa_i) is a no-good of R_i (not in the relation) iff $P_i(x_i|pa_i) = 0$. In this paper, when we talk about a constraint networks we refer also to any graphical model (e.g., belief networks) through the set of flat constraints that can be obtained from the CPTs.

Induced-graphs, induced width and path-width. An *ordered graph* is a pair (G, d) where G is an undirected graph, and $d = X_1, ..., X_n$ is an ordering of the nodes. The *width of a node* is the number of the node's neighbors that precede it in the ordering. The *width of an ordering* d, is the maximum width over all nodes. The *induced width of an ordered graph*, $w^*(d)$, is the width of the induced ordered graph obtained as follows: nodes are processed from last to first; when node X is processed, all its preceding neighbors are connected. The *induced width of a graph*, $w*$, is the minimal induced width over all its orderings. The set of maximal cliques (also called clusters) in the induced graph provide a tree-decomposition of the graph. The tree-width is the maximal number of variables in a cluster of an optimal cluster-tree decomposition of the graph [Arnborg1985]. It is well known that the induced-width of a graph is identical to its *tree-width*. The path-width of a graph is the smallest induced-width along a chain-like decomposition. The optimal path-width is denoted pw^*. For various relationship see [Arnborg1985, Bodlaender & Gilbert1991]. **AND/OR Search spaces.** An AND/OR state space representation has OR states and AND states, and a set of operators O. An OR operator transforms an OR state to another state, and an AND operator transforms and AND state to a set of states. There is a set of goal states $S_g \subseteq S$ and a start node s_0. Example problem domains modelled by AND/OR graphs are two-player games, parsing sentences, Tower of Hanoi.

The AND/OR states space model induces an explicit AND/OR search *graph*. Each node is a state and its child nodes are those obtained by applicable AND or OR operators. The search graph includes a *start* node. The terminal nodes (having no child nodes) are marked as Solved (S), or Unsolved (U).

A **solution subgraph** of an AND/OR search graph G is a subtree which 1. contains the start node s_0, 2. if n in the subtree is an OR node then it contains one of its child nodes in G and if n is an AND nodes it contains all its children in G. 3. All its terminal nodes are "Solved" (S). The primary tasks defined over an AND/OR graph is to determine the value of the root node, and if it is solved, to find a solution subtree with optimal cost if a cost is defined.

3 AND/OR Search Tree for Graphical Models

Consider a graphical model (e.g. a belief or a constraint network) $\mathcal{R} = (X, D, F)$ and its primal graph G. Let T be a DFS spanning tree of its primal graph rooted at X_0. For each node Y, $ch(Y)$ are its set of child nodes in T.

DEFINITION **1 (AND/OR search tree based on DFS tree)** *Given a graphical model $\mathcal{R}$ and it DFS spanning tree T rooted at X_0, the AND/OR search tree of $\mathcal{R}$ based on T, denoted $S_T(\mathcal{R})$ (or just S_T when $\mathcal{R}$ is unambiguous) is defined as follows. The nodes in the search tree are either OR nodes (e.g., X, Y), denoting variables, or AND nodes denoting Variable-value assignment pairs (e.g., $< X, v >$). The path of $< X, v >$ is the path from the initial state (the root X_0) to $< X, v >$, which corresponds to a* consistent *partial value assignments to all the variables along the path. The successors of a node in the AND/OR search tree are defined as follows:*

- *The successor-nodes of an OR node X are all its possible value assignments $\{< X, v > | v \in D_X\}$ that are consistent along the path to $< X, v >$. (The path alternates OR and AND nodes like ($X_0, < X_0, v_0 >, X_1, < X_1, v_1 >, ..., X_i, < X_i, v_i > ...$)*
- *The successor-nodes of an AND node $< X, v >$ are all its child nodes, $ch(X)$, in T.*

The consistency of a partial path is determined by considering all the relevant (or flat) constraints whose scopes are contained in the path.

DEFINITION **2 (Basic labeling of AND/OR nodes)** *Given an AND/OR tree $S_T(\mathcal{R})$, A terminal AND node (no child nodes in T) is always labeled "Solved" or "1". A*

terminal OR node is always Unsolved (no consistent value assignments) labeled "0". The labeling of internal nodes is defined recursively and is dependent on the specific task and the specific graphical-model. For the task of finding a solution in a constraint network, an OR internal node is labeled "1" iff one of its successor nodes is "1" and an AND internal node is labeled "1" iff all its successor nodes are labeled "1".

The above AND/OR search tree is well defined for any graphical model, not just for constraint networks. The various tasks can be distinguished by the function associated with a solution subtree which dictates the labels associated with terminal and intermediate nodes. For counting solutions, the value of an OR node is the sum values of its child nodes and the value of an AND node is their product. For computing the probability of evidence or computing belief in a Bayesian network the labeling of nodes is based on their CPTs, instantiated appropriately and propagated by sum and products over the OR and AND nodes. (More details ahead)

An AND/OR search tree becomes an OR search tree when its DFS tree is a chain. The virtue of an AND/OR search tree representation is that its size may be far smaller than the traditional (*OR*) tree representation.

Example 1 *Consider a graphical model in Figure 1a, over domains* $\{1, 2, 3\}$. *It can represent a graph-coloring problem or a belief network whose flat constraints are identical to the graph-coloring costraints. Its OR search tree along a DFS ordering is given in 1(b) and its AND/OR search tree based on DFS tree* T *rooted at* X*, is given in Figure 1c. A solution subtree is highlighted in 1c. We see that the size of the traditional OR search tree is* $O(2^7)$*, while the size of the AND/OR search tree is* $O(4 \cdot 2^3)$*. We ignore OR nodes when counting nodes because they provide a constant factor (k) at the most and we believe a non-naive implementation does not need to express OR nodes explicitly. Notice that if we add one constraint between* X *and* R *to this problem, we can still use the same DFS tree* T *yielding a similar structure AND/OR search tree, except that some values of variable* R *are no longer consistent with all their predecessors on the partial path. For example, in that case,* $< R, 1 >$ *in 1c will not be present under the subtree of* $< X, 1 >$.

3.1 From DFS Trees to Legal Trees

The construction of AND/OR search graphs can use as its basis not just a DFS spanning tree but a larger collection of spanning trees that we call *legal trees*. This generalization accommodates many more trees and can therefore yield better AND/OR search trees.

DEFINITION **3 (A Legal tree of a graph)** *Given an undirected graph* $G = (V, E)$*, a directed rooted tree* $T = (V, E')$ *defined on all its nodes is* legal *if any arc of* G *which is not included in* E' *is a back-arc, namely it connects a node to an ancestor in* T*. The arcs in* E' *may not all be included in* E*. Given a legal tree* T *of* G*, the extended graph of* G *relative to* T *is defined as* $G^T = (V, E \cup E')$.

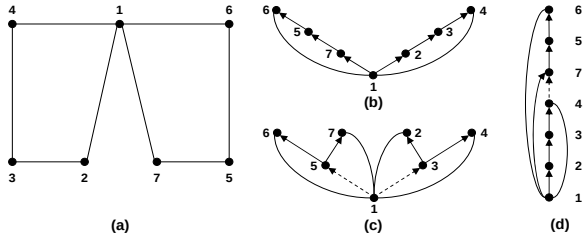


Figure 2: A graph (a), a DFS tree T_1 (b), a legal tree T_2 (c) a legal chain T_3

Clearly, any DFS tree and any chain are legal trees.

Example 2 *Consider the graph* G *displayed in Figure 2a. Ordering* $d_1 = 1, 2, 3, 4, 7, 5, 6$ *is a DFS ordering of a DFS tree* T_1 *having the smallest DFS-tree depth of 3 (Figure 2(b)). The tree,* T_2 *in Figure 2c is legal and has a tree depth of 2 only. The two tree-arcs (1,3) and (1,5) are not in* G*. In 2d, tree* T_3 *is a legal chain. The extended graphs* G^{T_1}*,* G^{T_2} *and* G^{T_3} *are presented in 2(b,c,d) when we ignore directionality.*

It is easy to see that,

THEOREM **3** *Given a graphical model* $\mathcal{R}$ *and a legal tree* T*, its AND/OR search tree* $S_T(\mathcal{R})$ *is sound and complete (i.e., it contains all and only solutions) and its size is* $O(n \cdot exp(m))$ *where* m *is the legal tree's depth.* □

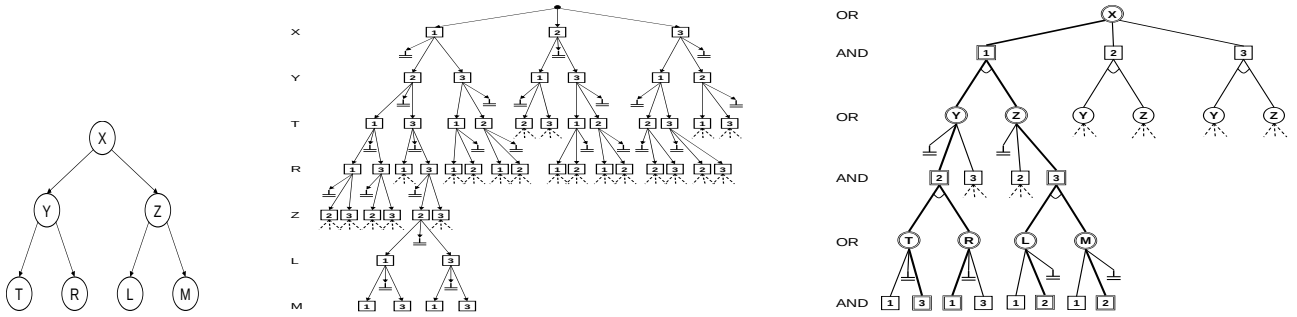
Finding a legal or a DFS tree of minimal depth is known to be NP-complete. However the problem was studied, and various greedy heuristics are available. For example, legal trees can be obtained by generating a heuristically good induced-graph along ordering d and then traversing it depth-first search breaking ties in favor of earlier variables [Bayardo & Miranker1996]. The following relationship between induced-width and legal trees is well known [Bayardo & Miranker1996, Dechter2003].

Proposition 1 *Given a tree-decomposition of a primal graph* G *having* n *nodes, whose tree-width is* w^**, there exists a legal tree* T *of* G *whose depth,* m*, satisfies:* $m \leq w^* \cdot logn$. □

THEOREM **4** *A graphical model that has a tree-width* w^* *has an AND/OR search tree whose size is* $O(exp(w^* \cdot logn))$. □

Table 1 shows the difference in hight between dfs spanning trees and legal spanning trees generated using common heuristics over randomly generated directed graphs that are moralized.

Table 2 demonstrate the size saving of AND/OR vs OR search spaces for 3 random networks having 20 bi-valued variables, 18 CPTs with 2 parents per child and 2 root nodes. The size of the OR space is the full binary tree of



(a) A constraint tree (b) OR search tree (c) AND/OR search tree with one of its solution subtrees

Figure 1: OR vs. AND/OR search trees; note the connector for AND arcs

Model (DAG)	width	Legal tree depth	DFS tree depth
(N=50,P=2,C=48)	9.5	16.82	36.03
(N=50,P=3,C=47)	16.1	23.34	40.60
(N=50,P=4,C=46)	20.9	28.31	43.19
(N=100,P=2,C=98)	18.3	27.59	72.36
(N=100,P=3,C=97)	31.0	41.12	80.47
(N=100,P=4,C=96)	40.3	50.53	86.54

Table 1: Average depth of legal trees vs. DFS trees. 100 instances of each random model.

		OR space		AO space		
width	height	time(sec.)	nodes	time(sec.)	AND nodes	OR nodes
4	10	3.125	2,097,150	0.02	7,806	3,903
6	9	3.124	2,097,150	0.02	6,318	3,159
5	10	3.114	2,097,150	0.02	7,326	3,663

Table 2: OR versus AND/OR search size, 20 nodes

depth 20. The size of the AND/OR space varies based on the legal tree.

3.2 Searching the AND/OR Search Tree

Any algorithm that traverses the AND/OR search tree in a depth-first manner is guaranteed to have time bound exponential in the depth of the legal tree of the graphical model and may operate using linear space. Given a belief network, we can use its moral graph as the primal graph and any of its legal trees to yield a well defined AND/OR search tree.

Figure 3 introduces a depth-first search algorithm traversing the AND/OR search tree computing the probability of evidence we refer to as AND-OR-TREE-BELIEF. The posterior beliefs of the root node can be obtained by normalizing. The reader should ignore all the lines with brackets. Those lines will be active for a graph searching algorithm we introduce later. (we present two algorithms at once for space considerations.) Given an AND/OR search tree based on a legal tree T, the algorithm labels the arcs emanating from OR nodes, using the bucket data structure as follows. Let $d_{dfs}(T)$ be a DFS ordering of legal tree T and let each variable X be associated with a bucket $B(X)$. The algorithm first partitions the CPTs relative to d_{dfs} placing each CPT-function into the bucket of the latest variable mentioned in its scope.

When expanding an OR node X, the algorithm associates each of its child nodes $< X, v >$ with the product of CPTS in $B(X)$ instantiated by the values of the path to $< X, v >$. In other words, given a path $\bar{v}_{X_i} = (< X_0, v_0 >, ... < X_i, v_i >)$, the node $< X_i, v_i >$ is labeled initially by $l(\bar{v}_i) = \Pi_{f \in B(X_i)} f(\bar{v}_i)$.

The g values are propagated from leaf to root as follows. The g values of a *leaf* AND node $< X, v >$ is equal to its l values. The g values of terminal OR nodes are 0. The g values of internal OR nodes are obtained by the sum of the g values of their successors. The g value of an internal AND node $< X, v >$ is the product of its own l-value and the g-values of its successors. Since the AND-OR-TREE-BELIEF explores each node in S_T exactly once, we get:

THEOREM **5** *The complexity of* AND-OR-TREE-BELIEF *is linear space and time $O(nk^m)$, when m is the depth of a legal tree spanning the belief network's moral graph. When the moral graph has a tree-decomposition having tree-width w, the algorithm can be bounded by $O(n \cdot exp(log n \cdot w))$.*

Relating to backjumping. Algorithm backjumping [Gaschnig1979], graph-based or conflict-based for constraint satisfaction was designed to overcome the redundancy imposed by the OR structure of the search tree. It can be shown that graph-based backjumping mimics the exploration of an AND/OR search tree. Indeed, it was shown that the depth of a DFS-tree or a legal-tree [Freuder & Quinn1987, Bayardo & Miranker1996] plays an important role in bounding backjumping complexity. It can be shown that the linear-space version of recursive-conditioning explores the AND/OR search tree as well [Darwiche1999].

4 Minimal AND/OR Search Graphs

It is often the case that certain states in the search tree can be merged because the subtree they root are identical. Any two such nodes are called *unifiable*, and when merged, transform the search tree into a search graph. For example, in Figure 1c, the search tree below any appearance of $< Y, 1 >$ are all unifiable. In the remainder of the

procedure AND-OR-BELIEF
Input: A belief network $\mathcal{BN} = (X, D, G, P)$, evidence e. constraints are the flat CPTs in P.
$\bar{v}$ is partial instantiation to current AND node. A legal tree T rooted at X_0
Output: The probability of the evidence $g(X_0)$.

1. OPEN $\leftarrow X_0$, type(X_0) = OR, assign evidence.
   Create a search graph G', $\leftarrow X_0$, $\bar{v} \leftarrow \phi$
   Create a list called $CLOSED$ (CL), initially empty.
2. $n \leftarrow$ first node in OPEN, move to CL
3. Expand n generating all its successors as follows:
   if $type(n) = OR$, $n = X$
      succ(X) $\leftarrow \{< X, v > \mid consistent(< X, v >)\}$
      if $succ(X) = \Phi$, $g(X) = 0$;(deadend)
        [[$no - good(X) \leftarrow \bar{v}_{pa_X}$,update flat CPTs]]
      else , add all $succ(X)$ to G', set pointers to X.
    for each $< X, v > \in succ(X)$ do,
      $\bar{v} \leftarrow (\bar{v}, < X, v >)$
      [[if $\bar{v}_{psa_X}$ is a new context not in $OP \cup CL$ **]]**
        [[then $c = c(< X, v >) = \bar{v}_{psa_X}$]]
        $type(< X, v >) \leftarrow AND$
        $l(< X, v >) = \Pi_{f \in Bucket_X} f(\bar{v})$.
        add $< X, v >$ to OP.
      [[**else** , if exists, $n =< X, v > \in OP \cup CL$]]
      [[s.t $c(n) = c$, then]]
        [[merge $< X, v >$ with n in G';]]
   if $type(n) = AND$, $n =< X, v >$
     $succ(< X, v >) \leftarrow \{Y | Y \in ch(X)\ in\ T\}$
     if X terminal in T (no successors)
     $g < X, v > \leftarrow l(< X, v >)$.
     put $succ(< X, v >)$ on top of OPEN and in G'
4. **Propagate: while** you can, propagate g values:
   a. For a non-terminal AND node $< X, v >$
     [[**if** $Y \in succ(< X, v >)$ and $g(Y) = 0$,]]
     [[remove siblings of Y from OPEN.]]
       [[set $g(< X, v >) = 0$.]]
     else ,
       if all $succ(< X, v >)$ evaluated,
       $g(< X, v >) = l(< X, v >) \Pi_{Y \in ch(X)} g(Y)$
   b. For a non-terminal OR node X:
     if all succ(X) have g values
       $g(X) = \sum_{< X, v > \in succ(X)} g(< X, v >)$
   end while
   c. **if** X_0 is evaluated, exit with $g(X_0)$
   d. Remove portion of G' that is not relevant
5. Go to step 2.

end procedure

Figure 3: The belief-updating for graphs algorithm.

paper we will characterize the smallest search graph that may result from merging nodes and will analyze the impact of graph search vs tree-search in the context of AND/OR spaces. Given an AND/OR search graph S_T of a network $\mathcal{R}$ relative to a legal tree T. A partial path in the AND/OR search-tree $(< X_1, a_1 >, < X_2, a_2 >, ... < X_i, a_i >)$ is abbreviated to $(\bar{X}, \bar{a}_i)$, where $\bar{X}$ is the sequence of variables and $\bar{a}$ is their corresponding sequence of value assignments, or $\bar{a}_i$ if $\bar{X}$ is clear.

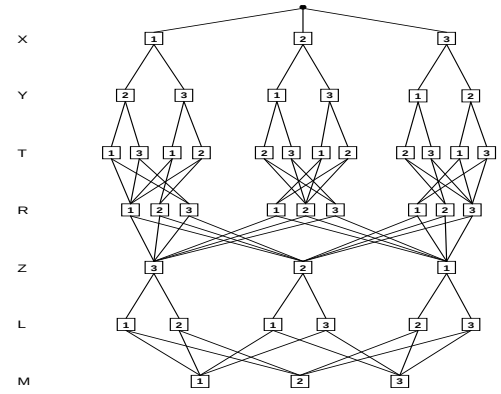


Figure 4: Condensed OR graph for the tree problem

DEFINITION **4 (legal transformation)** *Given two partial paths over the same set of variables, $s_1 = (\bar{X}_i, \bar{a}_i)$, $s_2 = (\bar{X}_i, \bar{b}_i)$ where $a_i = b_i$, we say that s_1 and s_2 are* unifiable *at $< X_i, v >$ (can be merged) iff the search subgraphs rooted at s_1 and s_2 are identical. The* Merge *operator over search graphs, $Merge_{(s_1, s_2)}$ transforms S_T into a graph S'_T by merging s_1 with s_2.*

It is clear that the merge operator preserves soundness and completeness of search graphs (OR or AND/OR), and it shrinks the search graph with each application. It can be shown that given an AND/OR search tree, its closure under the merge operator yields a fixed point, called the minimal AND/OR search graph.

DEFINITION **5 (minimal AND/OR search graph)** *The minimal AND/OR search graph relative to T is the closure under* merge *of the AND/OR search tree S_T.*

The above definition is applicable, when T is a legal-chain, to the traditional OR search tree as well. But, in many cases we will not be able to reach the same compression we see in the AND/OR search graph, because of the linear structure imposed by the OR search tree.

Example 6 *The smallest OR search graph of the problem in Figure 1a is given in Figure 4 along the DFS order X, Y, T, R, Z, L, M. The smallest AND/OR graph of the same problem along some tree is given in Figure 5. We see that some variable-value pairs must be repeated in Figure 4 while in an AND/OR case they appear just once. For example, the subgraph below the paths $< X, 1 >, < Y, 2 >$ and $< X, 3 >, < Y, 2 >$ cannot be merged.*

5 Rules for merging nodes

Given a graphical model $\mathcal{R} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ and a legal tree T, there could be many AND/OR graphs relative to T that are equivalent to the AND/OR search tree S_T, each obtained

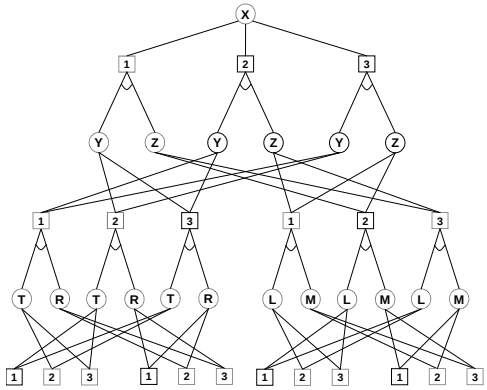


Figure 5: The AND/OR search graph of the tree graphical-model in Figure 1a and one of its solution subtrees

by some sequence of merging. In this section we will discuss effective rules for unifying subtrees, targeting the minimal AND/OR search graph as much as computational resources allow. The rules provide an efficient way for generating the graph without creating the whole search tree S_T. To get the basic idea we focus first on AND/OR search graphs for graphical-models having no cycles, called *tree-models*.

5.1 The case of tree-models

Consider again the graph in Figure 1a and its AND/OR search tree in Figure 1c. Observe that at level 3, the node $< Y, 1 >$ appears twice, (and so is $< Y, 2 >$ and $< Y, 3 >$). Clearly however, the subtrees rooted at each of these 2 AND nodes are identical because any assignment to Y uniquely determined its subtree. Indeed, the AND/OR search graph in Figure 5 is equivalent to the AND/OR search tree in Figure 1c. Its size however, is far more condensed (note that Figure 1c displays only a small portion of the AND/OR tree.)

DEFINITION **6 (Explicit AND/OR graphs for tree-models)** *Given a tree-model and a rooted DFS tree T (note that we restrict ourselves to the special case of DFS trees here), the* explicit AND/OR search graph *of the tree-model relative to T is obtained from S_T, by merging all identical AND nodes $< X, i >$ that reside in the same level of S_T.*

Proposition 2 *Given a rooted tree-model T, 1. Its explicit AND/OR search* graph, *is equivalent to S_T and is therefore sound and complete. 2. The size of the explicit AND/OR search graph of a tree-model is $O(nk)$. 3. For some tree models the explicit AND/OR search graph is minimal. Namely, the minimal AND/OR search graph for tree-models is $\theta(nk)$.*

Generalized width parameter. Next we introduce the *induced-width of a legal tree of G* which is instrumental for characterizing Or graphs vs AND/OR graphs. We denote by $d_{dfs}(T)$ a DFS ordering of a tree T.

DEFINITION **7 (induced-width of a legal tree)** *Given G^T, an extended graph of G relative to a legal T, the induced width of G relative to T, $w_T(G)$ is the induced-width of G^T along $d_{dfs}(T)$.*

We can show that,

Proposition 3 *1. The minimal induced-width of G relative to all legal trees is identical to the induced-width (tree-width) of G. 2. The induced-width of a legal chain is identical to its path-width pw along d.*

Example 7 *In Figure 2b, the induced graph of G relative to T_1 contains also the arcs (1,3) and (1,5) and its induced-width is 2. G^{T_2} is already triangulated and its induced-width is 2 as well. The induced-graph of G^{T_3} has the additional (1,5) and (1,3) edges having induced-width 2 as well.*

5.2 A general merging rule.

This section provides a general generative rule for unifying nodes in the AND/OR search graph and will use the generalized width parameter to characterize its effect. Given the induced graph of G^T, denoted G^{*T}, each variable and its parent set is a clique. We associate each variable with its parent-separator as follows:

DEFINITION **8 (parent-separators)** *Given the induced-graph, G^{*T}, the parent-separators of X denoted psa_X, are its earlier neighbors in the induced-graph that are also neighbors of some future variable in T .*

In G^{*T}, for every node X_i, the parent-separators of X_i separates in T its ancestors on the path from the root, and all its descendents in G^T. Therefore,

THEOREM **8** *Given G^{*T} Let $s_1 = (\bar{a}_i, < X_{i+1}, v >)$ and $s_2 = (\bar{b}_i, < X_{i+1}, v >)$ be two partial paths of assignments in its AND/OR search tree S_T, ending with the same assignment variable $< X_{i+1}, v >$. If $s_1[psa_{i+1}] = s_2[psa_{i+1}]$, then the AND/OR search subtrees rooted at s_1 and s_2 are identical and s_1 and s_2 can be merged at $< X_{i+1}, v >$.*

DEFINITION **9 (context)** *For every state s_i, $s_i[psa_i]$ is called the context of s_i when psa_i is parent-separators set of X_i relative to the legal tree T.*

THEOREM **9** *Given G, a legal tree T and its induced width $w = w_T(G)$, the size of the AND/OR search graph based on T obtained when every two nodes in S_T having the same context are merged is $O(n \cdot k^w)$, when k bounds the domain size.*

Thus, the minimal AND/OR search graph of G relative to T is $O(n \cdot k^w)$ where $w = w_T(G)$. Since $\min_T\{w_T(G)\}$ equals w^* and since $\min_{T \in chain}\{w_T(G)\}$ equals pw^* we get

Corollary 1 *The minimal AND/OR search graph is bounded exponentially by the graph-model tree-width*

while the OR minimal search graph is bounded exponentially by its path-width.

Example 10 *For the balanced tree in Figure 1 we see the AND/OR tree in Figure 5 vs the OR tree on Figure 4. The induced-width is 2 along the legal-chain* (X, Y, T, R, Z, L, M). *Since the context of T along* d *is* TXY *in the first 3 levels of the OR search graph in Figure 4 there are no merged nodes. Along the DFS tree the context of every node is itself and indeed all AND nodes having the same variable-value pairs are merged.*

It is well known [Bodlaender & Gilbert1991] that for any graph $w^* \leq pw^* \leq w^* logn$. It is also easy to place m^* (the minimal depth legal tree) yielding $w^* \leq pw^* \leq m^* \leq w^* \cdot logn$. We next demonstrate that the bound $pw^* \leq w^* logn$ is tight.

Consider a balanced rooted tree T of depth r having branching degree p, $n = p^r - 1$. Clearly, the induced-width of T is 1 and indeed its AND/OR search graph size, denoted s_{aor} is $O(n \cdot k) = O(p^r \cdot k)$. In contrast,

Proposition 4 *Consider a legal-chain of* T *along* $d = d_{dfs}(T)$. *The induced-width of* T *relative to the legal-chain along* d *is* r. *Namely,* $w_d(T) = r$.

We can show that for a balanced tree of depth r, the minimal induced-width along any legal chain is not less than r. Thus, for a balanced tree of depth r and branching degree p, there exists T whose minimal AND/OR search graph is $O(p^r k)$, while the smallest minimal OR search graph for any ordering is $O(n \cdot k^r)$. Therefore, if s^*_{or} is the minimal search graph over all chains while s^*_{aor} is the minimal one over all AND/OR search graphs, we get that $\frac{s^*_{or}}{s^*_{aor}} = (\frac{k}{p})^{log_p n}$. When we move from trees to tree-decompositions of cluster size w (e.g., w-trees[1]) we can extend the above conclusion to $\frac{s^*_{or}}{s^*_{ao}} = (\frac{k}{\sqrt[w]{p}})^{w \cdot log_p n}$ (details are omitted for space reasons).

5.3 Searching AND/OR search graphs

Any algorithm traversing the AND/OR search graph needs to record nodes during search so merging would be possible. To do that the algorithm associates a context with each AND node and whenever a new AND node is generated, its context is compared against the list of contexts for the same variable in the same level. If it was generated already only pointers will be established appropriately.

Algorithm AND-OR-GRAPH-BELIEF is presented in Figure 3. For this version the reader should include all the bracketed lines. In step 3, the algorithm expands the next node in OPEN (OP). If this is an OR node that has no consistent successors, it is identified as dead-end, assigned $g = 0$. A no-good is recorded and the set of flat constraints are modified to include this new constraint. This step will cause pruning of the search tree (the following section ellaborates). Otherwise, for each consistent, value v of X the algorithm computes the context of $< X, v >$ denoted $c(< X, v >)$ and check it against recorded contexts.

THEOREM **11** *The complexity of algorithm* AND-OR-GRAPH-BELIEF *is time and space exponential in the induced width of the legal tree.*

6 The backtrack-free AND/OR space

Most advanced constraint processing algorithms incorporate no-good caching during search, use variable-elimination algorithms such as *adaptive-consistency* [Dechter2003] to generate all no-goods prior to search, or apply constraint propagation at each node. Such schemes can be viewed as traversing, a *pruned* AND/OR search *tree* or graph that we call *backtrack-free*. The backtrack-free AND/OR search tree of $\mathcal{R}$ based on T, denoted $BF_T(\mathcal{R})$, is obtained by pruning all subtrees labeled "0" from $S_T(\mathcal{R})$. It can be shown that the constraint algorithm *adaptive-consistency* compiles a constraint network $\mathcal{R}$ into a graphical model $\mathcal{R}'$ whose AND/OR search tree is identical to $BF_T(\mathcal{R})$ (details are omitted).

The notion of minimality vs that of pruning of inconsistent subtrees (aiming at the backtrack-free search space) are related but not the same. While recording the context of a dead-end is a no-good, no-goods can effect and prune the search tree itself being used as new constraints. Therefore, we can have a minimal search *graph* that is NOT backtrack-free as well as a search *tree* that is backtrack-free.

When the search space is backtrack-free (no dead-end nodes) and if we seek a single solution, the size of the minimal AND/OR search graph and its being OR vs AND/OR are irrelevant. Minimality and AND/OR will affect a traversal algorithm that counts all solutions or compute beliefs, however, even when the search space is backtrack-free.

7 Conclusions, Discussion, Related Work

The primary contribution of this paper is in viewing search for graphical model in the context of AND/OR search spaces rather than OR spaces. We introduced the AND/OR search tree, and showed that its size can be bounded exponentially by the depth of its legal tree. This implies exponential saving for any linear space algorithms traversing the AND/OR graph. We observe that many known advanced algorithms for constraint processing and satisfiability can be explained as traversing the AND/OR search tree (e.g. backjumping [Freuder & Quinn1987, Bayardo & Miranker1996]). Also, recent algorithms in probabilistic reasoning such as recursive-conditioning [Darwiche1999, F. Bacchus & Piassi2003] can operate in linear space and can be viewed as searching the AND/OR search tree.

The AND/OR search tree was extended into a graph by

[1] a k-tree of size k+1 is a clique. Given a k-tree of size n, a k-tree of size k+1 can be obtained by connecting a new node to each node in a clique of size k

merging identical subtrees. We showed that the size of the minimal AND/OR search graph is exponential in the tree-width while the size of the minimal OR graph is exponential in the path-width. Since for some graphs the difference between the path-width and tree-width is substantial (e.g., balanced legal trees) the AND/OR representation implies exponential time and space savings for algorithms traversing the AND/OR graph. Searching the AND/OR search *graph* can be implemented by goods caching during search, while no-good recording is interpreted as pruning and collapsing portions of the search space independent of it being a tree or a graph, an OR or an AND/OR. For finding a single solution, pruning the search space is the most significant action. For counting and probabilistic inference, using AND/OR graphs can be of much help even on top of no-good recording.

Many memory-intensive algorithms can be viewed as searching the AND/OR search *graph*. For example, recent work [Terrioux & Jegou2003] perform search guided by a tree-decomposition either for constraint satisfaction andoptimization is searching the AND/OR search *graph*, whose legal tree is constructed along the tree-decomposition. Algorithm recursive-conditioning in its full caching mode, for belief updating [Darwiche1999], is a depth-first search of an AND/OR search graph. Another variant is value-elimination, introduced recently [F. Bacchus & Piassi2003] both for belief updating and counting models of a CNF formula. Most advanced SAT solvers employ both backjumping , clause learning (e.g., no-good learning) thus search the AND/OR search tree.

Relationship with bucket-elimination. The worst-case complexity of AND-OR-GRAPH-BELIEF is identical to that of variable-elimination and join-tree clustering. This is not surprising. We can show that bucket elimination, if applied along any reversed order of the legal tree, will generate every context node exactly once, and with each it associates the final g value. The bucket-elimination algorithms can be viewed as searching the merged AND/OR graph that is backtrack-free.

Realtionships with OBDDs. The notion of minimal OR search graphs is similar to the known concept of *Ordered Binary Decision Diagrams (OBDD)* in the literature of hardware and software design and verification. The properties of OBDDs were studied extensively in the past two decades [McMillan1993]. It is well known that the size of OBDDs is bounded exponentially by the *path-width* of the CNF's interaction graph. Our notion of minimal AND/OR search graphs, if applied to CNFs, resembles *tree OBDDS* developed subsequently [McMillan1994]. The OBDDs definitions incorporates also inconsistent subtree pruning and consistent subtree collapse. Some relationship between graphical model compilation and OBDDs were studied in [Darwiche & Marquis2002].

References

[Arnborg1985] Arnborg, S. A. 1985. Efficient algorithms for combinatorial problems on graphs with bounded decomposability - a survey. *BIT* 25:2–23.

[Bayardo & Miranker1996] Bayardo, R., and Miranker, D. 1996. A complexity analysis of space-bound learning algorithms for the constraint satisfaction problem. In *AAAI'96: Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 298–304.

[Bodlaender & Gilbert1991] Bodlaender, H., and Gilbert, J. R. 1991. Approximating treewidth, pathwidth and minimum elimination tree-height. In *Technical report RUU-CS-91-1, Utrecht University*.

[Darwiche & Marquis2002] Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research (JAIR)* 229–264.

[Darwiche1999] Darwiche, A. 1999. Recursive conditioning. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI99)*.

[Dechter1990] Dechter, R. 1990. Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition. *Artificial Intelligence* 41:273–312.

[Dechter2003] Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann Publishers.

[F. Bacchus & Piassi2003] F. Bacchus, S. D., and Piassi, T. 2003. Value elimination: Bayesian inference via backtracking search. In *Uncertainty in AI (UAI03)*.

[Freuder & Quinn1987] Freuder, E. C., and Quinn, M. J. 1987. The use of lineal spanning trees to represent constraint satisfaction problems. Technical Report 87-41, University of New Hampshire, Durham.

[Gaschnig1979] Gaschnig, J. 1979. Performance measurement and analysis of search algorithms. Technical Report CMU-CS-79-124, Carnegie Mellon University.

[McMillan1993] McMillan, K. L. 1993. *Symbolic Model Checking*. Kluwer Academic.

[McMillan1994] McMillan, K. L. 1994. Hierarchical representation of discrete functions with application to model checking. In *Computer Aided Verification, 6th International conference, David L. Dill ed.*, 41–54.

[Nillson1980] Nillson, N. J. 1980. *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA.

[Terrioux & Jegou2003] Terrioux, C., and Jegou, P. 2003. Hybrid backtracking bounded by tree-decomposition of constraint networks. In *Artificial Intelligence*.