# Greedy Coverage of Incomplete Planning Domain Interpretations by $t$-strength Diagnoses

Daniel Bryce
SIFT, LLC.
dbryce@sift.net

**Abstract**

In this work, we present a greedy algorithm for covering the set of incomplete STRIPS planning domain interpretations by $t$-strength diagnoses. We present a greedy algorithm to cover the incomplete domain model interpretations with a set of plans by iteratively generating plans so that each additional plan is biased to cover at least one new interpretation not previously covered. We also present a second greedy algorithm to construct a set of plans that covers all $t$-strength diagnoses of plan failure for plans in the incomplete domain model. We show that covering domain interpretations by $t$-strength diagnoses leads to increased coverage by a set of plans despite potentially lower coverage per plan because covering by $t$-strength diagnoses leads to a more scalable approach to planning where more plans can be found.

## 1 Introduction

Our work is motivated by a problem encountered in decision support systems that plan activities for Mars exploration rovers [Bresina *et al.*, 2005], defense [Myers *et al.*, 2003], personal daily plans [Pollack *et al.*, 2003], and logistics [Carbonell *et al.*, 1991], among others. In these scenarios, the users model their knowledge about the application domain and use an automated planner to construct a sequence of actions. A problem arises in that the users are not always experts in knowledge engineering, and while understanding the domain, cannot state their knowledge perfectly. In such cases, it is possible to construct a plan given the incomplete knowledge of the domain (using our planner DeFAULT) [Weber and Bryce, 2011a], which can

JCMCC **86** (2013), pp. 33-49

be used to elicit more knowledge from the user [Weber and Bryce, 2011b]. When knowledge about the domain is incomplete; we find plans that cover as many interpretations as possible. The approach advocated in this work is to drive the elicitation process by presenting alternative plans that will succeed under different interpretations of the incomplete domain knowledge. Providing different plans can support user decision making because it is often easier to compare concrete examples [Viappiani *et al.*, 2006]. Furthermore, user confirmation of the plans leads to refined knowledge of the domain.

The contribution of this work is to formalize the problem of constructing a set of plans $\Pi$ as a set cover, where each plan $\pi$ does not cover a set of domain model interpretations $d(\pi)$ ($d(\pi)$ represents the domain mode interpretations where the plan will fail, and $\neg d(\pi)$ indicates those where it succeeds). We develop two variations (described below) of the well-known greedy set cover algorithm [Cormen *et al.*, 1990] that generate plans sequentially to cover uncovered domain interpretations, denoted by $\phi$. We say that $\Pi$ covers the interpretations represented by $|phi$ if $\wedge_{\pi \in \Pi} d(\pi) \wedge \phi = \bot$, or equivalently $\vee_{\pi \in \Pi} \neg d(\pi) = \phi$ (the union/disjunction of the domain interpretations where the plans succeed is equal to $\phi$). That is, the intersection/conjunction of the cases where the plans fail and the interpretations that need to be covered is logical false (i.e., an empty intersection). Our approach is different from the traditional set cover problem in that the sets $d(\pi)$ (elements not covered by each plan) are not known *a priori*, and each successive set $d(\pi)$ is generated by a planner that attempts to maximize coverage of the uncovered elements $\phi$.

The two variations of the greedy set cover algorithm differ in how they track the uncovered elements $\phi$. The first, an exact approach, represents the uncovered elements by binary decision diagrams (BDDs) [Bryant, 1986]. The second, an approximate approach, represents the uncovered elements by a set of bounded prime implicants of at most size $t$. We refer to the bounded prime implicants as $t$-strength diagnoses because of their correspondence in the diagnosis literature [Reiter, 1987; de Kleer and Williams, 1987]. Intuitively, a diagnosis is a conjunction of Boolean literals that if satisfied represents failure cases. If $\phi$ denotes a set of domain model interpretations that are not covered by plans in $\Pi$, and we approximate $\phi$ by $t$-strength diagnoses, then these are diagnoses of our failure to cover the domain model interpretations. That is, any domain interpretation that satisfies a diagnosis is not covered by $\Pi$, but because we only consider $t$-strength diagnoses there may also be domain interpretations not satisfying a diagnosis that are not covered by $\Pi$. The difference between the two approaches is that the exact method tracks exactly which domain model interpretations are covered by the plans, and the $t$-strength diagnosis method tracks only

those $t$-strength diagnoses covered by the plans (potentially leaving domain model interpretations uncovered).

Either representation of uncovered domain model interpretations can then be used within our planning algorithm to bias the next plan to cover the uncovered domain model interpretations. Using $t$-strength diagnoses leads to a more scalable approach to planning, as shown in our previous work [Weber and Bryce, 2011a]. We note that it may not always be possible to completely cover the incomplete domain interpretations because of either computational intractability faced by the planning algorithm or there are interpretations under which no plan exists. The question evaluated in this work is whether covering $t$-strength diagnoses as a heuristic for incomplete domain model coverage is scalable without giving up coverage quality. We find that indeed, using this heuristic can provide equal and sometimes better coverage because it scales better and can generate plans where the exact coverage approach cannot.

Our presentation begins with the background and representations used in planning, and then we describe our greedy coverage algorithms. We discuss how we generate plans in each iteration of the greedy algorithms and present an empirical evaluation of the techniques. We end with related work and a conclusion.

# 2 Background & Representation

We formulate planning with incomplete domain knowledge as Incomplete STRIPS planning. Incomplete STRIPS relaxes the classical STRIPS model [Fikes and Nilsson, 1971] to allow actions that have possible preconditions and effects [Garland and Lesh, 2002], and are otherwise identical to STRIPS domains. Much like planning with incomplete state information [Bonet and Geffner, 2000], the action incompleteness is not completely unbounded. The preconditions and effects of each action can be any subset of the state propositions $P$; the incompleteness is with regard to a lack of knowledge about which of the subsets correspond to each precondition and effect. We review the STRIPS model and then define Incomplete STRIPS.

## 2.1 STRIPS Planning Domains

STRIPS planning domains [Fikes and Nilsson, 1971] correspond to the classical planning model. A STRIPS planning domain $D$ defines the tuple ($P$, $A$, $I$, $G$), where $P$ is a set of Boolean propositions, $A$ is a set of complete action descriptions, $I \subseteq P$ defines a set of initially true propositions, and

$G \subseteq P$ defines the goal propositions. Each $a \in A$ defines $\text{pre}(a) \subseteq P$, a set of preconditions; $\text{add}(a) \subseteq P$, a set of add effects; $\text{del}(a) \subseteq P$, a set of delete effects.

For example, consider the following domain, which we will use a running example:

- $P = \{p, q, r, g\}$
- $A = \{a, b, c\}$
  - $\text{pre}(a) = \{p, q\}, \text{add}(a) = \{r\}, \text{del}(a) = \{\}$
  - $\text{pre}(b) = \{r\}, \quad \text{add}(b) = \{r\}, \text{del}(b) = \{q\}$
  - $\text{pre}(c) = \{q, r\}, \text{add}(c) = \{g\}, \text{del}(c) = \{\}$
- $I = \{p, q\}$
- $G = \{g\}$

A plan $\pi$ for $D$ is a sequence of actions, that when applied to the initial state, leads to a state where the goal is satisfied. A plan $\pi = (a_0, ..., a_{n-1})$ in a STRIPS domain $D$ is sequence of actions, that corresponds to a sequence of states $(s_0, ..., s_n)$, where $s_0 = I$; $\text{pre}(a_k) \subseteq s_k$ for $k = 0, ..., n$; $s_{k+1} = s_k \backslash \text{del}(a_k) \cup \text{add}(a_k)$ for $k = 0, ..., n-1$; and $s_n \subseteq G$.

For example, the plan $(a, b, c)$ corresponds to the state sequence $(s_0 = \{p, q\}, s_1 = \{p, q, r\}, s_2 = \{q, r\}, s_3 = \{q, r, g\})$, where the goal is satisfied in $s_3$.

## 2.2 Incomplete STRIPS Domains

An incomplete STRIPS domain $D$ defines the tuple ($P$, $A$, $I$, $G$, $F$), where: $P$ is a set of propositions, $A$ is a set of incomplete action descriptions, $I \subseteq P$ defines a set of initially true propositions, $G \subseteq P$ defines the goal propositions, and $F$ is a set of propositions describing incomplete domain features. Each action $a \in A$ defines $pre(a) \subseteq P$, a set of known preconditions, $add(a) \subseteq P$, a set of known add effects, and $del(a) \subseteq P$, a set of known delete effects. The set of incomplete domain features $F$ is comprised of propositions of the form $pre(a, p)$, $add(a, p)$, and $del(a, p)$, each indicating that $p$ is a respective possible precondition, add effect, or delete effect of $a$.

Consider the following incomplete domain:

- $P = \{p, q, r, g\}$

- $A = \{a, b, c\}$
  - $\text{pre}(a) = \{p, q\}, \text{add}(a) = \{\}, \text{del}(a) = \{\}$
  - $\text{pre}(b) = \{r\}, \quad \text{add}(b) = \{r\}, \text{del}(b) = \{\}$
  - $\text{pre}(c) = \{r\}, \quad \text{add}(c) = \{g\}, \text{del}(c) = \{\}$
- $I = \{p, q\}$
- $G = \{g\}$
- $F = \{pre(a, r), add(a, r), del(a, p), del(b, q), pre(c, q)\}$

A domain model interpretation $F^i \subseteq F$ of the incomplete STRIPS domain defines a STRIPS domain, in that every feature $f \in F^i$ indicates that a possible precondition or effect is a respective known precondition or known effect; those features not in $F^i$ are not preconditions or effects. For example, $F^0 = \{add(a, r), del(b, q), pre(c, q)\}$ is an interpretation that corresponds the STRIPS model example from the previous subsection.

**Incomplete STRIPS Plans:** A plan $\pi$ for $D$ is a sequence of actions, that when applied, *can lead* to a state where the goal is satisfied. A plan $\pi = (a_0, ..., a_{n-1})$ in an incomplete domain $D$ is a sequence of actions, that corresponds to the *optimistic* sequence of states $(s_0, ..., s_n)$, where $s_0 = I$, $pre(a_k) \subseteq s_k$ for $k = 0, ..., n$, $G \subseteq s_n$, and $s_{k+1} = s_k \backslash del(a_k) \cup add(a_k) \cup \{p | add(a, p) \in F\}$ for $k = 0, ..., n - 1$.

For example, the plan $(a, b, c)$ corresponds to the state sequence $(s_0 = \{p, q\}, s_1 = \{p, q, r\}, s_2 = \{q, r\}, s_3 = \{q, r, g\})$, where the goal is satisfied in $s_3$.

# 3 Greedy Coverage Algorithms

We present two algorithms for covering a set of incomplete domain model interpretations $\phi$. The first directly reasons with the domain interpretations (which are compactly represented by BDDs) under which plans fail, and the second indirectly reasons with $t$-strength diagnoses of plan failure (represented as prime implicants). While the objective of both algorithms is to cover domain interpretations, indirectly reasoning with diagnoses of plan failure promises to reduce the algorithm cost. Plan synthesis is intractable [Bylander, 1994], in even the STRIPS model. Plan evaluation (of existing plans) is #P-complete when directly reasoning with incomplete domain interpretations (which is phrased as propositional model counting [Gomes *et al.*, 2009]), but polynomial when indirectly reasoning with $t$-strength plan failure diagnoses.

**Algorithm 1**: Greedy Domain Interpretation and Diagnoses Cover

**Input:** $D$: Incomplete Planning Domain
**Output:** $\Pi$: Set of plans

```
1 Π ← {}; φ ← ⊤;
2 repeat
3 |  π ← Plan(D, φ);
4 |  d(π) ← Evaluate(π, φ);
5 |  φ ← φ ∧ d(π); Π ← Π ∪ {π};
6 until φ =⊥ or π = () ;
```

Our hypothesis is that based upon our findings in previous work [Weber and Bryce, 2011a], that plan synthesis guided by counting $t$-strength plan failure diagnoses results in plans of comparable quality to that of plan synthesis guided by counting domain interpretations. The hypothesis is that these results extend to the case of synthesizing plan sets by coverings. That is, we evaluate whether covering $t$-strength diagnoses of a plan set is comparable to covering domain interpretations of plan set. The potential reason why this hypothesis might not be supported is that covering $t$-strength diagnoses can lead to cases where fewer domain model interpretations are covered because the $t$-strength diagnoses over estimate the number of covered domain model interpretations.

Algorithm 1 lists pseudocode for greedily covering the set of incomplete domain interpretations $\phi$ by a set of plans $\Pi$. We represent $\phi$ in propositional logic, and note that each model (i.e., truth assignment to the propositions in $F$ that satisfies $\phi$) refers to an domain interpretation $F^i$. In the same manner, we represent the interpretations $d(\pi)$ under which a plan $\pi$ fails in propositional logic. The two approaches differ in the restrictions placed upon this representation, as either a BDD or set of prime implicants.

The greedy cover algorithm attempts to cover $\phi$ by the successful interpretations of plans in $\Pi$. The set of plans $\Pi$ is initially empty, and $\phi$ is initially set to logical true $\top$, where each domain interpretation corresponds to a model of $\top$. The algorithm repeatedly generates plans by invoking `Plan`, computes the failed domain interpretations $d(\pi)$ by invoking `Evaluate`, updating the set of plans, and tracking which interpretations remain to be covered in $\phi$. As previously noted, both $\phi$ and $d(\pi)$ are represented in propositional logic, with their models referring to sets of domain interpretations; the logical conjunction $\phi \wedge d(\pi)$ refers to the corresponding set intersection of the respective domain interpretations.

The $t$-strength diagnosis approach to coverage represents $\phi$ and $d(\pi)$ by prime implicants of at most cardinality $t$. We note that both $\phi$ and

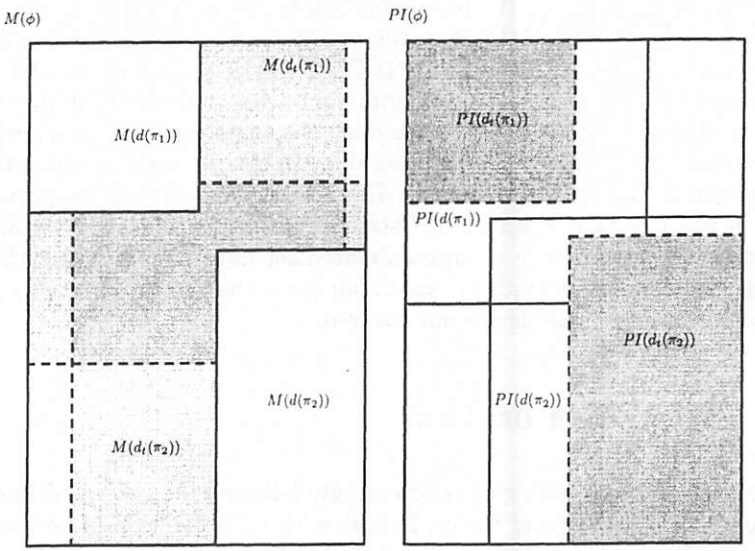


Figure 1: Example of the domains covered by plans using models $M(\phi)$ and prime implicants $PI(\phi)$.

$d(\pi)$ can be represented exactly with prime implicants, but are represented approximately with $t$-strength prime implicants. For example, when $t = 2$ and we take the conjunction $\phi_1 \wedge \phi_2$ where $\phi_1 = (a \wedge b) \vee c$ and $\phi_2 = (a \wedge d) \vee (c \wedge d)$, the unbounded prime implicant representation defines $\phi_1 \wedge \phi_2 = (a \wedge b \wedge d) \vee (c \wedge d)$ (after removing the subsumed conjunctive clauses to return to the prime implicant form), and the 2-strength representation defines $\phi_1 \wedge \phi_2 = (c \wedge d)$ (dropping the $(a \wedge b \wedge d)$ prime implicant because its cardinality is three). Intersection (taking the logical conjunction) of $\phi$ and $d(\phi)$ has the effect of determining those interpretations in $\phi$ that remain to be covered, and are not covered by $\pi$ (recall that $d(\pi)$ denotes interpretations where $\pi$ will fail).

Consider the abstract example in Figure 1. The two boxes denote the set of models $M(\phi)$ (i.e., domain interpretations) of $\phi$ and the set of prime implicants $PI(\phi)$ (i.e., diagnoses). The portions of the respective sets covered by two plans $\pi_1$ and $\pi_2$ are denoted by the regions in each box. There are two regions in each box corresponding to each plan; that correspond to the exact representation of the domain interpretations $d(\pi)$ and that

correspond to the $t$-strength representation $d_t(\pi)$. We note that the effect of approximating the prime implicants by bounding their cardinality is that: i) $M(d(\pi)) \subseteq M(d_t(\pi))$, the approximation overestimates the set of covered models, and ii) $PI(d_t(\pi)) \subseteq PI(d(\pi))$, by definition, the approximation reduces the number of prime implicants. Recall that the objective is to ensure that the set of plans does not share interpretations where the respective plans fail (i.e., the set of domain interpretations where the plans succeed are completely covered). Because of the $t$-strength approximation, it is possible that the sets of $t$-strength prime implicants have an empty intersection, while the un-approximated set does not, meaning that there are models where the plans can all fail (as is the case in Figure 1), and the domain interpretations are not covered.

# 4 Plan Generation

The approach to plan generation largely follows that described in our previous work [Weber and Bryce, 2011a], with the exception of how each successive invocation of the planner makes use of a different set of remaining domain interpretations to cover $\phi$. Recall that $\phi$ denotes a propositional sentence (in $t$-strength prime implicant form, or BDD form) that indicates which domain interpretations remain to be covered. The planner returns plans that do not fail in at least one remaining domain interpretation or diagnosis, or returns the empty plan, making each invocation during the greedy cover unique. In the following, we describe the plan synthesis approach and point out how it is modified from out previous work[Weber and Bryce, 2011a] to return such plans.

Algorithm 2 lists the pseudocode for the DeFAULT planning algorithm, which is based on greedy best first heuristic search [Russell and Norvig, 2010]. The algorithm involves search in the state space, in which states are represented by nodes and actions by edges. The algorithm maintains a list of *Open* states, whose successors can be added to the search graph, and a set of *Closed* states, whose successors have already been added. The *Open* list, which is implemented as a heap, will return the state with the lowest $h$-value when calling *pop*(). If the state is a goal state, then a plan is extracted by following the action edges back from the state to the initial state. If the state is not in *Closed*, then its successors are added to the search graph (one for each action). As part of adding a successor state, the failed domain interpretations or diagnoses and its heuristic are computed. We describe heuristic computation as solving a relaxed planning problem in the following subsection. We note that the addition beyond the prior work is to restrict the failure explanations in lines 10 and 12 to those cases also

**Algorithm 2:** `Plan`$(D, \phi)$

**Input:** $D$: Incomplete Planning Domain
**Output:** $\Pi$: Set of plans

$Open \leftarrow (s_0)$;
$Closed \leftarrow \{\}$;
**while** $Open \neq \emptyset$ **do**
  $s_k \leftarrow Open.pop()$; // Select $s_k$ with lowest $s.h$ value
  **if** $s_k \subseteq G$ **then**
    **return** $s_k.extractPlan()$;
  **else if** $s \notin Closed$ **then**
    $Closed \leftarrow Closed \cup \{s_k\}$;
    **for** $a \in A$ **do**
      $d(a_k, s_k) \leftarrow d(a_k, s_k) \wedge \phi$;
      $s_{k+1} \leftarrow s_k \backslash del(a) \cup add(a) \cup \{p | add(a, p) \in F\}$;
      $d(p, a_k, s_k) \leftarrow d(p, a_k, s_k) \wedge \phi$;
      $s_{k+1}.h \leftarrow RelaxedPlan(s_{k+1}, D, \phi)$;
      $Open \leftarrow Open \cup s_{k+1}$;
    **end**
  **end**
**end**

satisfying $\phi$ (the domain model interpretations remaining to be covered).

We denote by $d(\pi)$ a plan's failure explanations/diagnoses and by $\phi$ the current domain interpretations that we wish to cover, and both are represented by a propositional sentence defined over propositions in $F$.

We label predicted state propositions and actions with domain interpretations that will respectively fail to achieve the proposition or fail to achieve the preconditions of an action. That is, labels indicate the cases where a proposition will be false (i.e., the plan fails to establish the proposition). Labels $d(\cdot)$ are represented as propositional sentences over $F$ whose models correspond to failed domain interpretations.

For all $k \geq 0$, we define:

$$d(a_k, s_k) = d(a_{k-1}, s_{k-1}) \vee \bigvee_{\substack{p \in pre(a_k) \text{ or} \\ \phi \models pre(a_k,p)}} d(p, a_k, s_k) \vee \bigvee_{p:\phi ? pre(a_k,p)} (d(p, a_k, s_k) \wedge pre(a_k, p))$$

$$d(p, a_k, s_k) = \begin{cases} d(p, a_{k-1}, s_{k-1}) \wedge d(a_k, s_k) & : p \in add(a_k) \\ & \text{or } \phi \models add(a_t, p) \\ d(p, a_{k-1}, s_{k-1}) \wedge (d(a_k, s_k) \vee & : \phi ? add(a_k, p) \\ \qquad \neg add(a_k, p)) & \\ \top & : p \in del(a_k) \\ & \text{or } \phi \models del(a_k, p) \\ d(p, a_{k-1}, s_{k-1}) \vee del(a_k, p) & : \phi ? del(a_k, p) \\ d(p, a_{k-1}, s_{k-1}) & : otherwise \end{cases}$$

where $d(a_{-1}, s_{-1}) = \bot$, $d(p, a_{-1}, s_{-1}) = \bot$ if $p \in I$ and $\top$ otherwise, $\phi \models f$ if $\phi$ satisfies $f$, and $\phi ? f$ if $\phi \not\models f$ and $\phi \not\models \neg f$.

The intuition behind the label propagation is that an action $a_k$ applied in state $s_k$ will fail in the domain interpretations $d(a_k, s_k)$ where a prior action failed, a known precondition is not satisfied, or a possible precondition is not satisfied. As defined for $d(p, a_k, s_k)$, the plan will fail to achieve a proposition at time $k+1$ after applying $a_k$ in state $s_k$ in all interpretations where i) the plan fails to achieve the proposition at time $k$ and the action fails, ii) the plan fails to achieve the proposition at time $k$ and the action fails or it does not add the proposition in the interpretation, iii) the action deletes the proposition, iv) the plan fails to achieve the proposition at time $k$ or in the interpretation the action deletes the proposition, or v) the action does not affect the proposition and prior failures apply.

We define $d(\pi) = d(a_{n-1}, s_{n-1}) \vee \bigvee_{p \in G} d(p, a_{n-1}, s_{n-1})$. We note that this formula propagation is used in both plan synthesis and plan evaluation.

For example, our plan example from the previous section has the failure explanation label $d(\pi) = pre(a, r) \vee del(a, p) \vee (del(b, q) \wedge pre(c, q))$. If we count the number of interpretations that $\pi$ covers, they are the models of $\neg d(\pi) = \neg pre(a, r) \wedge \neg del(a, p) \wedge (\neg del(b, q) \vee \neg pre(c, q))$. For example, the interpretation $\{\}$ corresponding to the model $\{\neg pre(a, r) \wedge \neg del(a, p) \wedge \neg del(b, q) \wedge \neg pre(c, q) \wedge \neg pre(b, r)\}$ of $\neg d(\pi)$ is covered by the plan, along with several others for a total of six and leaving $2^5 - 6$ interpretations to be covered. If the next plan $\pi'$ has the failure explanation $d(\pi') = \neg add(a, r)$, then $d(\pi) \wedge d(\pi') = (\neg add(a, r) \wedge pre(a, r)) \vee (\neg add(a, r) \wedge del(a, p)) \vee (\neg add(a, r) \wedge del(b, q) \wedge pre(c, q))$. If instead we used 1-strength diagnoses, then $d_1(\pi) = pre(a, r) \vee del(a, p)$, $d_1(\pi') = \neg add(a, r)$, and $d_1(\pi) \wedge d_1(\pi') = \bot$ (after removing the 2-strength diagnoses $pre(a, r) \wedge \neg add(a, r)$ and $del(a, p) \wedge \neg add(a, r)$).

The search is guided by a heuristic computed in the `RelaxedPlan` subroutine. We refer the reader to [Weber and Bryce, 2011a] for a complete discussion. The relaxed planning problem can be solved in polynomial time, and involves solving a planning problem starting in the current state, but dropping all delete effects from the actions. We bias the solution of the relaxed plan to fail in as few domain interpretations as possible by using similar failure propagation to that above. We modify this propagation in this work to ignore interpretations that do not satisfy $\phi$ because we want to bias the plan toward solutions that cover new interpretations.

# 5 Evaluation

To evaluate the incomplete domain coverage algorithms, we attempted all instances in the test suite used in prior work [Weber and Bryce, 2011a]. In the following, we describe the test suite, the evaluation metrics used to compare the algorithms, the experiments, and results.

## 5.1 Test Suite

There are four domains that we use in the evaluation: a modified Pathways, Bridges, a modified PARC Printer, and Barter World. In all domains, we derived multiple instances by randomly (with probabilities 0.25, 0.5, 0.75, and 1.0) injecting incomplete domain features. The manner by which the features were injected varies by domain, as we describe below; we injected incomplete features that made sense for each domain, rather than modifying the domains in a uniform manner. For each probability of injecting incomplete features (except 1.0) and each instance, we derived ten instances with different random seeds and present the results for each seed. The problem instances generators and `DeFAULT` planner are available at https://github.com/danbryce/DeFault.

The Pathways domain from the international planning competition involves actions that model chemical reactions in signal transduction pathways. Pathways is a naturally incomplete domain where the lack of knowledge of the reactions is quite common because they are an active research topic in biology. We introduced each type of incompleteness to model incomplete knowledge of products required, created, or destroyed by reactions.

The Bridges domains consist of a traversable grid and the task is to find a different treasure at each corner of the grid. There are three versions where each subsequent version has an additional type of incompleteness. In

Bridges1, a bridge might be required to cross between some grid locations, modeled as an incomplete precondition. In Bridges2, many of the bridges may have a troll living underneath that will take all the treasure accumulated, and are modeled by a possible delete effect. In Bridges3, some of the corners may give additional treasures, modeled as a possible add effect. The instances involve different size grids (2, 4, 8, 16, and 32)

The PARC Printer domain from the international planning competition involves planning paths for sheets of paper through a modular printer. A source of domain incompleteness is that a module accepts only certain paper sizes, but its documentation is incomplete. Thus, for such modules a possible delete effect models that the module will become jammed.

The Barter World domain involves navigating a grid and bartering items to travel between locations. Items are available at different locations and may be required to travel between other locations. The domain is incomplete because some of the actions that acquire certain items are not always known to be successful (possible add effects) and traveling between some locations may require certain items (possible preconditions) and may result in the loss of an item (possible delete). The instances involve different size grids (2, 4, 8, 16, 32, and 64) and types of items (1, 2, and 4).

## 5.2 Experiments

To compare the algorithms, we evaluate them on each instance in the test suite and measure the total time to complete a covering and the final number of covered domain interpretations. We note that each invocation of the planner is limited to 5 minutes, and if exceeded, the planner returns the empty plan. If an empty plan is returned, the covering algorithm terminates. Thus, the covered domain interpretations may not include all possible domain interpretations – it is sometimes not possible to cover all interpretations because the instance is inherently not coverable or the planner exceeds the time limit. We compare the results in a pair-wise fashion, plotting the ratio of total time or covered domain interpretations between the exact coverage algorithm based on computing models, and the $t$-strength diagnosis based coverage algorithm.

## 5.3 Results

Figures 2 to 7 list the results as the coverage ratio and total time ratio between a $t$-strength diagnosis algorithm and the exact method. The first pair of plots lists the ratio results for the 1-strength diagnoses, the second

lists strength two, and the third lists strength three.

We see in Figure 2 that in a majority of instances the 1-strength diagnoses finds equal or better quality solutions (those instances with ratio greater than 1). In terms of run time, in Figure 3 the 1-strength approach tends to take considerably less (those instances with ratio less than 1). The same trends appear in Figures 4 to 7 when considering 2- or 3-strength diagnoses in comparison to the exact algorithm.

The way to interpret these results is that $t$-strength diagnoses appear to be overall more effective than the exact method. While the $t$-strength diagnoses technique tends to over estimate the number of domain interpretations covered by a plan, especially as $t$ becomes small, it still finds better quality plan sets. One explanation of this behavior is that reasoning with $t$-strength diagnoses is more computationally feasible and allows more scalable plan synthesis. That is, the exact approach may not be able to find any plan, where the $t$-strength diagnoses approach need only find one to have a better quality plan set. In the general case, finding one extra plan can make the difference in terms of quality. The total time results are also based on this observation because each call to the planner can be much faster under the $t$-strength approach. In summary, the ability of the $t$-strength approach to scale better on each step of the greedy algorithm helps it to attain better coverage in lower overall time.

# 6 Related Work

Planning to cover the interpretations of an incomplete domain will result in a set of plans that are diverse in the sense that they are subject to different failures, depending on the true domain. Synthesizing a diverse set of plans has been previously studied under similar motivations to ours, namely that the planner is situated within a decision support system and multiple plans provide the decision maker options.

Bryce *et al.* [2007] view diverse planning as a tool for eliciting preferences over plan optimization criteria by finding plans that optimize two objectives (cost and risk) differently by finding a Pareto set of plans. Coverage, in this case, is in finding plans that represent tradeoffs in the objectives. Bryce [2012] further studies how finding such diverse plans as multi-objective search can exploit positive interaction (i.e., shared plan prefixes) over multiple invocations of a single objective search.

Srivastava *et al.* [2007] also study diverse plan synthesis, but in terms of the structural properties of the plans (e.g., action sequences, causal links, and action partial orderings), and not their objectives. Diverse solutions

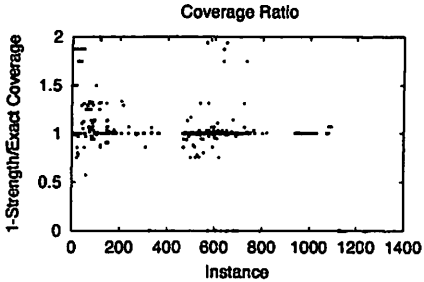


Figure 2: Coverage Ratio, 1-Strength Prime Implicants vs Exact

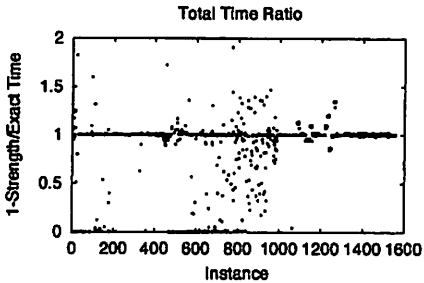


Figure 3: Time Ratio, 1-Strength Prime Implicants vs Exact

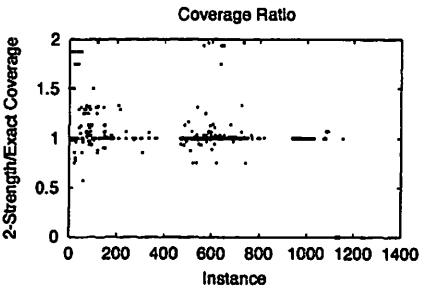


Figure 4: Coverage Ratio, 2-Strength Prime Implicants vs Exact

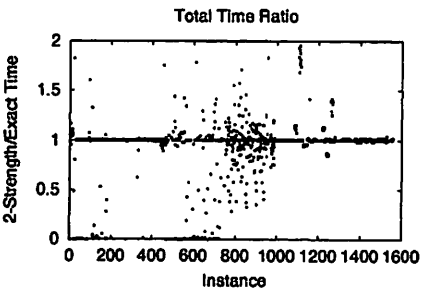


Figure 5: Time Ratio, 2-Strength Prime Implicants vs Exact

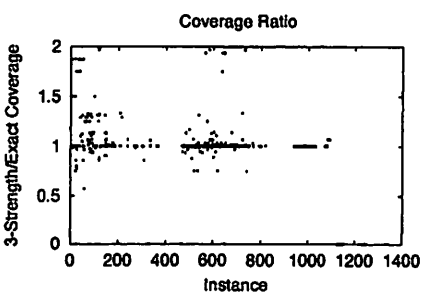


Figure 6: Coverage Ratio, 3-Strength Prime Implicants vs Exact

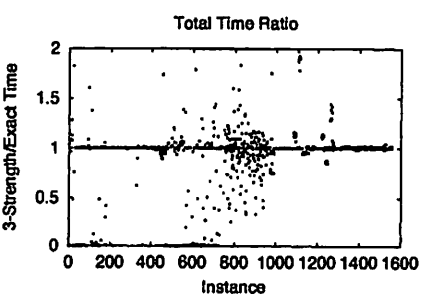


Figure 7: Time Ratio, 3-Strength Prime Implicants vs Exact

have also been studied in the context of constraint satisfaction problems [Hebrard *et al.*, 2005], which also rely on structural differences between solutions (i.e., variable assignments). This difference between objective and structural diversity is explained in the multi-objective optimization literature [Deb and Kalyanmoy, 2001] as diversity in the decision space versus diversity in the objective space. The work presented in this paper can be thought of as providing diversity in a third space – the information space (i.e., knowledge of the incomplete domain). Combining approaches to diverse planning to achieve simultaneous diversity across these spaces is currently an open topic.

# 7 Conclusion

We have shown that our finding that constructing a single plan by tracking $t$-strength diagnoses of plan failure also scales well when covering incomplete domain interpretations by a set of plans. The potential decrease in plan set coverage when using $t$-strength diagnoses is offset by the ability to scale and find more plans to cover the domain interpretations, so that while any one plan my cover fewer domain interpretations we can find more plans to attain better coverage.

# References


Fahiem Bacchus, Carmel Domshlak, Stefan Edelkamp, and Malte Helmert, editors. *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, Freiburg, Germany June 11-16, 2011.* AAAI, 2011.

B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proceedings of AIPS'00*, pages 52–61, 2000.

J. Bresina, A. Jonsson, P. Morris, and K. Rajan. Activity planning for the mars exploration rovers. In *Proceedings of ICAPS'05*, 2005.

R. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

Daniel Bryce, William Cushing, and Subbarao Kambhampati. Model-lite planning: Diverse mult-option plans and dynamic objective functions. In *Proceedings of the 3rd Workshop on Planning and Plan Execution for Real-World Systems*, 2007.

Daniel Bryce. Planning for multiple preferences versus planning with no preference. *ISRN Artificial Intelligence*, 2012:1–9, 2012.

Tom Bylander. The computational complexity of propositional strips planning. *Artif. Intell.*, 69(1-2):165–204, 1994.

Jaime G. Carbonell, Oren Etzioni, Yolanda Gil, Robert Joseph, Craig A. Knoblock, Steven Minton, and Manuela M. Veloso. Prodigy: An integrated architecture for planning and learning. *SIGART Bulletin*, 2(4):51–55, 1991.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

Kalyanmoy Deb and Deb Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

R. Fikes and N.J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. In *Proceedings of Second International Conference on Artificial Intelligence*, pages 608–620, London, United Kingom, 1971.

Andrew Garland and Neal Lesh. Plan evaluation with incomplete action descriptions. In *Proceedings of AAAI'02*, 2002.

Carla P. Gomes, Ashish Sabharwal, and Bart Selman. Model counting. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 633–654. IOS Press, 2009.

Emmanuel Hebrard, Brahim Hnich, Barry O'Sullivan, and Toby Walsh. Finding diverse and similar solutions in constraint programming. In Manuela M. Veloso and Subbarao Kambhampati, editors, *AAAI*, pages 372–377. AAAI Press / The MIT Press, 2005.

Karen L. Myers, Peter Jarvis, Mabry Tyson, and Michael Wolverton. A mixed-initiative framework for robust plan sketching. In Enrico Giunchiglia, Nicola Muscettola, and Dana S. Nau, editors, *ICAPS*, pages 256–266. AAAI, 2003.

Martha E. Pollack, Laura E. Brown, Dirk Colbry, Colleen E. McCarthy, Cheryl Orosz, Bart Peintner, Sailesh Ramakrishnan, and Ioannis Tsamardinos. Autominder: an intelligent cognitive orthotic system for

people with memory impairment. *Robotics and Autonomous Systems*, 44(3-4):273–282, 2003.

Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.

B. Srivastava, T.A. Nguyen, A. Gerevini, S. Kambhampati, M. B. Do, and I. Serina. Domain independent approaches for finding diverse plans. In *Proceedings of IJCAI'07*, pages 2016–2022, Hyderabad, India, 2007. IJCAI.

Paolo Viappiani, Boi Faltings, and Pearl Pu. Preference-based search using example-critiquing with suggestions. *J. Artif. Intell. Res. (JAIR)*, 27:465–503, 2006.

Christopher Weber and Daniel Bryce. Planning and acting in incomplete domains. In Bacchus et al. Bacchus *et al.* [2011].

Christopher Weber and Daniel Bryce. Reactive, proactive, and passive learning about incomplete actions'. In Bacchus et al. Bacchus *et al.* [2011].