ECAI 2014
T. Schaub et al. (Eds.)
© 2014 The Authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-419-0-843

Set-theoretic duality: A fundamental feature of combinatorial optimisation

John Slaney[1]

Abstract. The duality between conflicts and diagnoses in the field of diagnosis, or between plans and landmarks in the field of planning, or between unsatisfiable cores and minimal co-satisfiable sets in SAT or CSP solving, has been known for many years. Recent work in these communities (Davies and Bacchus, CP 2011, Bonet and Helmert, ECAI 2010, Haslum *et al.*, ICAPS 2012, Stern *et al.*, AAAI 2012) has brought it to the fore as a topic of current interest. The present paper lays out the set-theoretic basis of the concept, and introduces a generic implementation of an algorithm based on it. This algorithm provides a method for converting decision procedures into optimisation ones across a wide range of applications without the need to rewrite the decision procedure implementations. Initial experimental validation shows good performance on a number of benchmark problems from AI planning.

1 Introduction

The concept of duality within fields of sets is a central one for the science of combinatorial optimisation. Its basic logical properties have been observed many times in the literature, most famously by Reiter in his seminal work on diagnosis [13] but the fact that they apply uniformly to a much larger class of problems than just those arising in diagnosis has not been sufficiently appreciated. In a loose way, it appears to be something everyone knows, and yet every time it is applied to another problem it is a surprise, and every time the key underlying theorem gets proved once more. The present paper lays out the concept clearly, not as a fact about diagnosis, nor about SAT, nor about any particular kind of reasoning, but as a series of rather simple observations about fields of sets. Reiter's result that diagnoses can be found by covering minimal conflicts is a trivial corollary, as is the theorem of Stern *et al.* [17] that this relationship is reversible.

By a *hitting set* for a family of sets we mean a set whose intersection with every set in the family is non-null. Since the relationship between dual families of sets is generic, so is the method of solving problems by generating hitting sets for their duals. Here we report a problem-neutral implementation aimed at finding cost-minimal solutions, but adaptable to enumerate all inclusion-minimal solutions, which immediately yields solvers in areas as diverse as classical planning and MAX-SAT.

2 Logical preliminaries

Let Σ be a set.[2] Let θ be a set of subsets of Σ: the ones with some property P. We say that P (or equivalently θ) is *monotone* over Σ if it is up-closed under set inclusion: i.e. for subsets s and t of Σ, if $s \subseteq t$ and $s \in \theta$ then $t \in \theta$. Many combinatorial optimization problems can be expressed rather naturally in terms of finding a minimum-cost member of some monotone family θ of subsets of some suitable Σ, where cost is often additive, the cost of a set being the sum of costs of its members. In the travelling salesman problem, for instance, Σ is the set of arcs of a graph, θ is the family of solutions, cast as subsets containing a Hamiltonian circuit, and an optimal solution has a minimal sum of arc weights. In consistency-based diagnosis, it is common to define a [minimal] diagnosis as a [minimal] set of components whose removal from the set of those stated to be functioning normally suffices to render a system specification consistent with some observations. Here Σ is the set of components and θ the set of (possibly non-minimal) diagnoses. Again, the weighted MAX-CSP problem is to find a satisfiable subset of a set Σ of constraints minimising the sum of the penalties for the violated (i.e. unsatisfied) constraints. This is trivially equivalent to finding a minimal-cost co-satisfiable subset of the constraints. Hence any problem expressible as a weighted MAX-CSP is of the kind considered here, making the class of problems we consider rather general.

By the *dual* of a set $\theta \subseteq 2^{\Sigma}$ relative to a universe Σ we mean the set $\theta^* = \{s \subseteq \Sigma : \overline{s} \notin \theta\}$.[3] Thus if θ is the set of diagnoses (sets of components which may all be malfunctioning), then θ^* is the set of *conflicts*, or sets of components which cannot all be functioning correctly. Again, if θ is the family of co-satisfiable subsets of some constraints (i.e. the subsets whose complements are satisfiable), then θ^* is the family of unsatisfiable subsets of the constraints.

This duality operator has some nice properties. It is easy to show:

1. $\theta^{**} = \theta$
2. $(\theta \cap \pi)^* = \theta^* \cup \pi^*$
3. $(\theta \cup \pi)^* = \theta^* \cap \pi^*$
4. If θ is monotone (respectively, antitone) then so is θ^*.
5. $|\theta| + |\theta^*| = 2^{|\Sigma|}$

To prove (1), note that by definition

$$\begin{array}{rcl} \theta^{**} & = & \{s : \overline{s} \notin \theta^*\} \\ & = & \{s : \overline{s} \notin \{t : \overline{t} \notin \theta\}\} \\ & = & \{s : \overline{\overline{s}} \in \theta\} \\ & = & \theta \end{array}$$

For (2)

$$(\theta \cap \pi)^* \quad = \quad \{s : \overline{s} \notin \theta \cap \pi\}$$

[1] Australian National University, email: john.slaney@anu.edu.au
[2] While the definitions in this section apply to sets of any cardinality, for the rest of the present paper we assume that Σ is finite.

[3] By the complement $\overline{s}$ of a set s we mean, as should be obvious from the context, $\Sigma \setminus s$. Similarly, for a family θ of sets, $\overline{\theta}$ is $2^{\Sigma} \setminus \theta$.

$$
\begin{aligned}
&= \{s : \overline{s} \notin \theta \vee \overline{s} \notin \pi\} \\
&= \{S : s \in \theta^* \vee s \in \pi^*\} \\
&= \theta^* \cup \pi^*
\end{aligned}
$$

The argument for (3) is similar. In algebraic vocabulary, (1), (2) and (3) together make * an involutive dual automorphism on the lattice of subsets of Σ. That is mathematically neat, but properties (4) and (5) start to be more interesting computationally.

To prove (4) suppose θ is monotone, that $s \in \theta^*$ and $s \subseteq t$. Then $\overline{t} \subseteq \overline{s}$ and $\overline{s} \notin \theta$, so $\overline{t} \notin \theta$, which is to say $t \in \theta^*$. The argument that if θ is antitone then so is θ^* is similar. In the light of (4), monotone combinatorial problems come in dual pairs: if θ is the set of solutions to one monotone problem then θ^* is the set of solutions to another.

For (5), observe that the members of θ^* are just the complements of the non-members of θ, so obviously $|\theta^*| = |\overline{\theta}|$; but the disjoint union of θ and $\overline{\theta}$ is just 2^Σ, so $|\theta| + |\theta^*| = 2^{|\Sigma|}$.

Hence if θ is small—if its members are rare, as is the case with solutions to a problem which is close to critically constrained—then θ^* is large, and *vice versa*. This is good news and bad news. The bad: where there are enough elements in Σ to make a combinatorial optimisation problem interesting (say, more than about 50 of them), it is physically impossible to enumerate both θ and θ^* explicitly. The good: if one of the two problems is tightly constrained, then the other is radically underconstrained, and *vice versa*, so depending on the case there may be attacks which work better on one type of problem than on the other and which may therefore be employed against both—if only we have a way of relating them computationally to each other. Moreover, logic may come to the rescue by providing a concise statement of one of the two problems which, by duality, may allow access to the other.

Fortunately, there *is* a computationally useful relationship between θ and θ^*. The following hold quite generally:

6. Let s be a subset of Σ such that for every t in θ, $s \cap t \neq \emptyset$. Then $s \in \theta^*$.
7. Let θ be monotone. Then for any $s \in \theta$ and $t \in \theta^*$, $s \cap t \neq \emptyset$.

Proofs are again easy. Suppose the conditions of (6) hold. Clearly s has an empty intersection with $\overline{s}$, so $\overline{s} \notin \theta$, which is to say $s \in \theta^*$. Now for (7), suppose θ and θ^* are monotone and $s \in \theta$. If $s \cap t = \emptyset$ then $s \subseteq \overline{t}$, so by monotonicity $\overline{t} \in \theta$, meaning $t \notin \theta^*$. The upshot of (6) and (7) is that where θ is monotone, θ^* consists of exactly the hitting sets (drawn from Σ) for θ. Trivially, in that case, θ is also the set of hitting sets for θ^*. Moreover, θ is the set of hitting sets for the set of its own hitting sets.

Finding a minimum (optimal) member (resp. an inclusion-minimal member, an approximately optimal member, all minimal members, etc.) of θ is therefore the same thing as finding a minimum (resp. minimal, approximately minimal, etc.) hitting set for θ^*. Algorithms for solving hitting set problems are well developed, offering a general technique with wide applicability in optimization. In principle at least, any combinatorial optimisation problem which can be cast as finding a cost-minimal member of some monotone family θ of subsets of a carrier set Σ can be approached dually via the hitting set minimisation problem for θ^*. Similar remarks apply to the problem of enumerating the inclusion-minimal members of θ, as is common in diagnosis for example. We now outline algorithm templates for generating a single best solution.

3 Finding the best

We begin with the pure optimisation problem: given a specification of a monotone family of sets θ, find and return a single member of θ with minimal cost. This is to be done by determining an optimal hitting set for θ^*.

While it is rarely feasible to generate all of θ^* explicitly, fortunately there are better ways. The property P defining θ may admit of a relatively low-cost decision procedure, and there may be small subsets κ of θ^* such that a minimum hitting set for κ also happens to hit all of θ^*. This gives rise to an algorithm template which yields promising solutions in a range of cases.

In what follows, for any family $\mathcal{F}$ of sets, we write hs($\mathcal{F}$,s) to mean $\forall t \in \mathcal{F}(s \cap t \neq \emptyset)$. The first function required is a generator of optimal hitting sets. Assuming the cost function C somehow defined:

```
MHS ($\kappa$) : set
  Choose $s$ such that
    hs($\kappa$, $s$) $\wedge$
    $\forall t$ (hs($\kappa$, $t$) $\Rightarrow C(s) \leq C(t)$)
```

Any optimal hitting set generator may be used for this purpose. A standard branch and bound algorithm performs reasonably well, but it is also easy to encode the problem as a MIP, allowing any off-the-shelf MIP solver to be used instead. If optimality is not required, it is also possible to substitute an incomplete solver based on local search—large neighbourhood search is reported [10] to do well with cognate minimum set cover problems. We have not yet experimented with this, but it would obviously be straightforward to do so.

The technique for building κ will be to start with the empty set and iterate the following process: find a hitting set h for the currrent κ which is as large as possible while still remaining a non-member of θ—do this by starting from a small hitting set s and adding as many elements to it as possible; then $\overline{h}$ is a small member of θ^* not hit by h, so add it to κ. Continue in this fashion until a minimal-cost hitting set for κ is found which is a member of θ: this is the optimal solution.

Where θ is a monotone family of sets and s is any non-member of θ, we need to choose a set in θ^* disjoint from s. In practice it pays to choose sets of small cardinality, but in principle any choice (even, in the worst case, $\overline{s}$ itself) will suffice:

```
SEL ($theta$, $s$) : set
  Choose $t$ such that
    $t \in \theta^*$ $\wedge$
    $s \cap t = \emptyset$
  or fail
```

For any $t \subseteq \Sigma$, deciding whether t is in θ^* amounts to deciding whether $\overline{t}$ is in θ, so the decision procedure required to show $s \notin \theta$ can also be used to test any candidate t for membership of θ^*.

Now obviously if the cost function C is monotone-increasing with set inclusion, and $\kappa \subseteq \theta^*$, a minimum-cost hitting set for κ which also happens to be in θ is a minimum-cost hitting set for $theta^*$. So:

```
OPT ($\theta$, $\kappa$) : set
  Let $h \leftarrow$ MHS($\kappa$);
  If $h \in \theta$ then
    return $h$
  else
    return OPT($\theta$, $\kappa \cup \{$SEL($theta$, $h$)$\}$)
```

The required optimal member of θ is then OPT(θ, $\emptyset$).

The function OPT does not need to be much more complicated than the few lines of pseudo-code above. MHS is responsible for all of the optimisation. As noted, it can use any off-the-shelf solver for the minimum hitting set problem. For experiments, we used our own implementation of the algorithm presented by De Kleer [5] which is described below. The MHS function is generic—invariant over different problems. The function SEL and the decision procedure for θ, on the other hand, are problem-specific.

The algorithm works by progressively building up a subset κ of θ^*, starting with the null set and at each step adding a set not hit by an optimal hitting set (h) for the current κ. This means that θ^* need never be constructed explicitly, for a subset (typically much smaller than θ^*) suffices. Moreover, membership of θ^* is determined using the decision procedure for P, which in many cases is reasonably efficient [11] and again does not require θ^* to be known in detail.

In some cases, in fact, P is testable in polynomial time,[4] and even where it is not, decision is typically much easier than (provable) optimisation. This makes it possible for a version $\mathsf{SEL}_{\mathsf{min}}$ of SEL to return an inclusion-minimal member of θ^* at each iteration of OPT:

```
SEL_min ($\theta$, $s$) : set
  If $s \in \theta$ then
    Fail
  else
    let $A \leftarrow s$;
    For each $x \in \Sigma$ do
      If $A \cup \{x\} \notin \theta$ then
        $A \leftarrow A \cup \{x\}$
    Return $\overline{A}$
```

We check that s is not in θ, so A initially lacks P, and this property is preserved at every subsequent step, so $\overline{A}$ is always in θ^*, and since every possible one-element extension of A has been tried and rejected, at the end of the iteration through Σ it is also inclusion-minimal. Moreover, since $s \subseteq A$, obviously $\overline{A}$ and s are disjoint. Since true minimality is not actually required for the purposes of OPT, it is possible to use a sound approximation to the decision procedure for P inside $\mathsf{SEL}_{\mathsf{min}}$: false positives will only make the returned set $\overline{A}$ a little bigger than necessary, which is not fatal. This enables an approximate version of $\mathsf{SEL}_{\mathsf{min}}$ to run fast even where the decision procedure is not polytime.

On most calls to SEL, s is not in fact in θ. It is used only as the seed for another member of θ^* with which to extend κ. Hence, s does not actually need to be minimal: any hitting set for the current κ will do, though one of small cardinality is likely to be better than a large one because there is more freedom to extend it to at least one large non-member of θ. It therefore pays to use an approximately minimal hitting set which can be generated fast rather than calling MHS on every iteration. Only when s is in θ need MHS be called to replace it with a minimum-cost s' for the search to continue. This leads to a more elaborate version of OPT, making use of a function $\mathsf{HS}_{\mathsf{approx}}$ which returns a small, but not necessarily minimal, hitting set:

```
OPT^A ($\theta$, $\kappa$) : set
  Let $h \leftarrow \mathsf{HS}_{\mathsf{approx}}(\kappa)$;
  If $h \in \theta$ then
    Let $h \leftarrow$ MHS($\kappa$);
    If $h \in \theta$ then
      return $h$;
  return OPT^A($\theta$, $\kappa \cup \{\mathsf{SEL}(\theta, h)\}$)
```

[4] Blocks World planning and delete-free planning, described below, are examples of NP-hard optimisation problems with linear time decision procedures.

```
MCHS ($\mathcal{F}$, $H$, $b$) : set
  Simplify($\mathcal{F}$, $H$);
  If $C(H) + C_{\text{est}}(H, \mathcal{F}) \geq b$ then
    return FAIL
  else if $\mathcal{F} = \emptyset$ then
    return $H$
  else
    Choose $a \in \bigcup \mathcal{F}$;
    Let $\mathcal{F}' \leftarrow \{s \setminus \{a\} : s \in \mathcal{F}\}$;
    Let $H_1 \leftarrow$ MCHS($\mathcal{F}$, $H \cup \{a\}$, $b$);
    If $H_1$ = FAIL then
      Let $H_2 \leftarrow$ MCHS($\mathcal{F}'$, $H$, $b$)
    else
      Let $H_2 \leftarrow$ MCHS($\mathcal{F}'$, $H$, $C(H_1)$)
    If $H_2$ = FAIL then
      return $H_1$
    else
      return $H_2$
```

Figure 1. Recursive minimum cost hitting set algorithm

Since Σ is finite, termination is guaranteed provided SEL and the decision procedure for P terminate: MHS terminates (see below) and every call from OPT or OPT^A to SEL adds to κ a set which cannot have been already there as it is disjoint from h. In the limit, $\kappa = \theta^*$ and so h is eventually returned.

The problems addressed are generally NP-hard, so exponential running time is expected. In the worst case, it may be double-exponential in $|\Sigma|$, since the naïve bound on the number of calls to OPT is given by $|\theta^*|$ which may be close to $2^{|\Sigma|}$, and each such call may involve solving a co-NP-hard subproblem such as showing that a graph has no Hamiltonian circuit. The experimental results below, however suggest that this worst case is not an impediment to many applications.

Implementation

For the remainder of this section, we assume that costs are additive, and write $C(s)$ for the cost of s (the sum of the costs of its members). More sophisticated notions of cost could be handled, but this would require modification of the algorithm to compute hitting sets, which we have not considered at this point.

The heart of the algorithm (Figure 1) is the function returning minimum hitting sets. The method used here is rather simple, and not at all original with this paper, of course [5]. It consists of a search by DFBB, backtracking on the choice of whether to include an element in the hitting set or exclude it from the problem. At each node in the search tree the problem of generating a hitting set H for $\mathcal{F}$ is "simplified" by iterating to a fixpoint:

1. remove from $\mathcal{F}$ any set which is hit by H;
2. remove from $\mathcal{F}$ any subsumed set (i.e. any proper superset[5] of another in $\mathcal{F}$);
3. remove from all sets in $\mathcal{F}$ any element x dominated by another y (which is of no greater cost and in at least all the same members of $\mathcal{F}$);
4. add the member of any singleton in $\mathcal{F}$ to H.

[5] Since $\mathcal{F}$ is a set, it cannot contain duplicates, so all subsumption is proper.

Figure 1 shows the recursive search function. $\mathsf{MHS}(\mathcal{F})$ is just $\mathsf{MCHS}(\mathcal{F},\emptyset,\infty)$. The family $\mathcal{F}$ (which will be κ in practice) is given explicitly as a list of sets, in contrast to θ and θ^*, which are defined implicitly by some sort of decision procedure. Simplifying the problem at a node is polynomial-time in the number of sets and their cardinality. Note that $C_{\mathrm{est}}(s,\mathcal{F})$ is the estimated cost of extending s to become a hitting set for $\mathcal{F}$, according to some admissible heuristic. Termination is clear, since the search tree branches only two ways at each node, and the length of each branch is bounded by the cardinality of $\bigcup\mathcal{F}$ because the element on which the search splits is removed from the two resulting sub-problems. It is well known that the minimum hitting set problem is NP-equivalent. De Kleer [5] reports that the algorithm outlined here performs competitively with the state of the art across a range of hitting set problems, including both artificial ones and those drawn from problems in diagnosis of static systems. We implemented it straightforwardly in C with a simple API allowing it to be incorporated easily into problem-specific programs.

Evaluation

As this paper has been at pains to emphasise, the dual hitting set method is not new. Variants of it have been reported by many authors, and the fact that it is somehow generic has been pointed out, notably by Moreno-Centeno and Karp [11], but its set-theoretic basis and the facts proved in the first section of the present paper have, to the best of our knowledge, never been fully laid out. In 2011–12, three papers appeared, independently reporting excellent results for dual hitting set algorithms for MAX-SAT [4], static system diagnosis [17] and delete-free classical planning [8]. Other examples go back at least to the optimal blocks-world solver [15] which we reported in 1996.

Our generic tool for converting decision procedures into optimisers has yet to be fully validated experimentally. As an initial step, it has been compared with our own previous implementations of optimisation algorithms which use hitting set minimisation. In particular, in application to the AI planning problems noted above, the generic solver performs at a level competitive with the existing state of the art, without requiring problem-specific implementation.

Optimal delete-free planning

For the purposes of optimal planning, using standard algorithms such as A*, it is critically important to make use of good admissible heuristics. Most admissible heuristics used in practice, such as $h^{\max}$ and LM-cut [9], are based on the *delete-free relaxation* of planning problems, obtained by ignoring the delete lists of actions. They deliver lower bounds on h^+, defined as the cost of the optimal delete-free plan reaching the goal. In general, computing h^+ is hard: the problem is NP-equivalent [3] and not approximable [2]. Recently [8] we presented an effective algorithm for computing h^+ exactly in practical cases, using the dual hitting set method. This is arguably the technique of choice for the problem, as attested by its performance across standard planning benchmarks. It is therefore of interest to compare the generic solver presented in the present paper with the special-purpose one designed for the delete-free planning problem.

Since the relaxed plans do not involve deletions, they may be regarded as sets of actions: such a set is a valid plan if every goal proposition is eventually produced by closing the initial state description under actions in the plan. The dual notion is that of a *disjunctive action landmark*, or a set of actions at least one of which appears in every valid plan. Clearly, given a putative plan, reachability of the goal is decidable in time polynomial in the number of actions and propositions, simply by chaining forward to a fixpoint.

Domain	#	1 sec		30 sec		5 min	
		old	new	old	new	old	new
Airport	50	**26**	25	47	47	49	**50**
Barman	20	5	5	**20**	5	**20**	5
Blocks 3-ops	35	32	**35**	35	35	35	35
Blocks 4-ops	35	35	35	35	35	35	35
Cybersec	30	**8**	7	**27**	20	**28**	20
Depots	22	**12**	10	**17**	12	**18**	13
Driverlog	20	7	**8**	8	**10**	11	11
Freecell	60	0	0	**4**	2	**7**	3
Logistics	28	26	**28**	28	28	28	28
Miconic	150	150	150	150	150	150	150
Non-Mystery	20	**3**	0	4	4	4	4
Openstacks	30	27	27	30	30	30	30
Parc Printer	30	29	29	30	30	30	30
Pathways	30	4	**6**	5	**8**	7	**9**
PegSol	30	30	30	30	30	30	30
Pipes No Tank	50	9	9	**11**	10	**11**	10
Pipes Tankage	50	6	6	9	9	10	10
PSR small	50	**50**	49	50	50	50	50
Satellite	36	5	5	6	6	7	7
Scanalyzer	30	**6**	4	**11**	5	**14**	5
Sokoban	30	30	30	30	30	30	30
Storage	30	17	17	**21**	19	**27**	22
TPP	30	**10**	9	**13**	11	**16**	12
Transport	30	**5**	3	**6**	5	6	6
Trucks	30	12	**18**	19	**30**	21	**30**
Visitall	20	0	**4**	2	**11**	5	**20**
Woodworking	30	10	10	17	**22**	22	**27**

Table 1. Numbers of problems in each domain solved by each solver within a second, within 30 seconds and within 5 minutes.

For the experiment, the two solvers—the old one from 2012 and the new one using the generic method of this paper—share code for reading and preprocessing problems. The test for reachability of goals is also the same, as is the SEL function. The main differences lie in the ways hitting sets are generated, both where optimal ones are sought and where sub-optimal ones are used for quick discovery of new landmarks. The 2012 planning-specific solver generates better sub-optimal hitting sets, as it was tuned in this regard by means of many experiments with planning benchmarks whereas the new generic solver uses a simple general-purpose scheme which has not been tuned at all. However, the new solver is more efficient in the generation of optimal hitting sets. The behaviour of the two systems also diverges on problems with high numbers of zero-cost actions, although of course the values they return for h^+ are the same.

The benchmark problems are those from 27 domains used in the International Planning Competitions (IPC) from 1998 to 2011. These domains all require propositional strips planning, and some have action costs. A time limit of 5 minutes was imposed, rather than the 30 minutes allowed in the competition: this is reasonable, given that we are only solving the delete-free relaxations rather than the problems themselves.

Broadly, the performance with and without special-purpose coding is comparable (see Figures 1 and 2). Note the log scale. The results do vary somewhat between domains, as may be seen in Table 1: the Barman and Scanalyzer problems, for instance, are solved better by the old system, while the Visitall and Trucks domains are easier for the new one. In the main, however, the differences are not great. This

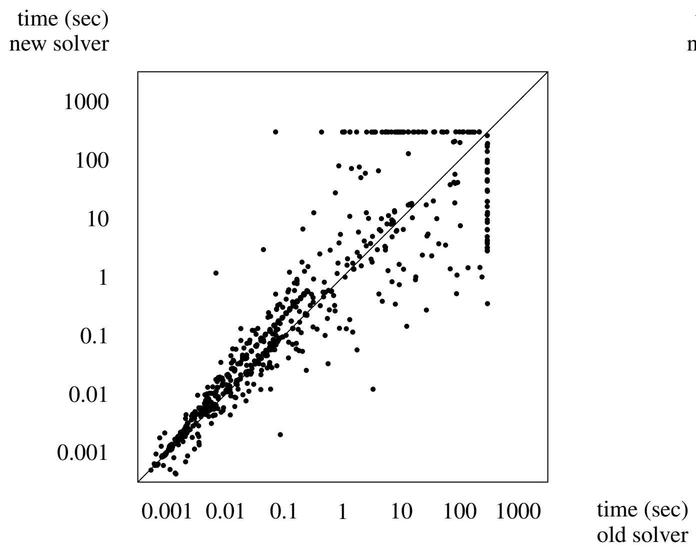

Figure 2. New solver for optimal delete-free planning, based on the generic hitting set generator, versus the existing special-purpose planner. Problems are those in the classical strips domains from the International Planning Competitions. Runtimes are in seconds, with a cutoff at 5 minutes.

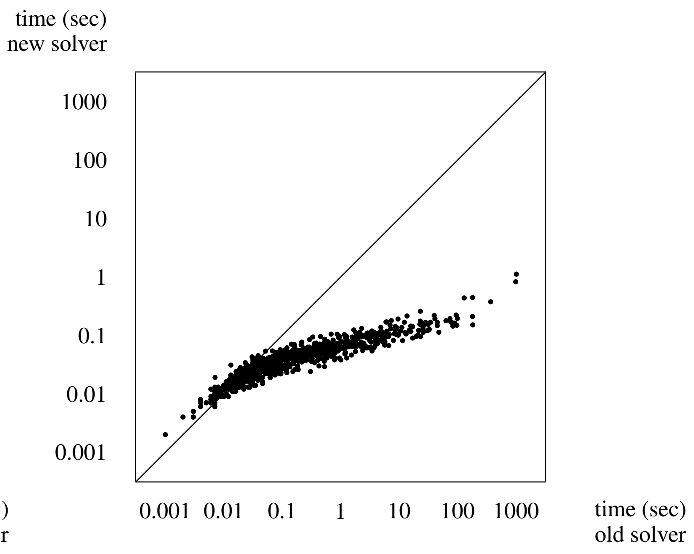

Figure 3. New version of the optimal Blocks World solver, based on the generic hitting set generator, versus the existing special purpose one. Problems are randomly generated in the range 100–180 blocks. Runtimes are in seconds.

is encouraging, as it suggests that there is little to gain by implementing the dual hitting set algorithm specially rather than using the generic version.

A special case: blocks world

The blocks world (BW) is a standard example and testbed for classical planners and planning formalisms. Though not, as naturally expressed, delete-free, it is sufficiently close to the above problems to be solved by similar means. A state of BW consists of a finite set of blocks stacked up into towers which rest on a surface, conventionally called the table, taken to be big enough to hold all the blocks if necessary. The only actions are to move a clear block (i.e. with nothing above it) from its current position to rest on top of another clear block or on the table. A plan is a sequence of such moves. For simplicity, we consider problem instances in which both the initial state and the goal state are fully specified, and identify actions by the block moved and its destination.

A block is said to be *in position* if (a) what it is on now is what it will be on in the goal, and (b) what it is currently on is in position. The table is always in position. Clearly, every "constructive" move which puts a block into position occurs in every successful plan, so its singleton is a landmark. Among the other landmarks are those corresponding to *deadlocks*, where a deadlock is a set of blocks forming a cycle, none of which can move into position until the next block in the cycle has moved [7]. A plan may easily be extracted [15] from any hitting set for the set of deadlocks. Put another way, the set of constructive moves together with a hitting set for the landmarks corresponding to deadlocks hits *all* landmarks, and a minimal such hitting set generates an optimal plan.

Provably near-optimal BW planning can be achieved in linear time [15] so large instances are easy provided optimality is not required. In the optimal planning case, however, problems with as few as 25 or 30 blocks pose difficulties for most planners. The optimal solver 'bwopt'[6] uses a linear-time near-optimal solver to decide θ and a version of MHS to generate plans out of the deadlocks. It returns optimal plans in acceptable time for arbitrary problems of 150 blocks. This performance has remained unchallenged for over 15 years.

A new implementation of the optimal solver, using the generic code for generating minimum hitting sets while retaining the same near-optimal solver for deciding θ, clearly out-performs the existing one, as shown in Figure 3. The main reason appears to be superior propagation in the optimal hitting set generator, though generation of non-optimal hitting sets is also relevant. Since minimum hitting set generation is the only bottleneck in the algorithm, a better implementation improves performance significantly. For this experiment, 10 problem instances of each size (number of blocks) from 100 to 180 were generated using the problem generator 'bwstates' [16] to produce uniformly distributed random problems. On all nontrivial instances (taking more than about 0.1 seconds to solve on the given hardware) the new version of the solver is faster. Note that the scale in Figure 3 is logarithmic, so it is clear from these results that the improvement is exponential. The median runtime for the new solver on random problems of 300 blocks is 1.5 seconds on the given hardware, whereas the old one has a median runtime of around half a minute on 200-block problems and is completely unable to handle 300-block ones.

4 Finding all minimal solutions

Much of the literature on solving problems by generating hitting sets for the duals concerns the challenge of enumerating all minimal (i.e. inclusion-minimal) solutions rather than generating a single cost-minimal solution. In diagnosis, for instance, it is frequently important to find all minimal diagnoses, whereas the notion of the "best" one may be of little significance. The algorithm in Reiter's 1987 paper [13] is such an enumerator of minimal solutions. One example from another field which has influenced the present work is Bailey and

[6] `http://users.cecs.anu.edu.au/ jks/bw.html`

Stuckey's dynamic programming technique [1] for enumerating unsatisfiable cores of problems arising in debugging Haskell programs. Mention should also be made of the work of Eiter and Gottlob [6] on the complexity of hypergraph traversal in the abstract.

It is not difficult to devise a generic algorithm like that of Bailey and Stuckey to enumerate minimal solutions using the function SEL and the iterative construction of κ. We do not detail such an algorithm here, as the focus of the present paper is on optimisation, but are experimenting with it in the context of correctness debugging of constraint models. Again, there seems no reason to expect performance to be significantly worse than that of a problem-specific implementation.

5 Conclusion

Dual hitting set minimisation is not a magic bullet: like most generic techniques, it is widely applicable in principle but for most problems there are domain-specific methods which work better. Most notably, problems calling for optimal permutations of sequences, which are common in scheduling for instance, are not naturally monotone and are not usefully approached through their hitting set duals. Nonetheless, as noted above, there are cases in which hitting set minimisation is the technique of choice, and many more in which it yields tolerable results without requiring much implementation work. It is therefore a worthwhile weapon to have in the optimisation armoury.

Here we have reported problem-neutral implementations of methods for generating an optimal solution and for generating all minimal ones. Preliminary tests comparing the generic implementation against the best problem-specific generator of optimal delete-free plans for a range of planning problems suggest that any loss in performance is slight in comparison with the gain in ease of programming.

Unfinished business includes extending the algorithm templates of this paper to include more techniques for dual reasoning, experimenting with more applications and deepening the underlying mathematical theory. The following are all indicated directions for future work:

- There are several good approaches to generating minimal hitting sets. Our system should make it easy to switch between the existing DFBB solver and others based on MIP or SAT, for instance.
- It is easy to use an approximate solution method such as large neighbourhood search or other kinds of local search to generate suboptimal but "good" hitting sets quickly in cases to which the exact methods will not scale. The quality of solutions obtainable in this way and their possible applications will surely bear investigation.
- Exploiting duality in both directions, to generate simultaneously minimal members of θ as hitting sets for θ^* and minimal members of θ^* as hitting sets for θ [17] is a natural extension of the ideas outlined here. Its correctness is immediate from observations 1 and 4 in the first section of the present paper. It may have applications in cases where neither θ nor θ^* is easy to explore by itself: these may arise in diagnosis or in MAX-SAT, for instance.
- The generic solvers need to be benchmarked against challenging problems from more fields, certainly including model-based diagnosis, to confirm or qualify the result from the experiments in planning, that there is little or no degradation in performance associated with using the generic system as opposed to special purpose ones.
- The mathematics of duality should be further pursued. The abstract algebra of dual automorphisms on distributive lattices has been well studied, and was related as long ago as the 1960s and 70s [12] to non-classical logic. The treatment of paraconsistent negation in the semantics of substructural logics [14] also makes use of a duality operator on possible worlds. Even without knowing where the inquiry will lead, it is natural to explore the relationships between fields suggested by the common mathematical thread.

REFERENCES

[1] James Bailey and Peter J. Stuckey, 'Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization', *Practical Aspects of Declarative Languages*, 174–186, (2005).

[2] Christoph Betz and Malte Helmert, 'Planning with h^+ in theory and practice', in *Proceedings of the 32nd German Conference on Artificial Intelligence (KI)*, pp. 9–16, (2009).

[3] Tom Bylander, 'The computational complexity of propositional STRIPS planning', *Artificial Intelligence*, **69**, 165–204, (1994).

[4] Jessica Davies and Fahiem Bacchus, 'Solving maxsat by solving a sequence of simpler sat instances', in *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP)*, pp. 225–239, (2011).

[5] Johan De Kleer, 'Hitting set algorithms for model-based diagnosis', in *Proceedings of the 22nd International Workshop on Principles of Diagnosis (DX)*, pp. 100–105, (2011).

[6] Thomas Eiter and Georg Gottlob, 'Identifying the minimal transversals of a hypergraph and related problems', *SIAM Journal on Computing*, **24**, 1278–1304, (1995).

[7] Naresh Gupta and Dana S. Nau, 'Complexity results for blocks-world planning', in *Proceedings of the 8th AAAI Conoference on Artificial Intelligence (AAAI)*, pp. 629–633, (1991).

[8] Patrik Haslum, John K. Slaney, and Sylvie Thiébaux, 'Minimal landmarks for optimal delete-free planning', in *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 353–357, (2012).

[9] Malte Helmert and Carmel Domshlak, 'Landmarks, critical paths and abstractions: What's the difference anyway?', in *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, (2009).

[10] Guanghui Lan, Gail W. DePuy, and Gary E. Whitehouse, 'An effective and simple heuristic for the set covering problem', *European Journal of Operational Research*, **176**, 1387–1403, (2007).

[11] Erick Moreno-Centeno and Richard M. Karp, 'The implicit hitting set approach to solve combinatorial optimization problems with an application to multigenome alignment', *Operations Research*, **61**, 453–468, (2013).

[12] Helena Rasiowa, *An Algebraic Approach to Non-Classical Logics*, North-Holland, Amsterdam, 1974.

[13] Raymond Reiter, 'A theory of diagnosis from first principles', *Artif. Intell.*, **32**, 57–95, (1987).

[14] Greg Restall, *An Introduction to Substructural Logics*, Routledge, Oxford and New York, 2000.

[15] John K. Slaney and Sylvie Thiébaux, 'Linear time near-optimal planning in the blocks world', in *Proceedings of the 13th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1208–1214, (1996).

[16] John K. Slaney and Sylvie Thiébaux, 'Blocks world revisited', *Artificial Intelligence*, **125**, 119–153, (2001).

[17] Roni Tzvi Stern, Meir Kalech, Alexander Feldman, and Gregory M. Provan, 'Exploring the duality in conflict-directed model-based diagnosis', in *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 828–834, (2012).