Advancing AND/OR Search for Optimization Using Diverse Principles

Radu Marinescu[1] and **Rina Dechter**[2]

Abstract. In recent years, several Branch-and-Bound and best-first search algorithms were developed to explore the AND/OR search graph for solving general constraint optimization problems. Previous work showed the tremendous gain obtained by exploiting problem's decomposition (using AND nodes), equivalence (by caching) and irrelevance (via the power of lower bound heuristics). In this paper, we show the additional improvements that can be gained by bringing together all the above, as well as diverse refinements and optimizing principles such as exploiting determinism via constraint propagation, using good initial upper bounds generated via stochastic local search and improving the quality of the guiding pseudo tree. We illustrate our results using a number of benchmark networks, including the very challenging ones that arise in genetic linkage analysis.

1 INTRODUCTION

Constraint satisfaction problems (CSPs) provide a formalism for formulating many interesting real world problems as an assignment of values to variables, subject to a set of constraints. A constraint optimization problem (COP) is defined as a regular CSP augmented with a set of cost functions (called soft constraints) indicating preferences. The aim of constraint optimization is to find a solution to the problem whose cost, expressed as the sum of the cost functions, is minimized or maximized.

The AND/OR Branch-and-Bound search (AOBB) introduced in [10] is a Branch-and-Bound algorithm that explores an AND/OR search tree for graphical models [5], in a depth-first manner. The AND/OR Branch-and-Bound search with caching (AOBB-C) [11] allows saving previously computed results and retrieving them when the same subproblem is encountered again. The algorithm explores the context minimal AND/OR graph. A best-first AND/OR search algorithm (AOBF-C) that traverses the AND/OR graph was also explored [12]. Earlier empirical evaluations demonstrated (1) the impact of AND decomposition, (2) the impact of caching, (3) the impact of some dynamic variable ordering heuristics, (4) the impact of the lower bound strength, as well as (5) the impact of best-first versus depth-first search regimes [10, 11, 12].

In this paper, we want to take these classes of algorithms as much further as we can by including additional known principles of problem solving and examine their interactive impact on performance. We investigate three key factors that impact the performance of any search algorithm: (1) the availability of hard constraints ($i.e.$, determinism) in the problem (2) the availability of a good initial upper bound provided to the algorithm, and (3) the availability of good quality guiding pseudo trees. We therefore extend AOBB-C (and whenever relevant, AOBF-C) to exploit explicitly the computational power of hard constraints by incorporating standard constraint propagation techniques such as unit resolution. We provide AOBB-C with a non-trivial initial upper bound computed by local search. Finally, we investigate randomized orderings generated via two heuristics for constructing small induced width/depth pseudo trees.

We evaluate the impact and interaction of these extensions on the optimization problem of finding the most probable explanation in belief networks using a variety benchmarks. We show that exploiting the determinism as well as good quality initial upper bounds and pseudo trees improves the performance dramatically in many cases.

[1] 4C, University College Cork, Ireland email:r.marinescu@4c.ucc.ie
[2] University of California, Irvine, USA email: dechter@ics.uci.edu

2 BACKGROUND

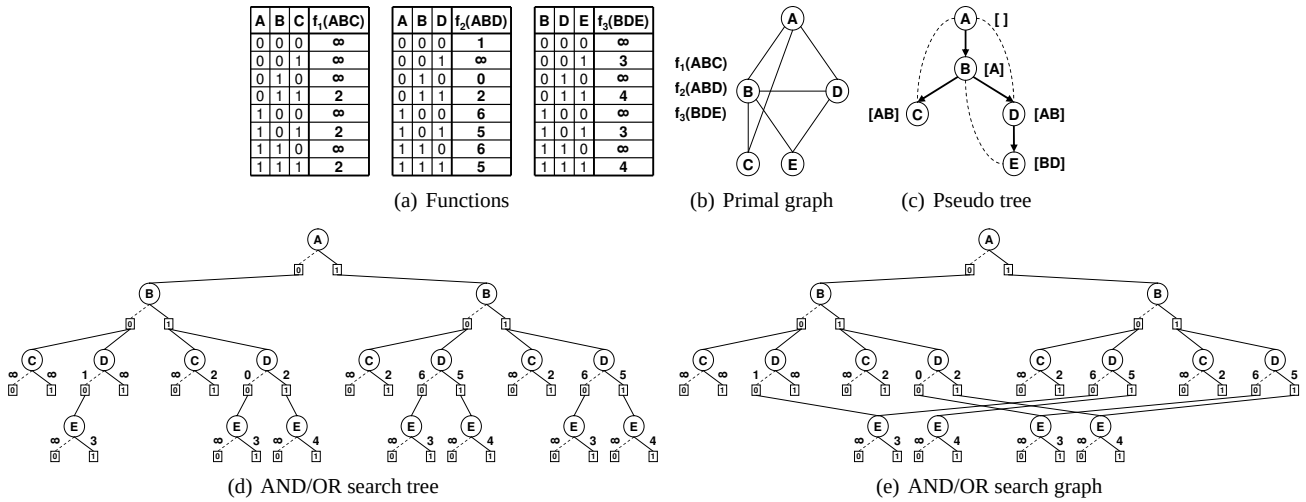
2.1 Constraint Optimization Problems

A finite *Constraint Optimization Problem* (COP) is a triple $\mathcal{P} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$, where $\mathbf{X} = \{X_1, ..., X_n\}$ is a set of variables, $\mathbf{D} = \{D_1, ..., D_n\}$ is a set of finite domains and $\mathbf{F} = \{f_1, ..., f_r\}$ is a set of cost functions. Cost functions can be either *soft* or *hard* (constraints). Without loss of generality we assume that hard constraints are represented as (bi-valued) cost functions. Allowed and forbidden tuples have cost 0 and ∞, respectively. The scope of function f_i, denoted $scope(f_i) \subseteq \mathbf{X}$, is the set of arguments of f_i. The goal is to find a complete value assignment to the variables that minimizes the global cost function $f(\mathbf{X}) = \sum_{i=1}^{r} f_i$, namely to find $x = \arg\min_{\mathbf{X}} \sum_{i=1}^{r} f_i$. Given a COP instance, its *primal graph* G associates each variable with a node and connects any two nodes whose variables appear in the scope of the same function.

Belief networks [14] provide a formalism for reasoning under conditions of uncertainty. A belief network represents a joint probability distribution over the variables of interest. A function of the model encodes the conditional probability distribution of a variable given its parents in the graph. The most common optimization task over belief networks is finding the *Most Probable Explanation* (MPE), namely finding a complete assignment with maximum probability that is consistent with the evidence in the network. It appears in applications such as speech recognition or medical diagnosis.

2.2 AND/OR Search Spaces for Constraint Optimization

The AND/OR search space [5] is a unifying framework for advanced algorithmic schemes for graphical models, including belief networks, constraint networks and cost networks. Its main virtue consists in exploiting conditional independencies between variables during search,



(a) Functions (b) Primal graph (c) Pseudo tree

(d) AND/OR search tree (e) AND/OR search graph

Figure 1. AND/OR search spaces for constraint optimization.

which can provide exponential speedups over traditional structure-blind search methods. The search space is defined using a backbone *pseudo tree* [6].

DEFINITION **1 (pseudo tree)** *Given an undirected graph $G = (\mathbf{X}, E)$, a directed rooted tree $\mathcal{T} = (\mathbf{X}, E')$ defined on all its nodes is called* pseudo-tree *if any edge of G that is not included in E' is a back-arc in $\mathcal{T}$, namely it connects a node to an ancestor in $\mathcal{T}$.*

AND/OR Search Trees. Given a COP instance $\mathcal{P} = \langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ its primal graph G and a pseudo tree $\mathcal{T}$ of G, the associated AND/OR search tree, $S_{\mathcal{T}}$, has alternating levels of OR and AND nodes. The OR nodes are labeled X_i and correspond to the variables. The AND nodes are labeled $\langle X_i, x_i \rangle$ (or just x_i) and correspond to value assignments of the variables. The structure of the AND/OR search tree is based on the underlying pseudo tree $\mathcal{T}$. The root of the AND/OR search tree is an OR node labeled with the root of $\mathcal{T}$. The children of an OR node X_i are AND nodes labeled with assignments $\langle X_i, x_i \rangle$ that are consistent with the assignments along the path from the root. The children of an AND node $\langle X_i, x_i \rangle$ are OR nodes labeled with the children of variable X_i in $\mathcal{T}$. It was shown in [6, 5] that given a COP instance $\mathcal{P}$ and a pseudo tree $\mathcal{T}$ of depth m, the size of the AND/OR search tree based on $\mathcal{T}$ is $O(n \cdot k^m)$, where k bounds the domains of variables.

AND/OR Search Graphs. The AND/OR search tree may contain nodes that root identical conditioned subproblems. Such nodes can be merged yielding an AND/OR graph. Its size becomes smaller at the expense of using additional memory by the search algorithm. Some mergeable nodes can be identified based on their *contexts*.

Given a pseudo tree $\mathcal{T}$ of an AND/OR search space, the *context* of an OR node X, denoted by $context(X) = [X_1 \dots X_p]$, is the set of ancestors of X in $\mathcal{T}$ ordered descendingly, that are connected in the primal graph to X or to descendants of X. The context of X separates the subproblem below X from the rest of the network. The *context minimal* AND/OR graph [5] is obtained from the AND/OR search tree by merging all the context mergeable nodes.

It can be shown [5] that given a COP $\mathcal{P}$, its primal graph G and a pseudo tree $\mathcal{T}$, the size of the context minimal AND/OR search graph is $O(n \cdot k^{w^*_{\mathcal{T}}(G)})$, where $w^*_{\mathcal{T}}(G)$ is the induced width of G over the DFS traversal of $\mathcal{T}$, and k bounds the domain size.

Weighted AND/OR Search Graphs. The OR-to-AND arcs from nodes X_i to x_i in an AND/OR search tree or graph are annotated by *weights* derived from the cost functions in $\mathbf{F}$. The *weight* $w(X_i, x_i)$ of the arc from the OR node X_i to the AND node x_i is the sum of all the cost functions whose scope includes X_i and is fully assigned along the path from the root to x_i, evaluated at the values along the path. Given a weighted AND/OR search graph, each of its nodes can be associated with a *value*. The value $v(n)$ of a node n is the minimal cost solution to the subproblem rooted at n, subject to the current variable instantiation along the path from the root to n. It can be computed recursively using the values of n's successors (see also [5] for details).

Example 1 *Figure 1 shows an example of AND/OR search spaces for a COP with binary variables. The cost functions are given in Figure 1(a). The value ∞ indicates a hard constraint. The primal graph is given in Figure 1(b), and the pseudo tree in Figure 1(c). The square brackets indicate the context of the variables. The AND/OR search tree is given in Figure 1(d). The numbers on the OR-to-AND arcs are the weights corresponding to the function values. The context minimal AND/OR graph is given in Figure 1(e).*

3 AND/OR SEARCH ALGORITHMS FOR CONSTRAINT OPTIMIZATION

In recent years several depth-first Branch-and-Bound and best-first search algorithms were developed to search the context minimal AND/OR graph for solving COPs [11, 12]. We next briefly overview these two classes of algorithms.

AND/OR Branch-and-Bound (AOBB-C) traverses the context minimal AND/OR graph in a depth-first manner via full caching. It interleaves forward expansion of the current partial solution tree with a backward cost revision step that updates node values, until search terminates. The efficiency of the algorithm also depends on the strength of its heuristic evaluation function ($i.e.$, lower bound). Specifically, each node n in the search graph has an associated heuristic function $h(n)$ underestimating $v(n)$ that can be computed efficiently when the node n is first expanded. The algorithm then computes the heuristic evaluation function $f(T')$ of the current partial solution T' and uses it to prune irrelevant portions of the

search space, as part of a Branch-and-Bound scheme.

Best-First AND/OR Search (AOBF-C) explores the context minimal AND/OR graph and interleaves forward expansion of the best partial solution tree with a cost revision step that updates the node values. First, AOBF-C finds the best partial solution tree by tracing down through the marked arcs of the explicit AND/OR search graph $C'_{\mathcal{T}}$ and expands one of its nonterminal leaf nodes. Starting with the node just expanded n, the algorithm then revises its value $v(n)$ and marks the outgoing arcs on the estimated best path to terminal nodes (OR nodes revise their values by minimization, while AND node by summation). The revised value $v(n)$, which is an updated lower bound on the cost of the subproblem rooted at n, is then propagated upwards in the graph. During the bottom-up step, AOBF-C labels an AND node as SOLVED if all of its OR child nodes are solved, and labels an OR node as SOLVED if its marked AND child is also solved. The optimal cost solution to the initial problem is obtained when the root node is labeled SOLVED.

Mini-Bucket Heuristics. The effectiveness of both depth-first and best-first AND/OR search algorithms greatly depends on the quality of the lower bound heuristic evaluation functions. The primary heuristic that we used in our experiments is the Mini-Bucket heuristic, which was presented in [11, 12]. It was shown that the intermediate functions generated by the Mini-Bucket algorithm MBE(i) [4] can be used to compute a heuristic function that underestimates the minimal cost solution to a subproblem in the AND/OR graph.

4 IMPROVING AND/OR BRANCH-AND-BOUND SEARCH

In this section we overview several principled improvements to the AND/OR Branch-and-Bound algorithm that we will incorporate.

4.1 Exploiting Determinism

When the functions of the COP instance express both hard constraints and general cost functions, it may be beneficial to exploit the computational power of the constraints explicitly via constraint propagation [1]. The approach we take for handling the determinism in COP is based on the known technique of *unit resolution* for Boolean Satisfiability (SAT) over a logical knowledge base (KB) in the form of propositional clauses (CNF) representing the hard constraints. One common way of encoding hard constraints as a CNF formula is the *direct encoding* [17].

The changes needed in the AND/OR Branch-and-Bound procedure are then as follows. Upon expanding an AND node $\langle X_i, x_j \rangle$ the corresponding SAT instantiation is asserted in KB, namely x_{ij} is set to *true*. If the unit resolution leads to a contradiction, then the current AND node is marked as dead-end and the search continues by expanding the next node on the search stack. Whenever the algorithm backtracks to the previous level, it also retracts any SAT instantiations recorded by unit resolution. Notice that the algorithm is capable of pruning the domains of future variables in the current subproblem due to conflicts detected during unit propagation.

4.2 Exploiting Good Initial Upper Bounds via Local Search

The AND/OR Branch-and-Bound algorithm assumed a trivial initial upper bound ($i.e.$, ∞), which effectively guarantees that the optimal solution will be computed, however it provides limited pruning in the initial phase. We therefore can incorporate a more informed upper bound, obtained by solving the problem via a local search scheme. This approach is often used by state-of-the-art constraint optimization solvers.

One of the most popular local search algorithms for COP is the *Guided Local Search* (GLS) method [16]. GLS is a penalty-based meta-heuristic, which works by augmenting the objective function of a local search algorithm ($e.g.$ hill climbing) with penalties, to help guide them out of local minima. GLS has been shown to be successful in solving a number of practical real life problems, such as the traveling salesman problem, radio link frequency assignment problem and vehicle routing.

4.3 Exploiting the Pseudo Tree Quality

The performance of the AND/OR search algorithms can be heavily influenced by the quality of the guiding pseudo tree. Finding the minimal depth or induced width pseudo tree is a hard problem [6, 2]. We describe next two heuristics for generating pseudo trees which we used in our experiments.

Min-Fill Heuristic. The *Min-Fill* ordering [9] is generated by placing the variable with the smallest *fill set* ($i.e.$, number of induced edges that need be added to fully connect the neighbors of a node) at the end of the ordering, connecting all of its neighbors and then removing the variable from the graph. The process continues until all variables have been eliminated. Once an elimination order is given, the pseudo tree can be extracted as a depth-first traversal of the min-fill induced graph, starting with the variable that initiated the ordering, always preferring as successor of a node the earliest adjacent node in the induced graph. An ordering uniquely determines a pseudo tree. This approach was first used by [2].

Hypergraph Decomposition Heuristic. An alternative heuristic for generating a low height balanced pseudo tree is based on the recursive decomposition of the dual hypergraph associated with the COP instance. The dual hypergraph of a COP $\langle \mathbf{X}, \mathbf{D}, \mathbf{F} \rangle$ is a pair $(\mathbf{V}, \mathbf{E})$ where each function in $\mathbf{F}$ is a vertex $v_i \in \mathbf{V}$ and each variable in $\mathbf{X}$ is a hyperedge $e_j \in \mathbf{E}$ connecting all the functions (vertices) in which it appears.

Generating heuristically good hypergraph separators can be done using a package called `hMeTiS` (*available at: http://www-users.cs.umn.edu/karypis/metis/hmetis*), which we used following [3]. The vertices of the hypergraph are partitioned into two balanced (roughly equal-sized) parts, denoted by $\mathcal{H}_{left}$ and $\mathcal{H}_{right}$ respectively, while minimizing the number of hyperedges across. A small number of crossing edges translates into a small number of variables shared between the two sets of functions. $\mathcal{H}_{left}$ and $\mathcal{H}_{right}$ are then each recursively partitioned in the same fashion, until they contain a single vertex. The result of this process is a tree of hypergraph separators which can be shown to also be a pseudo tree of the original model where each separator corresponds to a subset of variables chained together.

Randomization. Both the min-fill and hypergraph partitioning heuristics can randomize their tie breaking rules, yielding varying qualities of the generated pseudo tree.

5 EXPERIMENTS

In order to empirically evaluate the performance of the proposed improvements to AOBB-C algorithms, we have conducted a number of

minfill pseudo tree without randomization													
iscas89 (w*, h) (n, d)	SamIam CPLEX GLS	AOBB-C+SMB(i) AOBB-C+SAT+SMB(i) AOBB-C+GLS+SMB(i) AOBB-C+SAT+GLS+SMB(i) AOBF-C+SMB(i) i=6		AOBB-C+SMB(i) AOBB-C+SAT+SMB(i) AOBB-C+GLS+SMB(i) AOBB-C+SAT+GLS+SMB(i) AOBF-C+SMB(i) i=8		AOBB-C+SMB(i) AOBB-C+SAT+SMB(i) AOBB-C+GLS+SMB(i) AOBB-C+SAT+GLS+SMB(i) AOBF-C+SMB(i) i=10		AOBB-C+SMB(i) AOBB-C+SAT+SMB(i) AOBB-C+GLS+SMB(i) AOBB-C+SAT+GLS+SMB(i) AOBF-C+SMB(i) i=12		AOBB-C+SMB(i) AOBB-C+SAT+SMB(i) AOBB-C+GLS+SMB(i) AOBB-C+SAT+GLS+SMB(i) AOBF-C+SMB(i) i=14	
		time	nodes	time	nodes	time	nodes	time	nodes	time	nodes
		-	-	-	-	182.53	2,316,024	0.16	432	0.24	432
c432	out	374.29	4,336,403	189.13	2,043,475	1.02	9,512	0.16	432	0.25	432
(27, 45)	20.54	**0.05**	0	0.06	0	0.09	0	0.13	0	0.19	0
(432, 2)	0.08*	0.06	0	0.08	0	0.09	0	0.13	0	0.20	0
		out		out		106.27	488,462	0.20	432	0.28	432
		899.63	7,715,133	17.99	155,865	48.13	417,924	17.00	132,139	2.19	13,039
s953	out	0.19	829	0.16	667	0.20	685	0.31	623	0.74	623
(66, 101)	12.14	**0.12**	0	0.13	0	0.17	0	0.28	0	0.69	0
(440, 2)	0.05*	0.13	0	0.13	0	0.17	0	0.30	0	0.70	0
		out		41.03	150,598	110.45	408,828	36.50	113,322	4.06	12,256
		18.05	104,316	124.53	686,069	3.69	26,847	14.23	94,985	9.47	62,883
s1196	out	0.19	565	0.19	565	0.23	565	0.38	565	0.92	565
(54, 97)	92.19	0.14	0	0.16	0	0.20	0	0.34	0	0.89	0
(560, 2)	0.08*	**0.13**	0	0.14	0	0.20	0	0.34	0	0.87	0
		26.16	77,019	158.19	372,129	7.22	23,348	26.97	80,264	17.64	48,114
		13.22	82,294	1.02	5,920	2.50	15,621	1.19	6,024	1.47	3,516
s1488	out	0.20	708	0.20	667	0.25	667	0.44	667	1.06	667
(47, 67)	33.48	0.14	0	0.16	0	0.22	0	0.44	0	0.99	0
(667, 2)	0.13*	**0.13**	0	0.16	0	0.20	0	0.47	0	0.99	0
		21.75	74,658	1.67	5,499	4.22	14,445	1.84	5,372	1.80	3,124
		7.30	41,798	19.69	108,768	4.81	27,711	7.00	41,977	2.06	8,104
s1494	out	0.20	665	0.22	665	0.27	665	0.45	665	1.11	665
(48, 69)	42.1	**0.16**	0	0.17	0	0.22	0	0.41	0	1.09	0
(661, 2)	0.11*	**0.16**	0	0.17	0	0.22	0	0.42	0	1.22	0
		9.67	24,849	27.28	65,859	7.86	19,678	11.48	28,793	3.03	6,484

Table 1. CPU time and nodes visited for solving belief networks derived from the ISCAS'89 circuits. Time limit 30 minutes. Number of flips for GLS is 10,000. Pseudo tree generated by a single run of the min-fill heuristic, without randomization. SamIam ran out of memory.

experiments on the optimization problem of finding the most probable explanation in belief networks. We implemented our algorithms in C++ and ran all experiments on a 2.4GHz Pentium 4 with 2GB of RAM running Windows XP.

5.1 Overview and Methodology

Algorithms. We evaluated the following AND/OR Branch-and-Bound hybrid algorithms with full caching and static mini-bucket heuristics:

- AOBB-C+SAT+SMB(i), which exploits the determinism in the network by applying unit resolution over the CNF encoding of the zero probability tuples of the probability tables. We used a unit resolution scheme based on the one available in the `zChaff` SAT solver [13].
- AOBB-C+GLS+SMB(i), which exploits a good initial upper bound obtained by a guided local search algorithm. We used the GLS implementation for belief networks available from [8].
- AOBB-C+SAT+GLS+SMB(i), which combines the previous two approaches.

We compare these algorithms against the baseline AND/OR Branch-and-Bound with full caching and mini-bucket heuristics, AOBB-C+SMB(i). We also ran the best-first search version of the algorithm, denoted by AOBF-C+SMB(i), but the algorithm did not exploit any of the above principles. The guiding pseudo trees were constructed using both the min-fill and the hypergraph partitioning heuristics, described earlier.

We also compared with the SamIam version 2.3.2 software package (*available at http://reasoning.cs.ucla.edu*). SamIam is a public implementation of Recursive Conditioning [3] which can also be viewed as an AND/OR search algorithm, namely it explores a context minimal AND/OR graph [5]. Since any MPE problem instance can be converted into an equivalent 0-1 Integer Linear Program [15], we also ran the ILOG CPLEX 11.0 solver, with default settings ($i.e.$, best-bound control strategy, strong branching based variable ordering heuristic, and cutting planes).

Benchmarks. We tested the performance of the AND/OR search algorithms on belief networks derived from the ISCAS'89 digital circuits and genetic linkage analysis networks. All of these networks contain a significant amount of determinism.

Measures of performance. We report CPU time in seconds and the number of nodes visited. We also specify the number of variables (n), number of evidence variables (e), maximum domain size (k), the depth of the pseudo tree (h) and the induced width of the graph (w^*) for each problem instance. When evidence is asserted in the network, w^* and h are computed after the evidence variables are removed from the graph. We also report the time required by GLS to compute the initial upper bound. Note that in each domain we ran GLS for a fixed number of flips. Moreover, AOBB-C+GLS+SMB(i) and AOBB-C+SAT+GLS+SMB(i) do not include the GLS running time, because GLS can be tuned independently for each problem instance to minimize its running time. The best performance points are highlighted. In each table, "-" denotes that the respective algorithm exceeded the time limit. Similarly, "out" stands for exceeding the 2GB memory limit. A "*" by the GLS running time indicates that it found the optimal solution to the respective problem instance.

5.2 Empirical Results

ISCAS'89 Circuits[3] are a common benchmark used in formal verification and diagnosis. For our purpose, we converted each of these circuits into a belief network by removing flip-flops and buffers in a standard way, creating a deterministic conditional probability table for each gate and putting uniform distributions on the input signals.

Table 1 displays the results obtained on 5 ISCAS'89 circuits. We see that constraint propagation via unit resolution plays a dramatic role on this domain rendering the search space almost backtrack-free, across all reported i-bounds. For instance, on the s953 circuit, AOBB-C+SAT+SMB(6) is 3 orders of magnitude faster than AOBB-C+SMB(6) and 2 orders of magnitude faster than CPLEX, respectively, while AOBF-C+SMB(6) exceeded the memory limit. When looking at the AND/OR Branch-and-Bound algorithms that exploit a local search based initial upper bound, namely AOBB-C+GLS+SMB(i) and AOBB-C+SAT+GLS+SMB(i), we see that they did not expand any nodes. This is because the upper bound obtained by GLS, which was the optimal solution in this case, was equal to the the mini-bucket lower bound computed at the root node.

[3] available at http://www.fm.vslib.cz/kes/asic/iscas

		hypergraph pseudo tree				min-fill pseudo tree				
pedigree (n, d)	SamIam Superlink CPLEX GLS	(w*, h)	MBE(i) AOBB-C+SMB(i) AOBB-C+SAT+SMB(i) AOBB-C+GLS+SMB(i) AOBF-C+SMB(i) i=20 time	nodes	MBE(i) AOBB-C+SMB(i) AOBB-C+SAT+SMB(i) AOBB-C+GLS+SMB(i) AOBF-C+SMB(i) i=22 time	nodes	(w*, h)	MBE(i) AOBB-C+SMB(i) AOBB-C+SAT+SMB(i) AOBB-C+GLS+SMB(i) AOBF-C+SMB(i) i=20 time	nodes	MBE(i) AOBB-C+SMB(i) AOBB-C+SAT+SMB(i) AOBB-C+GLS+SMB(i) AOBF-C+SMB(i) i=22 time	nodes
ped7 (868, 4)	out - out 13.32	(36, 60)	25.26 30504.84 31701.54 30349.92 out	285,084,124 285,084,124 284,635,328	164.49 3005.66 3116.07 **2955.06** out	27,761,219 27,761,219 27,371,526	(32, 133)	117.03 - - - out	- - -	out	
ped9 (936, 7)	out - out 12.85*	(35, 58)	67.93 8922.81 10075.90 8866.40 out	117,328,162 117,328,162 117,011,941	300.06 **3292.30** 3657.91 3336.86 out	40,251,723 40,251,723 40,251,661	(27, 130)	76.31 1434.74 1515.50 **1163.09** out	15,825,340 15,825,340 12,444,961	out	
ped19 (693, 5)	out - out 10.23	(35, 53)	59.31 45075.31 47986.66 44585.84 out	466,748,365 466,748,365 459,741,495	150.38 8321.42 8774.51 **8070.95** out	90,665,870 90,665,870 87,060,723	(24, 122)	out		out	
ped34 (923, 4)	out - out 13.99	(34, 60)	42.21 67647.42 74020.63 64136.36 out	1,293,350,829 1,293,350,829 1,230,870,576	209.51 11719.28 12847.33 **11005.18** out	220,199,927 220,199,927 218,890,668	(32, 127)	out		out	
ped41 (886, 5)	out - out 12.28*	(36, 61)	35.41 3891.86 4055.15 3869.31 out	31,731,270 31,731,270 31,729,654	111.24 380.01 390.93 **374.95** out	2,318,544 2,318,544 2,317,321	(33, 128)	out		out	
ped44 (644, 4)	out - out 9.84	(31, 52)	32.92 3597.12 3904.39 3580.32 out	62,385,573 60,709,547 62,392,439	140.81 204.96 215.46 **196.57** out	1,355,595 1,355,595 1,213,051	(26, 73)	57.88 112.60 127.42 **95.09** out	1,114,641 1,114,641 752,970	344.68 385.30 385.47 366.18 out	668,737 668,737 447,514

Table 2. CPU time and nodes visited for solving genetic linkage analysis networks. Pseudo trees created using randomized min-fill and hypergraph partitioning heuristics. Time limit 24 hours. The maximum number of flips for GLS was set to 1,000,000.

Genetic Linkage Analysis. Table 2 displays the results obtained for 6 hard linkage analysis networks (*available at http://bioinfo.cs.technion.ac.il/superlink*) using randomized min-fill and hypergraph partitioning based pseudo trees. We selected the hypergraph based tree having the smallest depth over 100 independent runs (ties were broken on the smallest induced width). Similarly, the min-fill based tree was the one having the smallest induced width out of 100 tries (ties were broken on the smallest depth). For comparison, we also include results obtained with the state-of-the-art linkage analysis solver Superlink 1.6 [7]. To the best of our knowledge, these networks were never solved before for the MPE task ($i.e.$, maximum likelihood haplotype task [7]).

We see that the AND/OR Branch-and-Bound algorithms are the only ones that could solve all the problem instances, especially when guided by hypergraph partitioning based pseudo trees. This can be explained by the much smaller depth of these pseudo trees compared with the min-fill ones, which overcame the relatively poor quality of the mini-bucket heuristics obtained on these highly connected networks. Exploiting the GLS initial upper bound improved slightly the performance of AOBB-C+SMB(i). This was probably because AOBB-C+SMB(i) found the optimal or very close to optimal solutions quite early in the search. Similarly, we observe that applying unit resolution was not cost effective in this case. Notice also that Superlink exceeded the 24 hour time limit, whereas SamIam, CPLEX and AOBF-C+SMB(i) ran out of memory on all test instances.

6 CONCLUSION

The paper rests on three key contributions. First, we propose a principled approach for handling hard constraints in COPs within the AND/OR search framework, which builds upon progress made in the SAT community. Second, we allow for exploiting non-trivial initial upper bounds which are obtained by a local search scheme. Third, we investigate two heuristics for generating good quality pseudo trees: the min-fill heuristic which minimizes the induced width, and the hypergraph partitioning heuristic that minimizes the depth of the pseudo tree. We demonstrated empirically the impact of these factors on hard benchmarks, including some very challenging networks from the field of genetic linkage analysis.

ACKNOWLEDGEMENTS

This work was supported in part by the NSF grant IIS-0412854 and by the NIH grant R01-HG004175-02.

REFERENCES

[1] D. Allen and A. Darwiche, 'New advances in inference using recursive conditioning', in *UAI*, pp. 2–10, (2003).
[2] R. Bayardo and D. P. Miranker, 'On the space-time trade-off in solving constraint satisfaction problems', in *IJCAI*, pp. 558–562, (1995).
[3] A. Darwiche, 'Recursive conditioning', *Artificial Intelligence*, **125**(1-2), 5–41, (2001).
[4] R. Dechter, 'Mini-buckets: A general scheme of generating approximations in automated reasoning', in *IJCAI*, pp. 1297–1302, (1997).
[5] R. Dechter and R. Mateescu, 'AND/OR search spaces for graphical models', *Artificial Intelligence*, **171**(2-3), 73–106, (2007).
[6] E.Freuder and M.Quinn, 'Taking advantage of stable sets of variables in constraint satisfaction problems', in *IJCAI*, pp. 1076–1078, (1985).
[7] M. Fishelson and D. Geiger, 'Exact genetic linkage computations for general pedigrees', *Bioinformatics*, **18(1)**, 189–198, (2002).
[8] F. Hutter, H. Hoos, and T. Stutzle, 'Efficient stochastic local search for mpe solving', in *IJCAI*, pp. 169–174, (2005).
[9] U. Kjæaerulff, 'Triangulation of graph-based algorithms giving small total state space', Technical report, University of Aalborg, (1990).
[10] R. Marinescu and R. Dechter, 'And/or branch-and-bound for graphical models.', in *IJCAI*, pp. 224–229, (2005).
[11] R. Marinescu and R. Dechter, 'Memory intensive branch-and-bound search for graphical models.', in *AAAI*, (2006).
[12] R. Marinescu and R. Dechter, 'Best-first and/or search for graphical models', in *AAAI*, (2007).
[13] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, 'Chaff: Engineering an efficient sat solver', in *DAC*, (2001).
[14] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, 1988.
[15] E. Santos, 'On the generation of alternative explanations with implications for belief revision', in *UAI*, pp. 339–347, (1991).
[16] C. Voudouris, 'Guided local search for combinatorial optimization problems', Technical report, PhD Thesis. University of Essex, (1997).
[17] T. Walsh, 'Sat v csp', in *Principles and Practice of Constraint Programming (CP)*, pp. 441–456, (2000).