Compact Representation of Sets of Binary Constraints

Jussi Rintanen[1]

Abstract. We address the problem of representing big sets of binary constraints compactly. Binary constraints in the form of 2-literal clauses are ubiquitous in propositional formulae that represent real-world problems ranging from model-checking problems in computer-aided verification to AI planning problems. Current satisfiability and constraint solvers are applicable to very big problems, and in some cases the physical size of the problem representations prevents solving the problems, not their computational difficulty. Our work is motivated by this observation.

We propose graph-theoretic techniques based on cliques and bicliques for compactly representing big sets of binary constraints that have the form of 2-literal clauses. An n, m biclique in a graph associated with the constraints can be very compactly represented with only $n + m$ binary constraints and one auxiliary variable. Cliques in the graph are associated with *at-most-one* constraints, and can be represented with a logarithmic number of binary constraints. The clique representation turns out to be a special case of the biclique representation. We demonstrate the effectiveness of the biclique representation in making the representation of big planning problems practical.

1 Introduction

Binary constraints in the form of binary clauses $l_1 \vee l_2$ are very common in many applications of CSPs and propositional satisfiability. For example, most of the clauses in the representation of AI planning problems as CNF formulae in the propositional logic are binary clauses [11]. In structured formulae stemming from real-world applications, sets of binary clauses have regularities that make efficient reasoning and more compact representation possible. The goal of this work is to develop techniques for representing big sets of binary clauses more compactly.

The applications that motivate our work are planning and model-checking. The *planning as satisfiability* approach was introduced by Kautz and Selman [10, 11] and later generalized to a wide class of model-checking problems [2]. Current generation of satisfiability algorithms can solve planning and model-checking problems with thousands of variables and tens of thousands of actions. Interestingly, an obstacle to the scalability of the approach to still bigger problems is in some cases the enormous size of the propositional formulae.

For AI planning, for instance, there exist asymptotically optimal linear size translations into the propositional logic [14]. However, a part of the problem-specific information which is necessary for efficient reasoning, in the form of 2-literal *invariants* [3, 13], has a size that is in the worst-case quadratic in the size of the problem instance. Typically these 2-literal clauses constitute between 50 and 95 per cent of the formulae, or even more. An example of a typical invariant in a planning problem is for example ¬in(A,London)∨¬in(A,Paris) which indicates that an object cannot be in two locations simultaneously. For n state variables there are $\mathcal{O}(n^2)$ invariants that are 2-literal clauses. For a planning problem with $n = 5000$ state variables and a formula that encodes the search for plans of 100 time points, if only one of the state variables is true at any given time, the corresponding set of binary clauses has cardinality $100 \times \frac{n(n-1)}{2} = 1249750000 \sim 1.2 \times 10^9$. If each clause takes 10 bytes then the total size of the representation is about 12 gigabytes. The problem of representing the constraint graphs compactly arises.

In this paper we propose techniques for representing the constraint graphs more compactly with cliques and bicliques. We show how constraint graphs that contain big cliques or bicliques can be represented more compactly than the explicit representation, and we also show that constraint graphs arising from practically interesting applications indeed do contain big cliques and bicliques. The presence of cliques is more obvious and they have been addressed in earlier research. The main contribution of our work is the introduction of techniques based on bicliques. The underlying ideas in the use of bicliques are simple and powerful, yet their importance has not been recognized before. We show that the clique based techniques are subsumed by the biclique based techniques.

The outline of this paper is as follows. Section 2 introduces the graph-theoretic concepts of cliques and bicliques and discusses their computational properties. In Section 3 we discuss the compact representation of cliques of constraint graphs. In Section 4 we address the compact representation of constraint graphs more generally in terms of bicliques. In Section 5 we show the effectiveness of the biclique representation in reducing the sizes of formulae that represent an industrial size planning problem.

2 Preliminaries: Cliques and Bicliques

The techniques presented in this paper are based on cliques and bicliques of graphs.

Definition 1. *Let $\langle N, E\rangle$ be an undirected graph. Then* a clique *is $C \subseteq N$ such that $\{n, n'\} \in E$ for every $n, n' \in C$ such that $n \neq n'$.*

Hence cliques are complete subgraphs. Identification of cliques is expensive. Testing whether a graph contains a clique of size n is NP-complete [9, 6]. Hence we cannot expect to have a polynomial-time algorithm that is guaranteed to find the biggest cliques of a graph. Approximation algorithms for identifying cliques and related subgraphs exist [8]. Hochbaum's algorithms are based on reducing the problem to integer programming and showing that the corresponding linear programs have integers solutions. Linear programming problems can be solved in polynomial time.

Instead of using Hochbaum's algorithm, we have used the simpler polynomial-time algorithm without approximation guarantees that is given in Figure 1. In many of our example applications (see Section

[1] National ICT Australia, Canberra Research Laboratory, Canberra, Australia

5) this algorithm identifies all the maximal cliques because no two maximal cliques share a node. Of course, maximal cliques of many graphs do not have this property.

```
 1:  procedure partition-to-cliques(N,E)
 2:  n := 1;
 3:  S[1] := N;
 4:  change := true;
 5:  while change do
 6:     change := false;
 7:     if there are l1, l2 ∈ S[i] for some i ∈ {1, ..., n}
 8:           such that {l1, l2} ∉ E and l1 ≠ l2 then
 9:        change := true;
10:        n := n + 1;
11:        S[n] := {l2} ∪ {l|{l, l2} ∈ E};
12:        S[i] := S[i]\S[n];
13:     end if
14:  end while
```

Figure 1. A polynomial-time algorithm for partitioning the nodes of a graph to cliques. It identifies candidate cliques $S[i]$ that contain two nodes without an edge between them, and splits them to two new candidate cliques.

Theorem 2. *The algorithm* partition-to-cliques *with input $G = \langle N, E\rangle$ terminates after a number of execution steps that is polynomial in the size of G, and after termination $S[i]$ is a clique of G for every $i \in \{1, \ldots, n\}$.*

Proof. Sketch: The loop can only exit if for every $i \in \{1, \ldots, n\}$ there is an edge between every two nodes in $S[i]$. Hence on termination every $S[i]$ is a clique.

Since at every iteration n is incremented and the sets $S[1], \ldots, S[n]$ form a partition of all literals to nonempty sets, the algorithm terminates at the latest when all $S[i]$ have size 1. Hence the number of literals is an upper bound on the number of iterations. On every iteration the amount of computation is polynomial. □

A concept related to cliques is that of bicliques.

Definition 3. *Let $\langle N, E\rangle$ be an undirected graph. A* biclique *is a pair C, C' of sets $C \subseteq N$ and $C' \subseteq N$ such that $C \cap C' = \emptyset$ and $\{\{n_1, n_2\}|n_1 \in C, n_2 \in C'\} \subseteq E$.*

In other words, bicliques are complete bipartite subgraphs of a graph. The problem of testing for existence of bicliques under several size measures is NP-complete [16, 6, 12].

We have used a simple polynomial-time algorithm for identifying bicliques in constraint graphs. The algorithm is given in Figure 2. It

```
 1:  procedure identify-biclique(N,E)
 2:  C1 := ∅;
 3:  C2 := N;        (* C1, C2 is now a trivial 0-edge biclique *)
 4:  repeat
 5:     n := a node in N\C1;                    (* Maximizing... *)
 6:                    (* |C2| for the new value of C2 if C1 = ∅ *)
 7:                (* (|C1| × |C2|) − (|C1| + |C2|) if C1 ≠ ∅ *)
 8:     C1 := C1 ∪ {n}
 9:     C2 := C2 ∩ {m ∈ N\C1|{n, m} ∈ E};
10:  until |C1| × |C2| − (|C1| + |C2|) does not increase;
```

Figure 2. A polynomial-time algorithm for finding bicliques

starts with the dummy biclique $\emptyset, N$, and repeatedly adds nodes to the first part. Nodes from the second part are removed if there is no edge between them and the new node in the first part. The nodes are chosen to maximize the value of the biclique that is being constructed (the size reduction obtained by the compact representation.) The algorithm terminates when the value of the biclique does not increase any more. The passage from $\emptyset, N$ to $\{n\}, N'$ at the first step never increases the value of the biclique and this case is handled specially.

Theorem 4. *The algorithm* identify-biclique *with input $G = \langle N, E\rangle$ terminates after a number of execution steps that is polynomial in the size of G, and after termination C_1, C_2 is a biclique of G.*

Proof. First verify that after every iteration C_1, C_2 is a biclique.

The number of iterations is bounded by $|N|$ because after $|N| - 1$ iterations $C_1 = N\backslash\{m\}$ for some m and $C_2 \subseteq \{m\}$ and therefore $|C_1| \times |C_2| - (|C_1| + |C_2|)$ is $(|N| - 1) \times 1 - (|N| - 1 + 1) = -1$ or $(|N| - 1) \times 0 - (|N| - 1 - 0) = -|N| + 1$, and after N iterations it is $|N| \times 0 - (|N| + 0) = -|N|$, and the iteration terminates. Polynomial runtime follows because computation on every iteration takes polynomial time. □

3 Cliques in Constraint Graphs

We consider constraint graphs $\langle N, E\rangle$ that represent sets S of 2-literal clauses $l \vee l'$. The set N of nodes of the graph consists of all the literals a and $\neg a$ for $a \in A$ where A is the set of propositional variables, and there is an (undirected) edge $\{l, l'\} \in E$ between the nodes l and l' if and only if $l \vee l' \in S$ or $l' \vee l \in S$.

The clause set may have a very irregular structure and the best way of representing it may be as is, but in many cases there are regularities that make a more compact representation possible. The simplest regularity is perhaps the presence of an *at-most-one* constraint for a subset V of the variables, forbidding more than one of the variables to be true at a time. Hence for all $\{v, v'\} \subseteq V$ such that $v \neq v'$, the constraint graph has an edge $\{\neg v, \neg v'\} \in E$. The set of literals $\{\neg v|v \in V\}$ forms a clique in the constraint graph.

In the following sections we discuss ways of representing cliques and other regularities of constraint graphs more compactly. Irrespective of the details of the representation, the more compact representation of the cliques in a constraint graph proceeds in the same way.

1. Identify a big clique in the constraint graph.
2. If only small cliques were found, go to the last step.
3. Construct a propositional formula that represents the binary constraints corresponding to the clique more compactly (see Sections 3.2 and 3.3).
4. Remove the edges of the clique from the constraint graph.
5. Continue from step 1.
6. Represent the binary constraints corresponding to the remaining edges in the constraint graph explicitly as 2-literal clauses.

In the next sections we discuss three alternative representations of cliques of the constraint graph, starting from the trivial explicit representation.

3.1 Explicit $O(n^2)$ Representation

For a set C of literals forming a clique in the constraint graph the most obvious representation is as 2-literal clauses

$$\{l \vee l'|\{l, l'\} \subseteq C, l \neq l'\}.$$

This quadratic representation of the clique is illustrated in Figure 3 by 5 mutually exclusive literals of the form $v = i$. The number of non-equivalent clauses is $\frac{n(n-1)}{2}$ for n literals.

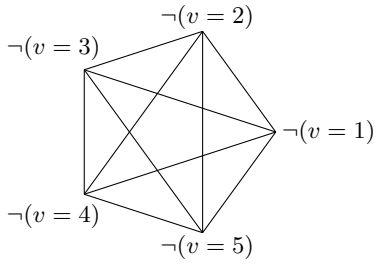

Figure 3. Representation of a clique of the constraint graph with $O(n^2)$ size and $O(1)$ additional propositional variables. An edge between literals $\neg(v = i)$ and $\neg(v = j)$ denotes their disjunction.

The $O(n^2)$ explicit representation of the constraints becomes less and less practical as n increases.

3.2 $O(n)$ Representation with $O(n)$ Auxiliary Variables

We have devised a linear-size representation that uses a linear number of auxiliary variables. This representation is illustrated in Figure 4. The representation can be viewed as a variant of the linear-size representations of interference constraints presented by Rintanen et al. [14] and the representation of many-valued propositions by Giunchiglia et al. [7].

The idea is to impose an arbitrary ordering on the n literals, and starting from each literal introduce a sequence of implications to the right through a sequence of auxiliary variables, leading to the negation of each literal on the right. When one of the literals is true, all of the auxiliary variables right from it must be true, and consequently all of the literals to the right must be false. This guarantees that if one of the literals is set true, all others are set false simply by applications of the unit resolution rule employed by most satisfiability algorithms.

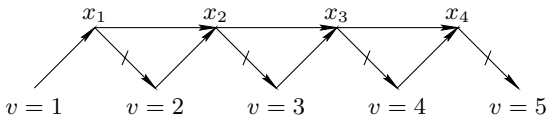

Figure 4. Representation of a clique in terms of $\mathcal{O}(n)$ clauses and additional variables. An edge $\rightarrow$ from l to l' denotes the formula $l \rightarrow l'$ and an edge $\not\rightarrow$ from l to l' denotes the formula $l \rightarrow \overline{l'}$.

The number of clauses in this linear-size representation is $3n - 4$ for $n \geq 2$. For 5 values the quadratic-size and linear-size representations respectively require 10 and 11 2-literal clauses, for 6 values respectively 15 and 14 clauses. Hence starting from 6 values the linear-size representation is smaller than the quadratic-size representation.

3.3 $O(n \log n)$ Representation with $O(\log n)$ Auxiliary Variables

Although the previous representation has an asymptotically optimal $O(n)$ size, the linear number of additional auxiliary variables may be considered high. Indeed, we can substantially decrease the number of auxiliary variables with the cost of a small increase in size.

The next representation has $\mathcal{O}(n \log n)$ size and $\mathcal{O}(\log n)$ auxiliary variables. Let $C = \{l_0, \ldots, l_{n-1}\}$ be a clique consisting of n literals. Let $x_0, \ldots, x_{m-1}$ be new $m = \lceil \log_2 n \rceil$ Boolean variables. The constraint that only one literal can be true at a time is imposed by associating a different binary number with each literal. Since these binary numbers are represented in terms of the same variables, at most one of the literals can be true at a time. The representation is

$$
\begin{aligned}
&l_0 \rightarrow (\text{bit}_0(0) \wedge \text{bit}_1(0) \wedge \cdots \wedge \text{bit}_{m-1}(0)) \\
&l_1 \rightarrow (\text{bit}_0(1) \wedge \text{bit}_1(1) \wedge \cdots \wedge \text{bit}_{m-1}(1)) \\
&\vdots \\
&l_{n-1} \rightarrow (\text{bit}_0(n-1) \wedge \text{bit}_1(n-1) \wedge \cdots \wedge \text{bit}_{m-1}(n-1))
\end{aligned}
$$

where $\text{bit}_i(j) = x_i$ if the ith bit of j is 1 and $\text{bit}_i(j) = \neg x_i$ if the ith bit of j is 0.

For n literals the number of the resulting 2-literal clauses is $n\lceil \log_2 n \rceil$. For 5 values the numbers of clauses are respectively 15 and 11 for this representation and the linear-size representation, and numbers of auxiliary variables are respectively 3 and 4. For 32 values the numbers of clauses are respectively 160 and 92 and number of auxiliary variables are respectively 5 and 31.

This representation has been used by Frisch and Peugniez [5] and in more recent works [1], and the idea of two redundant representations connected by channelling constraints is familiar from the CSP context [4, 15].

4 Bicliques of the Constraint Graph

The presence of cliques in constraint graphs seems to be tied to particular classes of planning, model-checking and constraint satisfaction problems, and cannot be claimed to be extremely common. Indeed, in standard planning benchmarks, cliques of size n restrict an n-valued state variable, which is represented in terms of n Boolean state variables, to have at most one value at a time.

There is a need for techniques for compactly representing more general classes of constraint graphs. It is easy to see by a simple combinatorial argument that most constraint graphs do not have a compact representation, but there are many practically important graphs that do have many regularities that may be taken advantage of.

It seems that bicliques in constraint graphs are much more common than cliques. First, by partitioning any clique C to two disjoint sets C_1 and C_2 yields a biclique C_1, C_2. So there are bicliques whenever there are cliques. Second, consider a subgraph C that is almost a clique but not quite. Let E' be the missing edges so that $\{\{n, n'\} | n \in C, n' \in C, n \neq n'\} \subseteq E \cup E'$. Now $C' = C \setminus \bigcup_{e \in E'} e$ is a clique, $C \setminus C', C'$ is a biclique because all the missing edges are between nodes in $C \setminus C'$, and $C \setminus C'$ may contain non-trivial bicliques.

A powerful discovery is that bicliques can be represented very compactly. When the literal sets C and C' form a biclique, the $|C| \cdot |C'|$ binary constraints can be represented by only $|C| + |C'|$ 2-literal clauses and only one auxiliary variable. For big bicliques this is a very big size reduction. This is illustrated in Figure 5. Notice that the original clause set can be obtained by resolving every pair of clauses that contain x and $\neg x$.

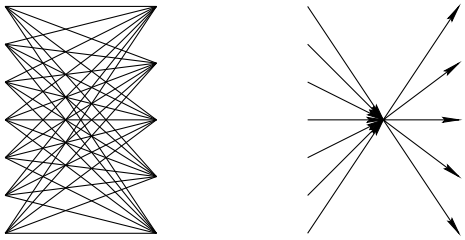

Figure 5. Representation of nm binary constraints forming an n, m biclique (left) in terms of only $n + m$ 2-literal clauses and one auxiliary variable (right). On the left, an edge between l and l' denotes the formula $l \vee l'$ (equivalently $\bar{l} \rightarrow l'$.) On the right, an edge $\rightarrow$ from l to the auxiliary variable x denotes $\bar{l} \rightarrow x$ and an edge from x to l' denotes $x \rightarrow l'$.

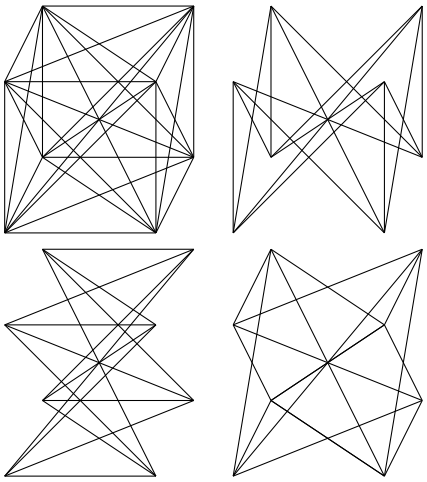

Figure 6. A clique of 8 nodes and 28 edges (upper left) decomposed to three 4,4 bicliques with 16 edges each. Division is along the up-down vertical axis (upper right), along the left-right horizontal axis (lower left), and along the front-back horizontal axis (lower right).

4.1 Relation to Compact Clique Representation

Decomposition of constraint graphs to cliques and their compact representation with only $O(n \log n)$ binary clauses can be understood in terms of bicliques. Hence the clique representation is subsumed by the biclique representation. A clique of n nodes corresponds to $\lceil \log_2 n \rceil$ orthogonal bicliques. Figures 6 and 7 illustrate the reduction from 28 edges of the original clique ($O(n^2)$) to 24 edges ($O(n \log n)$) in the biclique representation. This can also be illustrated by viewing the 8 nodes as 3-bit binary numbers. Now the three bicliques correspond to partitioning the 8 numbers by the values of the three digits. The partitions and bicliques are according to the first bit $\{000, 001, 010, 011\}, \{100, 101, 110, 111\}$, the second bit $\{000, 001, 100, 101\}, \{010, 011, 110, 111\}$ and the third $\{000, 010, 100, 110\}, \{001, 011, 101, 111\}$. The compact representation of these bicliques exactly corresponds to the $n \log n$ representation of cliques from Section 3.3.

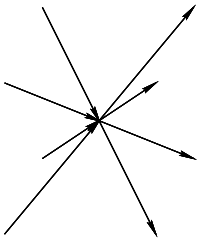

Figure 7. The invariants of the lower left biclique in Figure 6 represented more compactly in terms of an intermediate node.

5 Application: AI Planning

In most of the standard planning benchmarks all of the edges in the constraint graph stem from many-valued state variables that are encoded in terms of binary variables: the graph consists of cliques which say that each many-valued state variable can have only one value at a time. The representations from Section 3 encode these graphs compactly.

For the blocks world problems, if there are n blocks, the constraint that there can be at most one block below and above a block requires $2 \times \frac{n(n-1)}{2} = n(n-1)$ 2-literal clauses. For representing these cliques compactly by using the $O(n)$ representation only $6n - 8$ 2-literal clauses are needed. For 20 blocks this means a reduction from $20 \times 20 \times 19 = 7600$ to $20 \times (6 \times 20 - 8) = 2240$ clauses. For 40 blocks the reduction is from 62400 to 9280.

However, for many other problems the structure of constraint graphs is more complex and there are few cliques. One problem with only few big cliques is the airport scheduling problem from the 2004 planning competition. Most of this problem series can be efficiently solved by the planning as satisfiability approach but the biggest instances require the generation of huge propositional formulae with sizes of several gigabytes that exceed the physical memory restrictions of current 32-bit microprocessors. The invariants constitute more than 90 per cent of all the clauses in the CNF representation of the problems, and only a fraction of them forms cliques. The locations of airplanes correspond to many-valued state variables, but invariants on a number of related state variables that control the legal movement of airplanes through the airport do not.

We have very successfully used bicliques for making the representation of the invariants more compact. See Table 1. The size reduction of the formulae by a factor of about 9 makes it possible to generate all the formulae required for solving the problem series, and solve almost all of them efficiently save some of the last ones.

Interestingly, despite the big reduction in the formula sizes, those instances that were solvable before the reduction in formula size, the SAT solver (HaifaSAT, Siege) runtimes are improved not at all or only minimally. The biggest reductions in total runtimes are due to faster disk I/O and processing times during formula generation.

In addition to the airport problems, for other problems substantial reductions in the number of clauses for invariants are obtained, but the total reductions in the formula sizes are smaller, for example for the 30 block blocks world problems 50 per cent, because the invariants do not as strongly dominate the size of the formulae. For the simplest problems our biclique algorithm finds all maximal bicliques. This is probably not true for more complicated problems, for example the airport problem. Hence it would be interesting to test algorithms with approximation guarantees [8].

instance	clauses for invariants before	after	size in MB before	after
21_4halfMUC_P2	182094	13191	2.59	0.37
22_4halfMUC_P3	275927	21388	4.06	0.58
23_4halfMUC_P4	381675	31776	5.60	0.84
24_4halfMUC_P4	383791	30407	5.72	0.90
25_4halfMUC_P5	478455	41719	7.24	1.18
26_4halfMUC_P6	587951	50247	8.85	1.43
27_4halfMUC_P6	572292	53721	9.01	1.57
28_4halfMUC_P7	670530	66060	10.62	1.89
36_5MUC_P2	325136	18872	4.68	0.52
37_5MUC_P3	490971	30681	7.40	0.93
38_5MUC_P3	487600	29464	7.30	0.86
39_5MUC_P4	655616	44647	10.08	1.34
40_5MUC_P4	657309	43872	10.04	1.27
41_5MUC_P4	653940	42314	9.93	1.20

Table 1. Size reduction by the biclique representation. For each problem instance we give the number of clauses when each invariant is represented by one 2-literal clause and when bicliques of the constraint graph are represented compactly. The formula sizes are for DIMACS-encoded CNF formulae per time point, including the invariants and the rest of encoding of the planning problem. For example for 100 time points the total formula size is obtained by multiplying by 100. The first problem instance has 1696 state variables, the last has 4957. Solving the last instances of the series (42 to 50) require formulae with sizes up to 8 gigabytes which do not fit in the memory of a computer with a 32-bit address space. The shortest plan for instance 22 has length 53 and the corresponding formula without compression is 222 megabytes in the DIMACS CNF format, with clique compression 187 megabytes, and with biclique compression 31 megabytes. The numbers of clauses for invariants are respectively 275927, 231673 and 21388.

We also made a small experiment with constraint graphs that represent interference constraints for preventing simultaneous actions if they interfere [11]. These constraint graphs have in the worst case n^2 edges for n actions but there is a specialized representation with an asymptotically optimal linear size [14]. The biclique representation was often moderately or much smaller than the explicit and the specialized $\mathcal{O}(n)$ representation, but in some cases there was very little reduction and the specialized $\mathcal{O}(n)$ representation was by far the most compact. This possibility makes the biclique representation unattractive for this application.

6 Conclusions and Related Work

We have considered the problem of representing big constraint graphs more compactly based on representations that involve decomposing parts of the graphs to cliques and bicliques. The biclique representation, which is the new contribution of this work, turned out to be the more general representation as the clique representation is subsumed by it.

Constraint graphs with n nodes may have n^2 edges and our compact biclique representation does not improve this $\mathcal{O}(n^2)$ worst-case size. However, our experiments in Section 5 show that for practically interesting constraint graphs big size reductions are possible, by a factor of 10, and this may be the difference between an impractically big SAT problem (with respect to the memory requirements) and a relatively easily solvable one.

Acknowledgements

This research was supported by DFG grant RI 1177/2-1 (during the work at the Albert-Ludwigs-Universität Freiburg) and by National ICT Australia (NICTA). NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian National Research Council.

REFERENCES

[1] Carlos Ansótegui and Felip Manya, 'Mapping problems with finite-domain variables into problems with Boolean variables', in *SAT 2004 - Theory and Applications of Satisfiability Testing: 7th International Conference, SAT 2004, Vancouver, BC, Canada, May 10-13, 2004, Revised Selected Papers*, eds., Holger Hoos and David G. Mitchell, number 3542 in Lecture Notes in Computer Science, pp. 1–15. Springer-Verlag, (2005).

[2] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu, 'Symbolic model checking without BDDs', in *Tools and Algorithms for the Construction and Analysis of Systems, Proceedings of 5th International Conference, TACAS'99*, ed., W. R. Cleaveland, volume 1579 of *Lecture Notes in Computer Science*, pp. 193–207. Springer-Verlag, (1999).

[3] Avrim L. Blum and Merrick L. Furst, 'Fast planning through planning graph analysis', *Artificial Intelligence*, **90**(1-2), 281–300, (1997).

[4] B. M. W. Cheng, K. M. F. Choi, J. H. M. Lee, and J. C. K. Wu, 'Increasing constraint propagation by redundant modeling: an experience report', *Constraints*, **4**, 167–192, (1999).

[5] Alan M. Frisch and Timothy J. Peugniez, 'Solving non-Boolean satisfiability problems with stochastic local search', in *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, ed., Bernhard Nebel, pp. 282–288. Morgan Kaufmann Publishers, (2001).

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability*, W. H. Freeman and Company, San Francisco, 1979.

[7] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner, 'Nonmonotonic causal theories', *Artificial Intelligence*, 49–104, (2004).

[8] Dorit S. Hochbaum, 'Approximating clique and biclique problems', *Journal of Algorithms*, **29**, 174–200, (1998).

[9] R.M. Karp, 'Reducibility among combinatorial problems', in *Complexity of Computer Computations*, eds., R.E. Miller and J. W. Thatcher, pp. 85–103. Plenum Press, (1972).

[10] Henry Kautz and Bart Selman, 'Planning as satisfiability', in *Proceedings of the 10th European Conference on Artificial Intelligence*, ed., Bernd Neumann, pp. 359–363. John Wiley & Sons, (1992).

[11] Henry Kautz and Bart Selman, 'Pushing the envelope: planning, propositional logic, and stochastic search', in *Proceedings of the 13th National Conference on Artificial Intelligence and the 8th Innovative Applications of Artificial Intelligence Conference*, pp. 1194–1201. AAAI Press, (August 1996).

[12] René Peeters, 'The maximum edge biclique problem is NP-complete', *Discrete Applied Mathematics*, **131**(3), 651–654, (2003).

[13] Jussi Rintanen, 'A planning algorithm not based on directional search', in *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR '98)*, eds., A. G. Cohn, L. K. Schubert, and S. C. Shapiro, pp. 617–624. Morgan Kaufmann Publishers, (June 1998).

[14] Jussi Rintanen, Keijo Heljanko, and Ilkka Niemelä, 'Planning as satisfiability: parallel plans and algorithms for plan search', Report 216, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, (2005).

[15] Barbara. M. Smith, K. Stergiou, and T. Walsh, 'Using auxiliary variables and implied constraints to model non-binary problems', in *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000) and the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-2000)*, pp. 182–187. AAAI Press, (2000).

[16] M. Yannakakis, 'Node- and edge-deletion NP-complete problems', in *Proceedings of the Tenth Annual ACM Symposium on the Theory of Computing*, pp. 253–264. The Association for Computing Machinery, (1978).