A Covering Problem for Hypercubes

Jörg Hoffmann
Max-Planck-Institut für Informatik
Saarbrücken, Germany
hoffmann@mpi-sb.mpg.de

Sebastian Kupferschmid
Universität Freiburg, Institut für Informatik
Freiburg, Germany
kupfersc@informatik.uni-freiburg.de

Abstract

We introduce a new NP-complete problem asking if a "query" hypercube is (not) covered by a set of other "evidence" hypercubes. This comes down to a form of constraint reasoning asking for the satisfiability of a CNF formula where the logical atoms are inequalities over single variables, with possibly infinite variable domains. We empirically investigate the location of the phase transition regions in two random distributions of problem instances. We introduce a solution method that iteratively constructs a representation of the non-covered part of the query cube. In particular, the method is not based on backtracking. Our experiments show that the method is, in a significant range of instances, superior to the backtracking method that results from translation to SAT, and application of a state-of-the-art DP-based SAT solver.

1 Introduction

We introduce a new NP-complete problem, *QCOVER*, asking if there is a point in a given n-dimensional "query" hypercube that is not covered by – contained in the union of – a set of other n-dimensional "evidence" hypercubes. An n-dimensional hypercube is a cross product of n intervals. Covering problems of this kind arise, e.g., in the context of regression planning with numeric state variables [6]. Assume there are n numeric variables, which can take on rational values. A (sub-)goal for a single variable requires that the value of the variable be contained in an interval. A (sub-)goal for all variables is a hypercube. Assume one already knows that each single one of a set of such (sub-)goals, the evidence hypercubes, is unsolvable. Then, given a new (sub-)goal, the query hypercube, this goal is already proved unsolvable if and only if the query cube is completely covered by the evidence cubes.

More generally, the hypercube covering problem we consider is a form of constraint reasoning asking for the satisfiability of a CNF formula where the logical atoms are inequalities over single variables, with possibly infinite variable domains. Roughly, the correspondence is the following (we will be more precise later in the paper). An interval in our context can be defined as a statement of the form $l < [\leq]\ x < [\leq]\ u$ where "x" is a variable, "l" and "u" are members of x's domain, and "$<$" is a total order defined over this domain.[1] The hypercube covering problem is then a constraint problem with n variables x^d, one for each dimension d. The query hypercube specifies a region inside which the solution must lie, the evidence cubes specify regions inside which the solution must *not* lie. A hypercube corresponds to a conjunction of inequalities of the form $c < [\leq, >, \geq]\ x^d$. So the complement of a hypercube (of an evidence hypercube) corresponds to a *dis*junction of such inequalities, and the overall problem is a conjunction of disjunctive constraints. Vice versa, any conjunction of such disjunctive constraints can be expressed as hypercubes (if a disjunctive constraint does not mention a variable x^d, then the interval in dimension d is the whole variable domain).

We empirically explore two random distributions of QCOVER instances. The first one, which we call *Random QCOVER*, chooses the end points for all intervals uniformly from a set of m possible values. The second one, which we call *Random 3-QCOVER*, is similar to the fixed clause-length model for generating random 3SAT instances [9]. It always selects the query cube to be the cross-product of the (whole) variable domains, and, in the evidence cubes, assigns the whole variable domains to all but 3 randomly chosen dimensions. Details and our motivation for this choice of distributions are in Section 3 below. For both distributions, we investigate the location of the phase transition regions. As it turns out, Random 3-QCOVER shows a typical phase transition behaviour while Random QCOVER shows no such behaviour. We plot the 50% point, i.e., the number k of evidence cubes at which our random instances have (empirically) equal probability of being satisfiable or unsatisfiable, against different values of n and m. We use instances from the 50% points to evaluate different solution methods.

Today, in most cases the empirically most efficient solution methods for satisfiability problems are backtracking methods. Such methods are depth-first searches that split, in each search node, the search space along the possible values of a variable. An incomplete but correct polynomial propagation of constraints is used to determine inconsistencies (con-

[1] Throughout the paper, by square parentheses "$symb1\ [symb2]$" in mathematical notations we denote alternative possibilities, i.e. that $symb2$ can be substituted for $symb1$.

flicts) early on in the search tree, and analysis of conflicts is used to prune unnecessary search branches. Methods of this kind have proved successful for solving constraint satisfaction problems [7; 11]. In particular, the famous Davis Putnam procedure [2] for SAT is such a method, and its modern descendants (e.g., [8; 10; 5]) still constitute the state-of-the-art in determining the solvability of propositional CNF formulas. In our work, we have implemented a backtracking method for QCOVER by plugging a straightforward translation to SAT into Chaff [10], a state-of-the-art DP-based SAT solver. To contrast this method, we have also developed an algorithm that, instead of backtracking over possible variable values, iteratively constructs a representation of the non-covered part of the query cube. The algorithm maintains a set $\mathcal{Q}$ of hypercubes that initially contains only the query cube itself. Then iteratively all evidence cubes E are "eliminated" by subtracting them from all cubes Q in $\mathcal{Q}$. The result $Q \setminus E$ of such a subtraction is not necessarily a hypercube; we represent it as a *set* of hypercubes. When all evidence cubes have been eliminated, the non-covered part of the query cube is given by the union of the hypercubes in $\mathcal{Q}$ (if the query cube is completely covered by the evidence cubes, then $\mathcal{Q}$ becomes empty at some point). We call our algorithm *cube elimination*. In SAT, where all variables are boolean and have only two possible values, cube elimination comes down to transforming the CNF into a DNF. Intuitively it seems unlikely that such an algorithm can outperform state-of-the-art DP-solvers, and indeed that's what we found in some limited experiments. For our QCOVER distribution, however, our experiments show that cube elimination is often superior the backtracking method implemented by Chaff. The reason for this is, apparently, that backtracking becomes hard when the variable domains (the number of possible values in a dimension) become larger.[2]

The paper is organised as follows. Section 2 defines the QCOVER problem and shows its correspondence to CNF satisfiability. Section 3 describes our experiments on the location of the phase transition regions in our random distributions. Section 4 introduces the cube elimination algorithm, and Section 5 evaluates it against the backtracking method. In Section 6, we discuss our results and related work, and outline future directions.

2 QCOVER

We denote an *interval* as a constraint of the form $l < [\leq]\ x < [\leq]\ u$, where x is a variable associated with a domain D_x, and $<$ is a total order on D_x. We require that $l, r \in D_x$; to model missing lower (upper) bounds, we also allow $l = -\infty$ ($r = \infty$) where $-\infty$ (∞) is not contained in D_x, and $<$ is extended such that $-\infty < d$ for all $d \in D_x$ ($d < \infty$ for all $d \in D_x$). An *n-dimensional hypercube* is a tuple of n intervals over pairwise different variables. We identify hypercubes with their content, i.e., with the set of value vectors (points) that satisfy all interval constraints. When talking about more than one n-dimensional hypercube, we assume that all the hypercubes share the same variables (domains/orders).

Definition 1 *Let QCOVER denote the following problem: Given an n-dimensional hypercube Q, and a set of n-dimensional hypercubes $\mathcal{E}$, is there a point in Q that is not contained in $\bigcup_{E \in \mathcal{E}} E$?*

We say that an instance is *satisfiable* if the answer to the question is "yes". We continue to refer to Q as the *query* cube, and to the hypercubes in $\mathcal{E}$ as the *evidence* cubes. As said before, QCOVER instances are essentially CNFs of single-variable inequalities. Assume we have a formula ϕ that is a conjunction of disjunctive constraints (clauses) of the form $(c_1\, OP_1\, x_1) \vee \cdots \vee (c_k\, OP_k\, x_k)$ where $OP_i \in \{<, \leq, \geq, >\}$. The negation of such a clause is the conjunction $(c_1\, \overline{OP_1}\, x_1) \wedge \cdots \wedge (c_k\, \overline{OP_k}\, x_k)$ (where $\overline{OP}$ is the inverse comparison operator to OP, e.g. "$\overline{\leq}$"="$>$"). Such a conjunction corresponds to a hypercube in the variables used by the overall formula: for each variable x, let $l\, OP\text{-}l\, x$ be the strongest lower bound on x contained in the conjunction (or $-\infty < x$ if there is no lower bound on x). Similarly, let $x\, OP\text{-}u\, u$ be the strongest upper bound on x (or $x < \infty$ if there is no upper bound on x).[3] Then the intervals $l\, OP\text{-}l\, x\, OP\text{-}u\, u$ form a hypercube that contains exactly those points/value vectors that satisfy the conjunction. Overall, the formula ϕ corresponds to a QCOVER instance as follows. The conjunction of the unit clauses in ϕ forms the query hypercube (if there are no unit clauses then the resulting hypercube contains the whole variable value space). The negation of each non-unit clause in ϕ forms one evidence hypercube. The formula is satisfiable iff the QCOVER instance is.

Proposition 1 *QCOVER is NP-complete.*

Proof: Hardness holds because propositional CNFs are a special case of formulas ϕ as above: all variable domains are $\{0, 1\}$ with $0 < 1$, positive literals x are lower bounds $1 \leq x$, negative literals $\neg x$ are upper bounds $x \leq 0$. Membership holds because one can guess a point, and verify in polynomial time if it is contained in Q but in none of the evidence cubes. ■

Any QCOVER instance can be turned into a formula ϕ by doing the above transformation the other way round.

3 Phase Transitions

We empirically explore two random distributions of QCOVER instances, Random QCOVER and Random 3-QCOVER. Random QCOVER is just the most straightforward distribution that, given the parameter m, chooses every interval endpoint uniformly from the set $\{1, \ldots, m\}$ (the intervals are closed, i.e. of the form $l \leq x \leq u$). To avoid trivially superfluous evidence cubes/trivially unsatisfiable instances, during the generation we throw away (and regenerate) evidence cubes that have an empty intersection with

[2] Our translation to SAT implements the multiple-valued variable for each dimension by a set of boolean variables a that cover (only) the relevant case distinctions in that dimension; binary exclusion clauses of the form $\neg a \vee \neg a'$ ensure that (by unit propagation) only one variable a can be set to 1 at a time. See the details in Section 5.

[3] A lower bound $l\, OP\, x$ is stronger than a lower bound $l'\, OP'\, x$ iff $l' < l$, or $l' = l$ and OP ="$<$" and OP' ="$\leq$". Similarly for upper bounds.

the query cube/that contain the whole query cube. Random 3-QCOVER always selects the query cube to be $(1 \leq x \leq m)^n$, i.e., the whole variable value space. The evidence cubes are generated by uniformly choosing 3 dimensions; the interval endpoints in these dimensions are, as before, chosen uniformly from $\{1, \ldots, m\}$, the intervals in all other dimensions are set to $1 \leq x \leq m$.

Our reason for exploring Random QCOVER is that it is the simplest possible random distribution. Our reason for also exploring Random 3-QCOVER is an analogy to SAT. As explained above, QCOVER is a generalisation of SAT, where clauses that do not mention a variable correspond to evidence cubes that have the whole variable domain as the respective dimension. From this perspective, Random QCOVER corresponds to the constant-probability generation model for SAT [4; 9], while Random 3-QCOVER corresponds to the fixed clause-length generation model for 3SAT [9]. Typical phase transitions behaviour has only been reported for the fixed clause-length model. Similarly, we found typical phase transition behaviour only in Random 3-QCOVER.

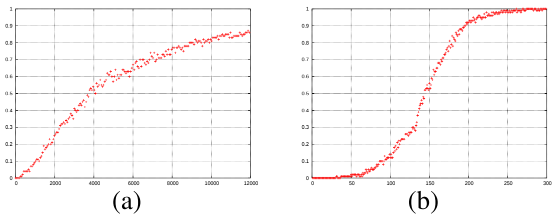


Figure 1: Typical plot of the proportion of unsatisfiable instances against k for (a) Random QCOVER and (b) Random 3-QCOVER.

For both distributions, we investigated the location of the phase transition regions. We tried different values for n and m. For each combination of n and m, we let k vary from very small to very high values (the border values and k increment were pre-determined by some limited experiments). For each k we ran 100 to 1000 instances, and we plotted the proportion of unsatisfiable instances against k. With*in* each of the random distributions, we obtained very similar pictures across all n, m combinations. In contrast, the results compared between the two random distributions are very different. Example plots (for $n = 10$ and $m = 5$) are given in Figure 1.

It is also interesting to observe how many evidence cubes are needed to obtain an equal proportion of satisfiable and unsatisfiable instances. We call this number of evidence cubes the "50% point". The location of the point depends on n, m, and the distribution. Figure 2 plots the point location against n and m for each distribution.

For $m > 10$, determining the location of the 50% point for higher values of n than shown in Figure 2 was prohibitively costly. Two interesting observations can be made. **(1) growth of the 50% point in** n: in Random QCOVER, exponential for higher values of m; in Random 3-QCOVER, around linear for all values of m. **(2) growth of the 50% point in** m: in Random QCOVER, around linear for all values of n; in Random 3-QCOVER, presumably exponential for higher values of n.

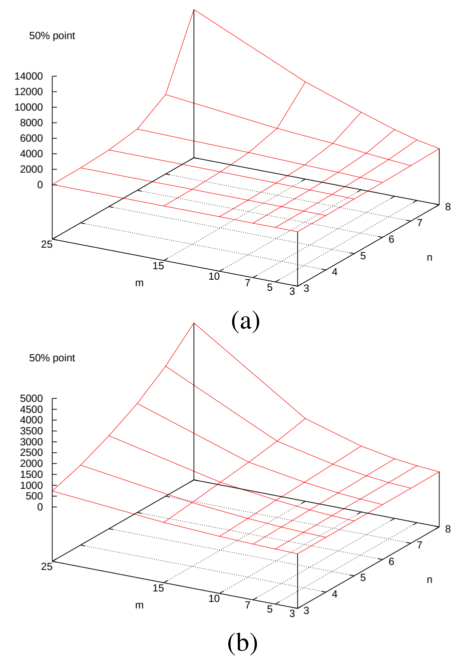


Figure 2: Location of 50% point plotted against n and m in (a) Random QCOVER and (b) Random 3-QCOVER. Note that the z scales are different.

Intuitively, (1) is because in Random 3-QCOVER all but 3 dimensions per evidence cube are fixed anyway while in Random QCOVER all dimensions are generated randomly. An explanation for (2) is that in Random 3-QCOVER the query cube – the value space to be covered – is always the entire value space while in Random QCOVER the query cube – like the evidence cubes – can be a small subset of the available space given by m.

4 Cube Elimination

We now introduce an algorithm, *cube elimination*, that solves QCOVER by constructing a representation of the non-covered part of the query cube. The representation is a set of hypercubes, whose union contains exactly those points in the query cube that are not contained in any of the evidence cubes. See Figure 3 for a pseudo-code description of the algorithm.

The overall cube elimination algorithm, depicted in the upper half of Figure 3, is very simple: the result set $\mathcal{Q}$ is initialised to contain only the query cube; then iteratively all evidence cubes E are eliminated by replacing each cube Q' in $\mathcal{Q}$ with a set of hypercubes covering exactly $Q' \setminus E$. If $\mathcal{Q}$ becomes empty after eliminating one evidence cube (which can happen because if $Q' \subseteq E$ the replacement set is empty, see below), the instance is proved unsatisfiable. Otherwise, upon termination $\bigcup_{Q' \in \mathcal{Q}} Q' = Q \setminus \bigcup_{E \in \mathcal{E}} E \neq \emptyset$ holds. The notations in the upper half of Figure 3 should be self-explanatory. The sub procedure depicted in the lower half of Figure 3 is a little bit more tricky. The procedure, called *minimal cover*,

```
procedure cube elimination(Q, ℰ),
        returns an answer to satisfiability, and
        a set of hypercubes covering exactly Q \ ⋃_{E∈ℰ} E
𝒬 := {Q}
for all E ∈ ℰ do
    𝒬' := ∅
    for all Q' ∈ 𝒬 do
        𝒬' := 𝒬' ∪ minimal cover(Q' \ E)
    endfor
    𝒬 := 𝒬'
    if 𝒬 = ∅ then answer "unsatisfiable", return ∅ endif
endfor
answer "satisfiable", return 𝒬

procedure minimal cover(Q \ E),
        returns a set of hypercubes covering
        exactly the points in Q \ E
if Q ∩ E = ∅ then return {Q} endif
𝒮 := ∅
for d = 1, ..., n do
    if Lower^d(Q \ E) ≠ ∅ then
        𝒮 := 𝒮 ∪ { (Q^1 ∩ E^1, ..., Q^{d-1} ∩ E^{d-1},
                    Lower^d(Q \ E), Q^{d+1}, ..., Q^n)}
    endif
    if Upper^d(Q \ E) ≠ ∅ then
        𝒮 := 𝒮 ∪ { (Q^1 ∩ E^1, ..., Q^{d-1} ∩ E^{d-1},
                    Upper^d(Q \ E), Q^{d+1}, ..., Q^n)}
    endif
endfor
return 𝒮
```

Figure 3: Cube elimination.

computes a set of hypercubes covering (exactly) $Q \setminus E$ for given Q and E. As we will see, the cardinality of the set is minimal – there is no smaller set of hypercubes that covers precisely $Q \setminus E$ – and the cubes in the set are pairwise disjoint. Let us take a little time to explain the procedure.

First, note that (for notational simplicity) "Q" is a parameter in the context of the minimal cover procedure. (Rather than the original query cube given by the QCOVER instance.) The other notations are the following. By "Q^i" ("E^i") we denote the interval of cube Q (E) in dimension i. By "$Q^i \cap E^i$" we denote the interval constraint that describes the conjunction of the constraints Q^i and E^i. (That is, the stronger ones of the respective lower and upper bounds are taken.) By $Lower^d(Q \setminus E)$ ($Upper^d(Q \setminus E)$) we denote the interval of all points in Q^d that lie to the left of (to the right of) E^d. More formally, say $l(Q^d)\, OP\text{-}l(Q^d)\, x^d$ is the lower bound of Q^d, and $l(E^d)\, OP\text{-}l(E^d)\, x^d$ is the lower bound of E^d. If the former bound is stronger than the latter bound, or if both bounds are the same, then $Lower^d(Q \setminus E)$ is empty. Otherwise, $Lower^d(Q \setminus E)$ is the non-empty interval $l(Q^d)\, OP\text{-}l(Q^d)\, x^d\, OP\text{-}u\, l(E^d)$, where $OP\text{-}u =$ "$<$" if $OP\text{-}l(E^d) =$ "$\leq$", and vice versa. The intervals $Upper^d(Q \setminus E)$ are computed in a similar fashion.

The minimal cover procedure first checks if the intersection of Q and E is empty (by a single sweep over the respective intervals in all dimensions). If so, $\{Q\}$ is returned as the result set. Otherwise, the result set $\mathcal{S}$ is initialised to the empty set. Then, in each dimension d, if $Lower^d(Q \setminus E)$ ($Upper^d(Q \setminus E)$) is non-empty then a certain cube, as specified in Figure 3, is included into $\mathcal{S}$. The properties of the procedure are the following.

1. **The procedure is correct, i.e., upon termination we have $\bigcup_{Q' \in \mathcal{S}} Q' = Q \setminus E$.** $\bigcup_{Q' \in \mathcal{S}} Q' \subseteq Q \setminus E$ is obvious. $\bigcup_{Q' \in \mathcal{S}} Q' \supseteq Q \setminus E$ holds because, if $(v^1, \dots, v^n)$ is a point in $Q \setminus E$ and d is the first (lowest) dimension where $v^d \notin E$, then one of the cubes added to $\mathcal{S}$ for dimension d contains $(v^1, \dots, v^n)$. (Note that, if $Q \subseteq E$, for all d the intervals $Lower^d(Q \setminus E)$ and $Upper^d(Q \setminus E)$ are empty, so the returned set $\mathcal{S}$ is empty.)
2. **The cubes in $\mathcal{S}$ are pairwise disjoint.** If two cubes Q' and Q'' are added into $\mathcal{S}$ for the same dimension d, they are disjoint because $Lower^d(Q \setminus E)$ and $Upper^d(Q \setminus E)$ are disjoint. If Q' is added for dimension d, and Q'' is added for dimension d', with $d < d'$, then the interval of Q' in dimension d has an empty intersection with E^d, while the interval of Q'' in dimension d is contained in E^d.
3. **The cardinality of $\mathcal{S}$ is minimal, i.e., there is no set of hypercubes that covers exactly $Q \setminus E$ and that contains less cubes.** If $Q \cap E = \emptyset$, there is nothing to show. Else, consider the cubes C of the form $(Q^1 \cap E^1, \dots, Q^{d-1} \cap E^{d-1}, Lower[Upper]^d(Q \setminus E), Q^{d+1} \cap E^{d+1}, \dots, Q^n \cap E^n)$ where $Lower[Upper]^d(Q \setminus E)$ is non-empty. The number of these cubes C is obviously equal to the number of cubes in $\mathcal{S}$. Minimality follows because if any hypercube H contains two points p_1 and p_2 out of different cubes C_1 and C_2 as above, then the intersection of H with $Q \cap E$ is non-empty (and thus all cubes C must be covered with different hypercubes H). If C_1 and C_2 differ only in dimension d, then the respective intervals are $Lower^d(Q \setminus E)$ and $Upper^d(Q \setminus E)$, and H^d must contain all of $Q^d \cap E^d$. If C_1 and C_2 differ in dimensions d_1 and d_2, then H^{d_1} must contain $p_2^{d_1} \in Q^{d_1} \cap E^{d_1}$, and H^{d_2} must contain $p_1^{d_2} \in Q^{d_2} \cap E^{d_2}$ (or vice versa).

Property 2 is beneficial for cube elimination because one wants to keep the cubes in the set $\mathcal{Q}$ as small as possible, increasing the chance that the evidence cubes ahead will have an empty intersection with them. Property 3 is beneficial because one wants to keep the set $\mathcal{Q}$ as small as possible. Still the number of hypercubes in $\mathcal{Q}$ can be exponential.[4] We remark that the *order* in which the evidence cubes are eliminated can make an exponential performance difference. Assume, e.g., that a cube E covers all of a set of cubes $E_1, \dots, E_l$. If E is eliminated before $E_1, \dots, E_l$, then at the time they are eliminated all E_i will have an empty intersection with all $Q' \in \mathcal{Q}$, and no new cubes Q' will be generated. If, however, $E_1, \dots, E_l$ are eliminated before E then this can result in exponentially (in l) many new cubes Q' (which will all be removed completely when E is eliminated). In our experiments described in the next section, we consider a simple ordering heuristic for the evidence cubes.

We remark that, when interpreting a propositional CNF formula as a QCOVER instance, cube elimination comes

[4] An obvious upper bound is $(2n)^k$ (where k is, as before, the number of evidence cubes). The worst case for the number of cubes needed to represent $Q' \setminus E$ is $2n$, and the bound assumes that this worst case occurs for all pairs Q' and E during cube elimination. Now, it is easy to see that the worst case can not occur for *all* Q' and E, due to geometrical properties of the hypercubes generated by the minimal cover procedure; deriving tighter upper bounds turned out to be highly non-trivial, however, and is an open problem.

down to an iterative transformation of the CNF into a DNF. The minimal cover operations for $Q' \setminus E$ correspond to multiplying a conjunction with a disjunction, and the cubes in $\mathcal{Q}$ correspond to the conjuncts in the developing DNF (if there are no unit clauses then the DNF is initialised to TRUE). From this perspective, cube elimination generalises the transformation of a CNF to a DNF to the case where the logical atoms are single-variable inequalities instead of boolean literals.

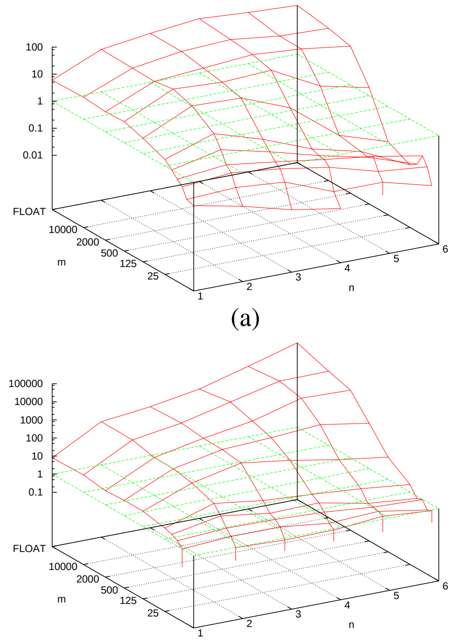
5 Cube Elimination vs. Backtracking

We implemented cube elimination in C. We implemented a backtracking method for QCOVER by plugging a translation to SAT into Chaff [10]. The translation works as follows. In each dimension d, if $x_1^d, \dots, x_l^d$ are the interval endpoints of all cubes in d, in increasing order, then we introduce boolean variables $x_{1;2}^d, x_{2;3}^d, \dots, x_{l-1;l}^d$. Setting $x_{i;i+1}^d$ to 1/to 0 means that the d-coordinate of our selected non-covered point of Q will be/not be between x_i^d and x_{i+1}^d. For each dimension, we introduce a clause $x_{l;l+1}^d \vee \dots \vee x_{u-1;u}^d$ where l and u are the indices of Q's interval endpoints in d. This ensures that our selected (region of) point(s) lies inside Q. For each evidence cube E, we introduce a clause $\neg x^1_{l^1;l^1+1} \vee \dots \vee \neg x^1_{u^1-1;u^1} \vee \dots \vee \neg x^n_{l^n;l^n+1} \vee \dots \vee \neg x^n_{u^n-1;u^n}$ where l^d and u^d are the indices of E's interval endpoints for each d. This ensures that our selected region of points lies outside E. Finally, for each pair $x_{i;i+1}^d$ and $x_{i';i'+1}^d$, $i \neq i'$, we introduce the clause $\neg x_{i;i+1}^d \vee \neg x_{i';i'+1}^d$ to ensure that we select just one atomic region of points.

It is easy to see that the above is a correct translation of QCOVER into SAT. Note also that the $x_{i;i+1}^d$ variables encode, for each dimension d, a single multiple-valued variable in the sense that setting $x_{i;i+1}^d$ to 1 sets, by unit propagation, $x_{i';i'+1}^d$ to 0 for all $i' \neq i$. Doing all these unit propagations can cause a runtime overhead; we account for this by measuring the number of explored search nodes in the different algorithms, rather than their runtime.

Figure 4 shows our results for the Random QCOVER distribution. The experiments were run on a Linux machine running at 3 GHz, with 1 GByte main memory. We let n range between 1 and 6. We let m range from 3 to the case where the interval endpoints are chosen as random floats. For every combination of n and m, we considered the 50% point, c.f. Section 3. For $m \in \{3, 5, 7, 10, 15, 25\}$, we knew this point from the experiments described in Section 3. For larger values of m, such experiments were too costly, so we estimated the location of the 50% points based on some limited tests.[5] At each point, we generated 50 satisfiable and 50 unsatisfiable instances. We take the search space size of our backtracking method to be the number of search decisions made by Chaff – the number of times the variable selection heuristic picked a variable and set it to a value (note that this does not count variable settings that resulted from unit propagation). The search space size of cube elimination is taken to be the total number of cubes generated by the algorithm (the cubes inserted into the set $\mathcal{Q}$, see Figure 3). Figure 4 shows the (averaged) quotient of the backtracking search space size divided by the cube elimination search space size. Clearly, cube elimination becomes superior as the value of m increases. This is particularly true in the unsatisfiable instances where the cube elimination search space is, in the largest instances, around 5 orders of magnitude smaller than that of backtracking.[6] To provide the reader with some absolute values, for $n = 6$ with random floats, on average (over all instances) Chaff made 2615804 search decisions while cube elimination generated 31109 hypercubes. We also measured runtimes, and found similar behaviour; where cube elimination's search space was c times smaller (than that of Chaff), its runtime was between c and $10c$ times smaller. Note that the QCOVER instances are most similar to propositional CNF formulas for small values of m. So Figure 4 shows quite nicely that backtracking is superior when the instances are similar to propositional CNFs, but that cube elimination (generalised DNF transformation) becomes more efficient as one moves into the more general QCOVER region.

For the results above, we used a simple ordering heuristic in cube elimination: the evidence cubes were eliminated in



(a)

(b)

Figure 4: Search space size quotient Chaff vs. cube elimination in Random QCOVER, averaged over (a) all instances, (b) only unsatisfiable instances. The z-axis is log-scaled, the plain $z = 1$ is included for orientation.

[5] As Figure 2 (a) shows, the location of the 50% point does not grow much in m for Random QCOVER and $n \leq 6$, so estimating that location is easy.

[6] To understand the performance difference of cube elimination regarding satisfiable and unsatisfiable instances, note that cube elimination might stop early (without eliminating all evidence cubes) in unsatisfiable cases, namely as soon as $\mathcal{Q}$ has become empty; in satisfiable cases this doesn't happen.

decreasing order of the volume of their intersection with Q. Sorting the evidence cubes never took more than 0.01 seconds in our experiments. The ordering heuristic had a quite significant impact on performance. Cube elimination with the heuristic outperformed (both in terms of search space size and runtime) cube elimination without the heuristic by 1 order of magnitude in the smaller instances, and by up to 3 orders of magnitude in the unsatisfiable cases of the largest instances.

In the Random 3-QCOVER distribution, we found backtracking to be generally superior to cube elimination. For lack of space, we do not include a picture. For each $m \in \{3, 5, 7, 10, 15, 25\}$ and $n \in \{1, \dots, 7\}$, we generated 50 satisfiable and 50 unsatisfiable instances at the 50% point. We averaged runtime and search space size over these 100 instances. Chaff becomes superior with growing n, largely independently of the value of m. With $n = 7$ Chaff is better by about a factor of 100 (both in terms of search space size and in terms of runtime). Interestingly, when averaging over only the satisfiable instances, Chaff's advantage factor grows in m (from 30 for $n = 7, m = 3$ to 350 for $n = 7, m = 25$); when averaging over only the unsatisfiable instances, Chaff's advantage decreases in m (from 30000 for $n = 7, m = 3$ to 50 for $n = 7, m = 25$). As Figure 2 indicates, the 50% point in Random 3-QCOVER grows a lot in m already for small values of n. Indeed, for higher values of m than 25 it becomes very hard to generate unsatisfiable instances (because the necessary number of evidence cubes is extremely large). For $m = 100$, the largest n where we could generate any unsatisfiable instances was $n = 5$. With $k = 30000$ evidence cubes, we generated 50 satisfiable as well as 50 unsatisfiable instances. Averaged over all these instances, Chaff performs better than cube elimination by a factor 30; averaged over the satisfiable instances, Chaff is better by a factor 100, in the unsatisfiable instances the factor is 15.

6 Discussion

We have considered a problem asking for the satisfiability of a CNF formula where the logical atoms are inequalities over single (numeric) variables. We determined, in two random distributions, the phase transition behaviour in dependency of the problem parameters, and introduced a solution method, called cube elimination, that generalises DNF transformation to our case. Our experiments show that, in the Random QCOVER distribution, cube elimination is superior to Chaff when the variable domains underlying the instances are large, in particular in unsatisfiable cases. In the Random 3-QCOVER distribution, we could not run meaningful experiments with large variable domains; for smaller variable domains, Chaff is clearly superior. It should be noted that, in difference to backtracking methods such as Chaff, cube elimination produces a representation of *all* satisfying assignments in the satisfiable cases.

There is a lot of work on covering relations between geometrical objects that, at first sight, seem related to our QCOVER problem. E.g., covering a polygon with rectangles [1] is relevant for the design of logical circuits. However, in such geometric contexts the dimensions usually correspond to the few dimensions in the real world, and one tries to solve covering problems concerning complex objects. In contrast, in QCOVER we try to solve a problem concerning simple objects with arbitrarily many dimensions.

In spirit, cube elimination is somewhat similar to the "bucket elimination" framework defined by Rina Dechter [3] (indeed, that's where the algorithm got its name from). The key difference is that, while bucket elimination eliminates the problem *variables* one by one, we eliminate the problem *constraints* one by one. It seems interesting to explore if this approach, eliminating constraints instead of variables, can be useful in other types of constraint problems as well.

We also consider it interesting that the cube elimination algorithm is easiest to understand, and was originally motivated by, viewing the satisfiability problem we consider as a *geometrical* problem. This opens up the question if other algorithmic approaches used in computational geometry can lead to useful solution methods for QCOVER, and if geometrical interpretations of other satisfiability problems can lead to interesting new methods for these problems.

References

[1] J.C. Culberson and R.A. Reckhow, 'Covering polygons is hard', *Journal of Algorithms*, 2–44, (1994).

[2] M. Davis and H. Putnam, 'A computing procedure for quantification theory', *Journal of the Association for Computing Machinery*, **7**(3), (1960).

[3] Rina Dechter, 'Bucket elimination: A unifying framework for reasoning', *Artificial Intelligence*, **113**, 41–85, (1999).

[4] Allen Goldberg, 'Average case complexity of the satisfiability problem', in *Proc. 4th Workshop on Automated Deduction*, ed., W. S. Joyner, pp. 1–6, (1979).

[5] E. Goldberg and Y. Novikov, 'Berkmin: A fast and robust SAT solver', in *Proceedings DATE-2002, Paris*, pp. 142–149, (2002).

[6] Jana Koehler, 'Planning under resource constraints', in *Proc. ECAI-98*, pp. 489–493, Brighton, UK, (August 1998). Wiley.

[7] Alan K. Mackworth, 'Constraint satifaction', in *Encyclopedia of Artificial Intelligence*, ed., S. C. Shapiro, 205–211, Wiley, Chichester, England, (1987).

[8] Joao Marques-Silva and Karem Sakallah, 'GRASP: A search algorithm for propositional satisfiability', *IEEE Transactions on Computers*, **48**(5), 506–521, (May 1999).

[9] David Mitchell, Bart Selman, and Hector J. Levesque, 'Hard and easy distributions of SAT problems', in *Proc. AAAI-92*, pp. 459–465, San Jose, CA, (July 1992). MIT Press.

[10] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, 'Chaff: Engineering an efficient SAT solver', in *Proceedings of the 39th Design Automation Conference, Las Vegas, June 2001*, (2001).

[11] Patrick Prosser, 'Hybrid algorithms for the constraint satisfaction problem', *Computational Intelligence*, **9**(3), 268–299, (1993).