IJCAI 2001 workshop on Stochastic Search
Seattle, Washington: http://www.ijcai-01.org
WP ref: 01/fantasy/fantasy (April 10, 2001).
URL ref: http://tim.menzies.com/pdf/01fantasy.pdf

Fantastic Stochastic Reductions in Design Options

Tim Menzies
Dept. Electrical Engineering & Computer Engineering, University of British Columbia, Vancouver, Canada;
<tim@menzies.com>

Abstract

Design means searching a space of options and this may be an enormous space. A mere 20 boolean options implies $2^{20} > 1,000,000$ possible arguments. If, in the usual case, this space of arguments contains many irrelevant and repeated disputes, then the space of *critical arguments* may be dramatically smaller than the space of all arguments. These critical arguments can be found via *stochastic abduction* plus *induction*.

1 Introduction

The late Herbert Simon characterized *design* as a search through a space of options [Simon, 1969]. This definition can be extended as follows: given some predicate *GOOD* that can assess a design, then a *design discussion* can be characterized as a debate between options in the design space that maximize the score of *GOOD*. In the usual case, the design discussion is complicated by a set of *uncontrollable variables* which are set via some nondeterministic process outside the control of the analyst.

At first glance, the number of design discussions seems depressing large. A mere 20 binary options implies more than $2^{20} > 1,000,000$ options. Combined with the uncontrollables, this implies that many of these options have a nondeterministic result.

Elaborate heuristic tools have been designed to cull the design discussion problem; e.g. [Steier, 1993]. But the topical nature of strong heuristic methods implies that these heuristics may have to laboriously hand-crafted for each new domain.

Taming the process of design discussions is a pressing industrial problem. We can't assume that software will be built by a single team in a single location using a single tool kit for a single purpose. Given recent advances in Internet technology (e.g. CORBA, the world-wide web), we should expect that software development will be geographically distributed. For such distributed development, it is pragmatic to permit the parallel development of separate 'work pieces" that will have to be unified at some later date. At that later date, we will always be searching a space of possibly contradictory ideas to find consistent parts of the unified design.

Searching a space of possibly contradictory ideas is NP-hard. Gabow et.al. showed that building pathways across spaces containing contradictions (e.g. the $\{x, \neg x\}$) is NP-hard for all but the simplest spaces (a space is very simple if it is very small, or it is a simple tree, or it has a dependency networks with out-degree ≤ 1) [Gabow *et al.*, 1976]. Hence searching designs can take exponential time and any practical proof procedure must use some form of incomplete search.

This article offers an optimistic alternative to the above pessimism. This new optimism is based on two points. Firstly, much research in the 1990s showed that theoretically slow NP-hard tasks are truly slow only for a very narrow range of problems. That same research applied stochastic search to NP-hard problems, with amazing success. For example, stochastic search methods are very effective for scheduling problems and can solve hard planning problems many times faster than traditional methods such as a systematic Davis-Putnam procedure [Kautz and Selman, 1996].

Secondly, many argument spaces contain *funnels*; i.e. a small number of critical variables that set all other variables in a system (the metaphor here is that all arguments run down the funnel) [Menzies *et al.*, 1999]. The concept of such critical variables has been reported in many domains. These have been called a variety of names such as:

- *Master-variables* in scheduling [Crawford and Baker, 1994];
- *Prime-implicants* in model-based diagnosis [Rymon, 1994] or machine learning [Rymon, 1993], or fault-tree analysis [Lutz and Woodhouse, 1999].
- *Backbones* in satisfiability [Parkes, 1999; Singer *et al.*, 2000];
- *Dominance filtering* in design (described below);
- *Minimal environments* in the ATMS [DeKleer, 1986];
- The *base controversial assumptions* of HT4 [Menzies and Compton, 1997].

The core intuition in all these terms is the same: what happens in the total space of a program is controlled by a small critical region within the program. The overall size of the design space is exponential on the number of different assignments to funnel variables in this critical region. Systems with *narrow funnels* have very few funnel assignments. If such small funnels exist, then the total space of design options can reduce to merely the space of options sanctioned by the critical zones. Once these are options are discussed and resolved, then the rest of the argument space reduces dramatically and there is little left to debate.

Finding the funnels can be complex. For example, computing the base controversial assumptions is a NP-complete task [Menzies *et al.*, 2001]. But note that we don't have to try to hard to find the funnel. Since the funnels control the search space, we need not seek the funnel, it will find us. *Any* stochastically selected pathway to goals must pass through the funnel (by definition). That is, repeated application of some fast stochastic search technique will stumble across the funnel variables, providing that search technique reaches the goals.

Stochastic tools are hence a possible method for exploring and reducing design options. This paper explores this possibility. A simple stochastic search engine called CHEETAH exercises argument spaces expressed in the JANE rule-based language. A monitor called TARZAN watches from above as CHEETAH chases JANE around the argument space. TARZAN builds a log of Jane's behaviour and learns how to nudge JANE into better behaviour. Funnel theory predicts that if design spaces contain narrow funnels, then the nudges will be few and fast to find.

The rest of this paper is structured as follows. Some related work is discussed first. This is followed by a description of funnel theory; evidence in the literature of narrow funnels; and a description of some experiments with JANE/CHEETAH/TARZAN.

2 Related Work

The connection of this work to abduction is discussed below (see §4.4). Another related work is the MAT SAT problem discussed by (e.g.) Asano and Williamson [Asano and Williamson, 2000]. In MAX SAT, each clause gets a weight and the inference problem is to find a set of variable bindings that maximizes the sum of the weights of the satisfied clauses. While the problem is NP-hard, polynomial time algorithms are known that generate solutions with a performance guarantee of α. Theoretically, $\alpha \leq \frac{7}{8} = 0.875$ except in the unlikely event that $P = NP$. The state-of-the-art in MAX SAT are algorithms that generate $\alpha = 0.8331$ [Asano and Williamson, 2000].

Standard MAX SAT is different to the design discussion problem discussed here in several ways. An open issue at this time is whether or not these differences rule out the use of MAX SAT in this domain.

- The design option space need not be necessarily be expressible as a finite CNF. In particular, if the software being designed is for *simulation* purposes, then multiple assignments may be made to the same variable, at different times in the simulation.
- Not all the variables in the theory can be set with certainty by the proof procedure. Recall that the uncontrollable variables are set via some nondeterministic process outside the control of the analyst. That is, the assignments made by a proof procedure to the uncontrollables are uncertain. Hence, the task in design discussion is to find *control actions* (assignments to the controllables) that tend to maximize the $GOOD$ness of design, *whatever* assignments are made to the uncontrollables.
- In the example shown below, the theory is explicitly available and is written in the JANE syntax. The CHEETAH system explores the options within the JANE rules to build a data set that TARZAN can process. In other cases, only the data set is available and not the theory. For example, in §4.2, an example will be presented where a procedural model generated data from Monte Carlo simulations of the inputs. The process discussed in the this article should be able to support design discussions in the theory-less case.

3 Funnel Theory

Funnel theory is a claim that within the space of arguments, there exist a small number of key decisions that determine all others. To introduce funnels, we first say that an argument space supports *reasons*; i.e. chains of reasoning that link inputs in a certain context to desired goals. Chains have links of at least two types. Firstly, there are links that clash with other links. Secondly, there are the links that depend on other links. One method of arguing less is to first debate the non-dependent clashing links. The resolutions to these arguments will have the greatest impact of reducing the subsequent argument(s). For example, suppose the following argument space is explored using the invariant $nogood(X, \neg X)$ and everything that is not a $context$ or a $goal$ is open to debate:

$$\begin{array}{r}
a \longrightarrow b \longrightarrow c \longrightarrow d \longrightarrow e \qquad\qquad\qquad \\
context1 \longrightarrow f \longrightarrow g \longrightarrow h \longrightarrow i \longrightarrow j \longrightarrow goal \\
context2 \longrightarrow k \rightarrow \neg g \longrightarrow l \longrightarrow m \rightarrow \neg j \longrightarrow goal \\
n \longrightarrow o \longrightarrow p \longrightarrow q \longrightarrow \neg e \qquad\qquad\qquad
\end{array}$$

While all of $\{a, b, ..q\}$ is subject to discussion, in the context of reaching some specified goals from *context1* and *context2*, the only important disputes are the clashes $\{g, \neg g, j, \neg j\}$. The $\{e, \neg e\}$ clash is not exercised in the context of $context1, context2 \ \vdash \ goal$ since no reason uses e or $\neg e$. Since $\{j, \neg j\}$ are fully dependent on $\{g, \neg g\}$, then the core of this argument is one variable $(\{g\})$ with two disputed values: true and false.

The *funnel* of an argument space contains the non-dependent clashing links; e.g. $\{g\}$[1] The arguments with *greatest information content* are the arguments about the funnel variables, since these variables set the others. If the space contains *narrow funnels* then the total argument space can be greatly reduced to a small number of highly informative disputes about funnel variables. For example, suppose our stakeholders agree that g is true, then in the context of arguing about how $context1, context2 \ \vdash \ goal$, the argument space reduces to:

$$context1 \longrightarrow f \longrightarrow g \longrightarrow h \longrightarrow i \longrightarrow j \longrightarrow goal$$

[1]Readers familiar with the ATMS [DeKleer, 1986] will note the similarities between the funnel and ATMS *minimal environments*. However, while both approaches rely on some *nogood* invariant, there are significant differences between the consistency-based *total envisionments* of the ATMS and the set-covering *relevant envisionments* discussed here; see [Menzies and Compton, 1997] for details.

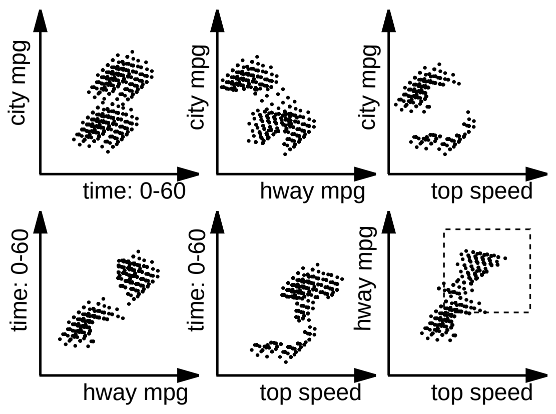

Figure 1: A stylized version of the SFV Viewer. Adapted from [Josephson *et al.*, 1998].

The reasoning starting with k has been culled since, by endorsing g, we must rejects all lines of reasoning that use $\neg g$. Also, the reasoning starting with a, n are ignored since they are irrelevant in this context; i.e. they do not participate in reaching a desired goal. Further, in this context, there is little point arguing about $\{f, h, i, j\}$ since if any of these are false, then no goal can be reached.

This small example shows how to argue less through funnel-based reasoning. Funnel-based argumentation finds the key arguments, and ignores numerous irrelevant arguments. In the above example, a argument space containing up to $2^{16} = 65536$ discussions about 16 boolean variables $\{a..q\}$ has been reduced to one discussion about one variable; i.e. "is g true or false?".

4 The Funnel Phenomena

If narrow funnels were common, then we should find evidence that large spaces of options can be reduced to a much smaller space of key options. These section offers several examples of exactly this phenomena.

4.1 SFV

The Seeker, Filter, Viewer architecture (SFV) explores a large space of design options [Josephson *et al.*, 1998]. The core intuition of SFV was that algorithmic approaches to design need not prematurely cull design options. Given the space CPU available to (e.g.) most engineering firms, a huge space of design options could be generated. This space could then be explored using a *dominance criteria*:

> We say that design candidate A dominates candidate B if A is superior or equal to B in every dimension of evaluation and distinctly superior in at least one dimension. Dominated designs need not be considered further and they may be filtered out [Josephson *et al.*, 1998].

SFV was first tested in the domain of hybrid electric vehicle design. Hybrid electric vehicles are automobiles that use both an electrical motor and an internal-combustion (IC) engine as power sources. Such designs must make extensive

abbreviations		current situation	proposed changes
prec = 0..5	precedentness	0, 1	
flex = 0..5	development flexibility	1, 2, 3, 4	1
resl = 0..5	risk resolution	0, 1, 2	2
team = 0..5	team cohesion	1, 2	2
pmat = 0..5	process maturity	0, 1, 2, 3	3
rely = 0..4	required reliability	4	
data = 1..4	database size	2	
cplx = 0..5	product complexity	4, 5	
ruse = 1..5	level of reuse	1, 2, 3	3
docu = 0..4	doco requirements	1, 2, 3	3
time = 2..5	runtime constraints	?	
stor = 2..5	main memory storage	2, 3, 4	2
pvol = 1..4	platform volatility	1	
acap = 0..4	analyst capability	1, 2	2
pcap = 0..4	programmer capability	2	
pcon = 0..4	programmer continuity	1, 2	2
aexp = 0..4	analyst experience	1, 2	
pexp = 0..4	platform experience	2	
ltex = 0..4	experience with tools	1, 2, 3	3
tool = 0..4	use of software tools	1, 2	
site = 0..5	multi-site development	2	
sced = 0..4	time before delivery	0, 1, 2	2
	# of combinations=	$6 * 10^6$	

Figure 2: A NASA software project. Unknowns in the current situation are shown as ranges or, in the case of total lack of knowledge, a "?".

trade offs between designs optimized for IC motors or electric motors. Millions of design options can be generated. Hence, after applying dominance, there are still thousands of options to consider. SFV's Viewer displayed all these options on all pairs of assessment criteria. Figure 1 shows that display: each dot represents one design. Note that the user has selected a region in the right-bottom plot. The user has indicated that the selected region is acceptable to them. All the designs within the preferred region now wink in the other plots; i.e. the user can see how their preference impacts on other dimensions. Also, the user can now remove from all plots the designs outside the selected region; i.e. the user can concentrate only on promising designs.

In a result consistent with narrow funnels, the SFV experience is 99 to 99.9% of those options could be culled. Josephson (personal communication) reports that dominance plus the Viewer let a design team find 7 best designs from a space of $2 * 10^6$ options.

4.2 Software Project Risk

Menzies & Sinsel found that a space of 54 million options contained found two key variables that could most control the rest of the system [Menzies and Sinsel, 2000]. In that application, a COCOMO-based tool [Madachy, 1997] was used to evaluate the risk that a NASA software project would suffer from develop-time overrun (that project is shown in Figure 2. The tool used in that study required a guesstimate of the source lines of code (SLOC) in the system and certain internal tuning parameters which, ideally, are learnt from historical data. Lacking such data, Menzies & Sinsel used

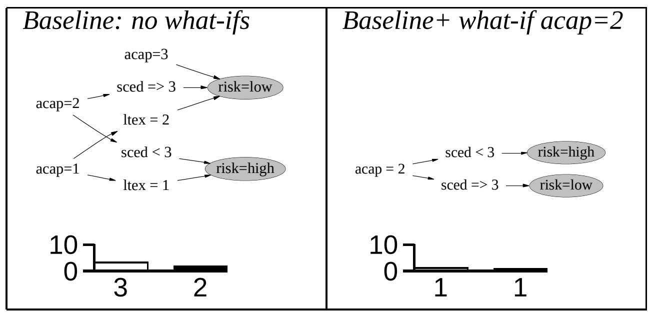

Figure 3: TOP: A decision tree (left) and a pruned tree (right) holding all branches that do not contradict *acap=2*.
BOTTOM: Number of branches to different risk classification. Legend: ☐ =low risk ■ =high risk.

three guesses for SLOC and three sets of tunings which they took from the literature. Competing stakeholders proposed 11 changes to a project. Some of the project features were unclear and, for those features, project managers could only offer ranges for the required inputs to the COCOMO-based tool. These ranges offered 2930 possible combinations for the inputs. When combined with the other uncertainties, this generated a space of 54 million possibilities ($2930 * 2^{11}$*three guesses for SLOC * three tunings).

Faced with this overdose of possibilities, Menzies & Sinsel performed 50,000 Monte Carlo simulations where the inputs were taken from the 54 million possibilities. A machine learning program generated decision trees from the 50,000 runs. A tree query language called TARZAN then swung through the learnt trees looking for the least number of attribute ranges that had the biggest impact on the overall software development risk. TARZAN treated the learnt trees as a space of possibilities within the logged behaviour. TARZAN what-if queries by pruning all branches in the learnt trees that contradict our what-if possibility[2]. For example, if we wonder "what-if acap=2", then Figure 3, top left, would be pruned to Figure 3, top right. This particular "what-if" turns out to be a bad idea. The histograms in Figure 3, bottom, show that this pruning drives us into a situation where the ratio to low risk to high risk projects changes from 3:2 to 1:1. That is, if *acap=2*, then we increase our chances of a high-risk project.

Figure 4 shows some of the what-if queries conducted over the trees learnt from the 50,000 runs. The baseline risk profile is shown in cell A1 of Figure 4: prior to the what-if queries, the learnt trees hold branches to 7,24,8 low,medium,high risk projects respectively. Seven of the proposed changes had little impact on the baseline. Of the remaining four proposed changes, two are clearly superior. Cell A2 shows that that having moderately talented analysts and no schedule pressure (*acap=[2], sced=[2]*) reduced the risk in this project nearly as much as any other, larger subset. Exception: B2

[2]The fanciful name of TARZAN arose when it was realized that these what-if queries are like benevolent agents swinging through the trees looking for ways to change what is going on. Tools that extend TARZAN should come from the same genre. Hence, JANE and CHEETAH.

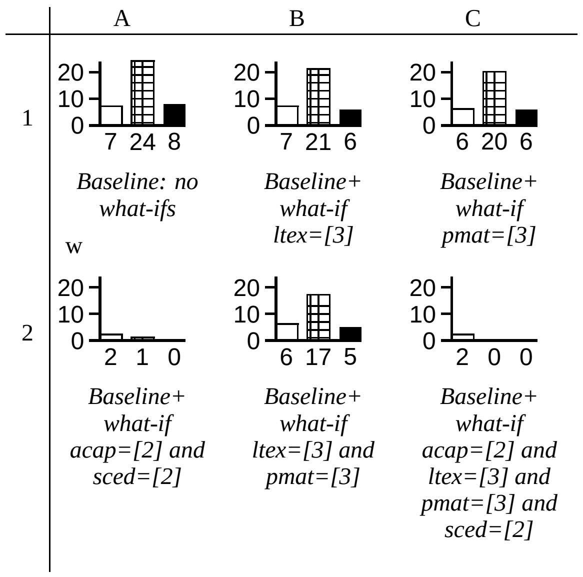

Figure 4: Number of branches to different risk classifications. Legend: ☐ =low risk ▦ =medium risk ■ =high risk.

applies actions to remove all branches to medium and high risk projects. Nevertheless, Menzies & Sinsel recommended A2, not B2, since A2 seemed to achieve most of what B2 can do, with much less effort.

Note that Figure 4 takes $\frac{1}{6}$th of a page to display and shows the key factors that control the classifications of 54,000,000 possibilities. This astonishing reduction in the argument space is consistent with the COCOMO-based tool containing narrow funnels.

4.3 MYCIN

Medical diagnosis is like a design task in that trades offs are made between competing diagnoses. To implement this trade-off process, the MYCIN medical expert system allowed authors to express their certainty in a rule as a certainty factor (cf) between -1000 to +1000 [Buchanan and Shortliffe, 1984]. Some ad-hoc combination rules were defined to control how these certainties propagated and combined. Subsequent research rebelled at the ad-hoc nature of the MYCIN approach and proposed more theoretically satisfying uncertain reasoning scheme [Gordon and Shortliffe, 1985].

In a result consistent with funnel theory, experiments suggested that elaborate reasoning about options within MYCIN was not necessary. Clancey and Cooper mapped the cfs into the nearest of N values. In the original MYCIN, N=2001. The Clancey and Copper analysis set N to 10,5,4,3, and 2. Only after $N \leq 3$ did this significantly effect the competency of MYCIN [Buchanan and Shortliffe, 1984]. Further investigation revealed that the MYCIN rules formed a broad and shallow reasoning network. That is, the topology of the MYCIN argument space was such that any the unknowns need only ever interact with a handful of other uncertainties. Consequently, arguing about the details of large scale interactions is irrelevant since such large scale interactions could not occur in that system.

4.4 HT0 and HT4

In a results consistent with the Clancey and Cooper results, it was found that most of the choices made within a space of conflicts had the same net effect [Menzies *et al.*, 1999]. That study compared two *abductive inference strategies*. Abduction is a method of tracking the choices made while studying a model. An abductive inference engine searches for goals while ensuring that all choices remain compatible [Kakas *et al.*, 1998]. When faced with incompatible choices, an abductive device has at least two choices. In *full worlds search*, the abductive device forks one *world of belief* for each possible resolution to the choice. In *stochastic worlds search*, the abductive device selects one resolution at random, then continues on. Stochastic worlds search is usually performed inside a "rest-retry" mechanism. That is, for a limited number of retries, when the stochastic search runs out of new options, all options are retracted and the whole stochastic worlds inference procedure runs again. In a very large case study (over a million runs), Menzies, Easterbrook, Nuseibeh and Waugh found that the average difference in reachable goals between the stochastic worlds search and full worlds search was less than 6% (!!); see Figure 5. That is, stochastic conflict resolution reached as many parts of an argument space as a more rigorous method. This result can be explained via funnel theory. Assuming narrow funnels, then the stochastic worlds search used by Menzies, Easterbrook, Nuseibeh, Waugh found as many goals as the full worlds search since both searches were controlled by the same funnels.

In yet another study, Menzies and Micheal [Menzies and Michael, 1999] showed that stochastic worlds search found 98% of the goals found by a full worlds search [Menzies and Michael, 1999] (a result consistent with Menzies, Easterbrook, Nuseibeh and Waugh). More interesting from a pragmatic perspective, the full worlds search ran in time exponential to model size while the stochastic abductive search can run much faster and scaled up to very large models (see Figure 6). This result can be explained via funnel theory. The stochastic worlds search used by Menzies & Micheal ran extremely fast since it could quickly sample the funnels without all the overheads of the more rigorous search.

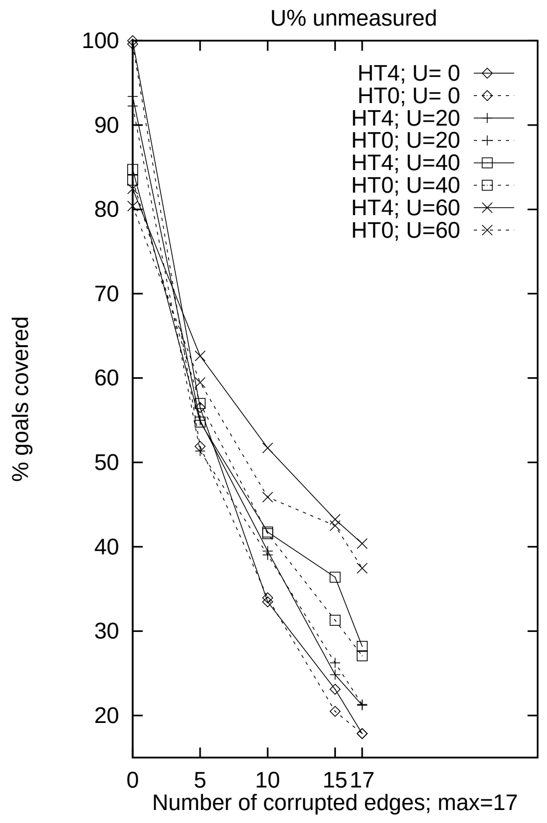

Figure 5: Rigorous standard abductive search (solid line) vs stochastic abductive search (dashed line). Y-axis denotes percentage of goals reached. As errors are introduced into the theory (increasing the x-axis value), this percentage decreases. U denotes what percentage of the theory was not mentioned in the input-output sets.

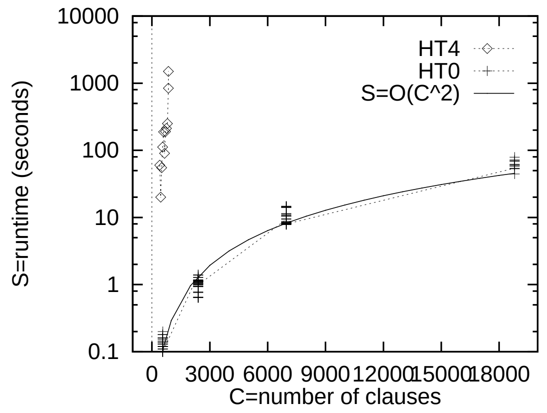

Figure 6: Rigorous standard abductive search (HT4) vs stochastic abductive search (HT0).In the zone where both algorithms terminated, HT0 reached 98% of the goals reached by HT4.

5 An Argument Reduction Environment

JANE/CHEETAH/TARZAN is a general toolkit for supporting less arguments based on stochastic abduction, followed by induction. JANE is a simple rule-based language for expressing options in a domain. Each rule and fact in JANE is stamped with the name of the author and the time and date of its creation. Rules and facts from different stakeholders can hence be stored together in one rule-base. Also, each rule and fact gets a heuristic *chances* measure (range 0 to 1) that stores the likelihood of that fact/rule. Finally, a dollar $cost$ value is added to each fact/rule. In the current version of JANE, $cost$ is a once-off set-up cost. Hence, if (e.g.) a fact is accessed more than once, its associated dollar cost is only incurred the first time.

$Chances$ and $cost$ need not be specified exactly. JANE authors can specify a minimum and maximum value, option-

```
 1  tim= [month=jan,day=18,year=2001
 2       ,elm='tim@menzies.com'].
 3  bob= [month=feb,day=10,year=2001
 4       ,elm='robertf@zbm.com'].
 5
 6  tim says cost = 0 and chances = 1.
 7  r1 if  rich rors healthy rors content
 8      then happy.
 9  r2 if  not tranquil then rich.
10  r3 if tranquil     then content.
11  r4 if no sick      then healthy.
12  r5 if overweight   then sick.
13  r6 if no exercise  then overweight.
14  t7 if baseball rany running rany swimming
15        rany football then exercise
16
17  bob says cost= 1 to +4 and chances= +0 to 1.
18  r8  if   enthusiasm rand likesSweat
19      then baseball.
20  r9  if   enthusiasm rand likesSweat
21      then running.
22  r10 if   enthusiasm rand likesSweat
23      then football.
24
25  tim says cost= 1 to +4 and  chances= +0 to 1.
26  r11 if   enthusiasm rand not likesSweat
27      then swimming.
28
29  tim says cost = 2 and chances = 1.
30  r12 if   true then enthusiasm.
31
32  run  :- prove(happy).
33  runs :- time(proves(1000,'experience.dat').
```

Figure 7: A sample JANE knowledge base.

ally marked with some "skew". For example, a sample JANE rule base, showing contributions from two stakeholders (Tim and Bob) is shown in Figure 7. Line 6 shows an exact specification of $cost$s and $chances$ while line 17 shows a range specification with a "+" symbol showing the skew. Internally, the skew is implemented as a $beta(X)$ distribution with mean X. $+0\ to\ 1$ means that the range is the the random variable $0 + beta(0.33) * (1 - 0)$ while $1\ to\ +4$ means that the range is a random variable $1 + beta(0.67) * (4 - 1)$. Note also that the rules $\{r1, r2, r3, r4, r5, r6\}$ all have the same $cost$ and $chances$ shown at line 6. Similarly, the $cost$ and $chances$ of $\{r8, r9, r10\}$ is set at line 17.

CHEETAH is a stochastic abductive inference engine that interprets rules written in the JANE syntax. CHEETAH supports *assumption-based reasoning* and *random walk*. If a JANE rule condition requires some assumption, and there is no evidence for or against that condition, then CHEETAH just makes the assumption. For example, rule $r11$ can only prove $swimming$ if it assumes $not\ likesSweat$. This assumption rules out $baseball, running$ or $football$ since those conclusions require $likesSweat$ (see line 18,20,22 in Figure 7).

Since assumptions rule out other conclusions, CHEETAH uses a *random walk* mechanism for stochastically selecting which assumptions are made. This random walk is a simple adaptation of standard disjunctions and conjunctions. In standard languages, if a test is specified as $X\ and\ Y\ and\ Z$ then that test is executed left-to-right to test X before Y before Z. CHEETAH supports the standard left-to-right and and or as well as a random ordered test $rand$ and ror. If a condition is specified (e.g.)

$$swimming\ ror\ football\ ror\ baseball$$

then the order of traversal is picked randomly. Recalling the last paragraph, then CHEETAH may or may not try to prove $swimming$ before $football$ in which case assumptions about our dislike of sweating would favor swimming and rule out the other sports.

When multiple methods exist for proving something X, then our belief in $\neg X$ should decrease. This is implemented via the noX operator which sums the evidence for X into Sum, then returns $1 - Sum$.

Other random walk operators of interest are $rors$ and $rany$ (see lines 7 and 14,15 in Figure 7). $Rors$ specifies a set of goals which we desire and $rany$ specifies a set of required, but not totally desirable goals. For example, $happiness$ might result from being $rich$, $healthy$ and $content$. However, in this imperfect world it is rare that we can achieve $rich$ <u>and</u> $healthy$ <u>and</u> $content$. Hence we combine them with a $rors$ to ask CHEETAH to try and prove as many of them as possible.

$Rany$ is similar in concept to $rors$, but opposite in intent. While $exercise$ could be done via $baseball$ <u>and</u> $running$ <u>and</u> $swimming$ <u>and</u> $football$, we probably don't want to do all four exercises at once since this might lead to (e.g.) muscle damage. Hence we combine them with a $rany$ which must prove at least one of them, but after that, it ignores some randomly selected portion of the $rany$ goals. As with $rand$ and ror, the traversal order of the testing in $rany$ and $rors$ is picked at random.

$Rors$ and $rany$ adopts the HT0 [Menzies and Michael, 1999] method for traversing a space of assumptions: *one shot-proofs*, *random ordering*, plus *reset-retry*. When proving a set of goals $X_1\ rors\ X_2\ rors\ X_3$..., then the goal X_j only gets *one-shot*. If a proof of X_j fails, then the system does not backtrack to find different solutions to prior goals $X_1 ... X_i (i = j - 1)$. Instead, X_j is marked as unproved and $rors$ skips on to the next goal X_{j+1}. One-shot is a very weak method for proving something and only it works in domains with narrow funnels (where any shot in the dark while hit something of interest). Numerous experiments [Menzies and Michael, 1999] strongly suggest that when one-shot is combined with *reset-retry*, then one-shot greatly reduces the computational cost of searching a space of contractions. Note that if we juggled the order of the goals, then we might avoid making an assumption before search for X_j that makes X_j impossible. Such order juggling comes for free as part of the stochastic traversal order of a $rors$, plus CHEETAH's *reset-retry* mechanism. Line 33 of Figure 7 shows that CHEETAH is called many times, with the resulting behaviour logged to the file *experience.dat*. Between each run, CHEETAH *resets* its assumption memory and *retries* its high-level goals for another time. During this later test, if the same $rors$ is accessed, then the random ordering or the $rors$ operator means that the goals may be explored in a different order. This im-

operator	X_{cost}	$X_{chances}$
$X =$ $or(Y)$ $ror(Y)$	$first(Y).cost$	$first(Y).chances$
$X =$ $and(Y)$ $rand(Y)$	$\sum_{i=1}^{\|Y\|} cost(Y_i)$	$\prod_{i=1}^{\|Y\|} chances(Y_i)$
$X =$ $rors(Y)$ $ors(Y)$ $rany(Y)$ $any(Y)$	$Z \subseteq Y$ $\sum_{i=1}^{\|Z\|} cost(Z_i)$	$Z \subseteq Y$ $\sum_{i=1}^{\|Z\|} chances(Z_i)$
$X =$ $no(Y)$	$\sum(allY.cost)$	$1 - \sum(allY.chances)$

Figure 8: The SET1 combination rules for $costs$ and $chances$. In this table, (a) the function "$first(Y)$" returns the first proved element in Y; (b) Z is the subset of Y that is proved by the operator; (c) $allX$ finds all solutions to X; (d) the value of $chances$ is always capped to one.

plies that different assumptions may be made before the proof reaches X_j and, hence, X_j may be provable for some subset of CHEETAH's runs.

TARZAN performs induction over the logs of behaviour seen when the CHEETAH abductive inference engine explores the JANE rules. Recall from Figure 7, when CHEETAH runs JANE, a log of JANE's behaviour is stored in *experience.dat*. TARZAN searches that log looking for the *fewest* number attribute ranges that have the *largest* impact on the overall behaviour of the system. An example of TARZAN will be presented below, after some discussion on certain open issues.

5.1 Open Issues

There are many features of the Figure 7 rule base that are open to debate. For example, what exactly are the precise $cost$s and $chance$s for each rule and how are we to tally them together? JANE tallies $cost$s and $chances$ using a customizable set of *combination operators*. This set can easily be changed but this leaves open the question: which operators should be used?

One set of combination rules is shown are the rules known as SET1, shown in Figure 8. The rules for $\{or, ror, and, rand\}$ are simple enough. However, $\{rors, ors, rany, any, no\}$ are more complex, more open to debate. The source of the complexity is that these operators search for multiple solutions within a disjunction. It could be argued that as the amount of evidence increases, the higher the $chances$ but the greater the $cost$ (since evidence collection is expensive). Hence, for $\{rors, ors, rany, any\}$ both $cost$ and $chances$ are summed together.

(Note the absence of a not operator in SET1: JANE applies deMorgan's theorem to convert e.g. $a\ and\ not\ (b\ and\ c)$ to $a = t\ and\ (b = false\ or\ c = false)$. Hence, at runtime, not is never called.)

SET2 is another set of combination rules which is almost the same as SET1 but takes a different stance on how $cost$s are combined. In SET2, the $cost$ of finding multiple solutions within a disjunction (i.e. $\{rors, ors, rany, any\}$) is the $maximum$ of the cost of the proved parts of the disjunction.

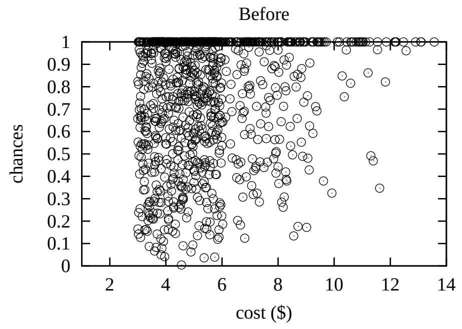

Figure 9: $Cost$ and $chances$ from 1,000.

		Cost	
		low if $\leq$ \$5	high if $>$ \$5
Chances	low if $\leq$ 0.85	24.7%	24.3%
	high if $>$ 0.85	21.7%	29.4%

Figure 10: Percentage of 1,000 runs that fall into classes that combine low/high $cost$ and low/high $chances$.

In keeping with the whole JANE/CHEETAH/TARZAN approach, if a debate is possible, we should randomly simulate across the space of possibilities, then use induction to check which (if any) of the debate points are key. In the case study shown below, JANE simulated:

- Across the $cost$ and $chances$ range specified in Figure 7.
- Using either SET1 or SET2 (picked once for each run).

5.2 An Example

Figure 9 shows the $cost$ and $chances$ seen in 1,000 proofs of $happy$ (as defined on lines 6,7 of Figure 7). TARZAN's task of restricting this range of behaviour begins by dividing the output into several classes. The classifications shown in Figure 10 were chosen so as to balance the size of the different classes (classes of different sizes can bias an inductive learner). Of these classes, one is clearly inferior (low $chances$, high $cost$), and one is clearly superior (high $chances$, low $cost$). TARZAN's task is to find methods for nudging the system away from inferior and toward superior classes.

The version of TARZAN used in this study collected frequency counts of attribute ranges in the different classes. These counts were expressed as the relative measure:

$$\frac{\text{seen in superior class}}{\text{seen in other class}}$$

It was found that, over-riding issues of variations in $costs$ or $chances$, or the combination rules, assumptions about $likesSweat$ greatly controlled the behaviour of the system. $LikesSweat = false$ (is false) appears 3.5 times more frequently in low $cost$, high $chances$ class than in the high $cost$, high $chances$ class. This ratio of 3.5 was the biggest observed difference in the frequency of attribute ranges between the desired and undesired classes. Hence, to argue less, we could just try one what-if query: what-if we set

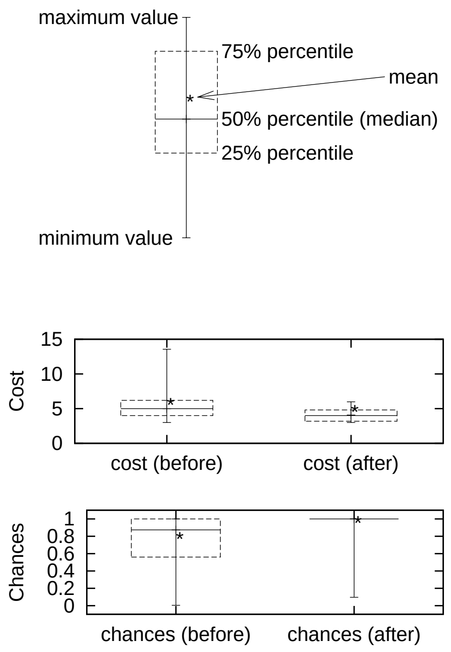

Figure 11: Box plots showing changes in the $cost$s, and $chances$ before and after “what-if likesSweat=false”.

“$likesSweat = false$”? The effects of that what-if query is shown in the box plots of Figure 11. The variance in the $cost$ has been greatly reduced and most of the $chances$ are close to one. (this explains why the the right-hand $chances$ “box” in Figure 11 is squashed flat: the 25% to 75% percentile values are all the same).

6 Conclusion

Stochastic search can quickly find the key assumptions within a space of conflicting design options. Assuming narrow funnels, then:

1. These key assumptions will be few in number;
2. A fantastic reduction in the number of design options can be realized.

Acknowledgments

Partial funding for this work was provided through the NASA IV&V facility by the NASA Office of Safety and Mission Assurance.

References

[Asano and Williamson, 2000] T. Asano and D.P. Williamson. Improved approximation algorithms for max sat. In *SODA ’00: 11th Annual ACM Symposium of Discrete Algorithms*, pages 96–115, 2000. Available from `http://www.almaden.ibm.com/cs/people/dpw/Recent/asano.ps`.

[Buchanan and Shortliffe, 1984] B.G. Buchanan and E.H. Shortliffe. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, chapter 10. Uncertainty and Evidential Support, pages 209–232. Addison Wesley, 1984.

[Crawford and Baker, 1994] J. Crawford and A. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. In *AAAI ’94*, 1994.

[DeKleer, 1986] J. DeKleer. An Assumption-Based TMS. *Artificial Intelligence*, 28:163–196, 1986.

[Gabow *et al.*, 1976] H.N. Gabow, S.N. Maheshwari, and L. Osterweil. On two problems in the generation of program test paths. *IEEE Trans. Software Engrg*, SE-2:227–231, 1976.

[Gordon and Shortliffe, 1985] J. Gordon and E. H. Shortliffe. A method for managing evidential reasoning in a hierarchical hypothesis space. *Artificial Intelligence*, 26(3):323–357, July 1985.

[Josephson *et al.*, 1998] J.R. Josephson, B. Chandrasekaran, M. Carroll, N. Iyer, B. Wasacz, and G. Rizzoni. Exploration of large design spaces: an architecture and preliminary results. In *AAAI ’98*, 1998. Available from `http://www.cis.ohio-state.edu/~jj/Explore.ps`.

[Kakas *et al.*, 1998] A.C. Kakas, R.A. Kowalski, and F. Toni. The role of abduction in logic programming. In C.J. Hogger D.M. Gabbay and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming 5*, pages 235–324. Oxford University Press, 1998.

[Kautz and Selman, 1996] Henry Kautz and Bart Selman. Pushing the envelope: Planning, propositional logic and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 1194–1201, Menlo Park, August 4–8 1996. AAAI Press / MIT Press. Available from `http://www.cc.gatech.edu/~jimmyd/summaries/kautz1996.ps`.

[Lutz and Woodhouse, 1999] R. Lutz and R. Woodhouse. Bidirectional analysis for certification of safety-critical software. In *1st International Software Assurance Certification Conference (ISACC’99)*, 1999. Available from `http://www.cs.iastate.edu/~rlutz/publications/isacc99.ps`.

[Madachy, 1997] R. Madachy. Heuristic risk assessment using cost factors. *IEEE Software*, 14(3):51–59, May 1997.

[Menzies and Compton, 1997] T.J. Menzies and P. Compton. Applications of abduction: Hypothesis testing of neuroendocrinological qualitative compartmental models. *Artificial Intelligence in Medicine*, 10:145–175, 1997. Available from `http://tim.menzies.com/pdf/96aim.pdf`.

[Menzies and Michael, 1999] T. Menzies and C.C. Michael. Fewer slices of pie: Optimising mutation testing via abduction. In *SEKE ’99, June 17-19, Kaiserslautern, Germany. Available from `http://tim.menzies.com/pdf/99seke.pdf`*, 1999.

[Menzies and Sinsel, 2000] T. Menzies and E. Sinsel. Practical large scale what-if queries: Case studies with software risk assessment. In *Proceedings ASE 2000*, 2000. Available from `http://tim.menzies.com/pdf/00ase.pdf`.

[Menzies *et al.*, 1999] T.J. Menzies, S. Easterbrook, Bashar Nuseibeh, and Sam Waugh. An empirical investigation of multiple viewpoint reasoning in requirements engineering. In *RE ’99*, 1999. Available from `http://tim.menzies.com/pdf/99re.pdf`.

[Menzies *et al.*, 2001] T.J. Menzies, R.F. Cohen, S. Waugh, and S. Goss. Applications of abduction: Testing very long qualitative simulations. *IEEE Transactions of Data and Knowledge Engineering (to appear)*, 2001. Available from `http://tim.menzies.com/pdf/97iedge.pdf`.

[Parkes, 1999] A. Parkes. Lifted search engines for satisfiability, 1999.

[Rymon, 1993] Ron Rymon. An SE-tree based characterization of the induction problem. In *International Conference on Machine Learning*, pages 268–275, 1993.

[Rymon, 1994] R. Rymon. An se-tree-based prime implicant generation algorithm. In *Annals of Math. and A.I., special issue on Model-Based Diagnosis*, volume 11, 1994. Available from `http://citeseer.nj.nec.com/193704.html`.

[Simon, 1969] H. Simon. *The Science of the Artificial*. MIT Press, 1969.

[Singer *et al.*, 2000] Josh Singer, Ian P. Gent, and Alan Smaill. Backbone fragility and the local search cost peak. *Journal of Artificial Intelligence Research*, 12:235–270, 2000.

[Steier, 1993] D.M. Steier. CYPRESS-SOAR: A case study in search and learning in algorithm design. In P.S. Rosenbloom, J.E. Laird, and A. Newell, editors, *The SOAR Papers*, volume 1, pages 533–536. MIT Press, 1993.