Flexible Blackbox: Preliminary Results

Peter Jarvis†, Ian Miguel‡, and Qiang Shen‡

†Artificial Intelligence Center
SRI International,
333 Ravenswood Ave, Menlo Park, CA 94025, USA.

‡Division of Informatics
The University of Edinburgh
80 South Bridge, Edinburgh, EH1 1HN, UK

Jarvis@ai.sri.com, {ianm, qiangs}@dai.ed.ac.uk

Abstract

We argued in our Flexible Graphplan (FGP) work that the classical definition of the planning problem is too rigid to capture the full subtlety of many real problems. In light of this, we provided a new *flexible* definition and described a solution strategy based upon the Graphplan framework. Under this definition an action must determine *how well* its preconditions are met and assert a relative satisfaction degree along with its effects. In this paper, we describe how the Blackbox framework can be modified to solve the same flexible problem. Our Flexible Blackbox (FBB) system can synthesise a range of plans for a given flexible problem, trading the compromises made in a plan versus plan length in the same manner as FGP. We detail the modifications required and provide empirical results that compare our implementation of FBB with our FGP solver on a range of flexible problems and against vanilla Blackbox, STAN, IPP, and Graphplan on imperative problems.

Keywords: Flexible Planning Problems, Blackbox, and Graphplan.

Introduction

In our Flexible Graphplan work (Miguel, Jarvis, & Shen 2000), we argued that the classical definition of the planning problem, which casts such problems in terms of imperative constraints that are either wholly satisfied or violated, is too restrictive to capture the full subtlety of real-world problems. We defined a new *flexible* planning problem and described a solution strategy based upon the Graphplan framework (Blum & Furst 1997).

In this paper, we describe our work on adapting the Blackbox framework (Kautz & Selman 1999) to solve the same flexible planning problem. This development is interesting because in contrast to the Dynamic Flexible CSP engine (Miguel & Shen 1999) used in FGP, the plan extraction phase in FBB deploys a standard SAT solver. To achieve this we introduce a new procedure for extracting from a flexible plan graph a structurally valid subgraph with a given minimum satisfaction degree. This is necessary as a flexible plan graph is underpinned by multivalued logic, which is not recognised by standard Boolean SAT solvers. This work provides further insights into the additional effort needed to solve a flexible planning problem and further demonstrates that the recent heuristic advances in planning enable new epistemic benefits to be realised economically, too.

To motivate the need to consider flexibility in planning problems, consider the UM-Translog domain (Andrews *et al*. 1995), derived from Veloso's work (1992), where a valuable package must be carried on an armored truck and the loading and unloading is to be accompanied by a guard. The preconditions of the LOAD-TRUCK action state that (*i*) the truck and the valuable package must be at the same location, (*ii*) the truck must be armored, and (*iii*) a guard is on station. While precondition (*i*) is imperative, preconditions (*ii*) and (*iii*) are preferences or soft constraints (Fox 1987) and can be relaxed with associated reduction in the quality of the resultant plan. It is this ability to reason about the relaxation of soft constraints and the resultant reduction in the quality of to the plan synthesised that we have added to the Graphplan framework and in this paper we add to the Blackbox framework.

The remainder of this paper is structured as follows. The next section describes related work in the areas of contingent, conformant, and mixed-initiative planning and describes how the flexible planning problem is distinct and complementary to these areas. The formal definition of the flexible planning problem from our FGP work is then reprised and a solution strategy using the Blackbox framework is proposed. This strategy is evaluated against FGP on flexible problems and against the vanilla Blackbox system on imperative problems. We finish with a summary and an outline of future work. The reader is assumed to be familiar with the basic operation of Graphplan and Blackbox.

Related Work

The assumptions that an agent possesses complete and correct information has received much attention in the form of contingent and conformant AI planning research. In contingent planning, a planner is equipped with sensing actions that can be used to determine the state of the world during plan execution (Warren 1976; Peot & Smith 1992; Etzioni *et al.* 1992; Goldman & Boddy 1994a; Golden, Etzioni, & Weld 1994; Pryor & Collins 1996). In conformant planning, a planner is provided with knowledge about the possible states the world can be in at the start of execution and the possible outcomes of executing each action (Kushmerick, Hanks, & Weld 1994; Smith & Weld 1998; Blum & Langford 1999). The approaches are complementary and their integration has been explored (Draper, Hanks, & Weld 1994; Blythe & Veloso 1997; Goldman & Boddy 1994b; Onder & Pollack 1999).

The proposed flexible approach is distinct from, but complementary to, contingent and conformant techniques. With respect to contingent approaches, FBB like FGP makes the closed-world assumption and is not furnished with information-gathering actions. However, the flexible propositions used complement the natural imprecision in real-world sensing actions as had been exploited in the area of fuzzy control (Pedecrycz & Gomide 1999).

With respect to conformant approaches one might initially consider overloading a probabilistic formalism to represent the action satisfaction degree property that we introduce later. However, as probability theory and fuzzy logic make different ontological commitments, the inferences made by a probabilistic planner on such an overloaded description will not be sound. To see this consider the finite-horizon Markov Decision Process that underpins PGraphplan (Blum & Langford 1999). The policies synthesised by this system consider repeating an action if its desired effects do not occur during execution. In a flexible context, repeating an action is nonsensical as this will not affect a plan's satisfaction degree. Repeating an action will not change the quality of a plan.

Recent applied planning research explored a mixed-initiative paradigm where the user is supported in exploring the range of plans that can be used to achieve a task. Two approaches have emerged. In TRIPS (Ferguson & Allen 1998) and O-Plan (Tate, Dalton, & Levine 1998), the user is expected to interact extensively with the planner to explore the solution space. This has the weakness that a user may not understand the solution space that he or she is exploring and has the danger that vital solution areas may be missed. The present work is in line with that of Myers and Lee's (1999) development of the SIPE-2 planner (Wilkins 1988) where the system automatically produces a number of courses of action. Their approach is based on the biasing of planner choice points against a meta-theory that relates domain objects and actions to user evaluation criteria. The planner is then run several times on the problem, each time with bias settings from different points in the meta-theory. The proposed approach is similar in that it minimises user-interactions with the system. The important difference is that it is capable of automatically finding a range of plans, given a flexible planning problem, by using a single run.

A Flexible Planning Problem

As defined in (Miguel, Jarvis, & Shen 2000), a flexible planning problem, Ψ, consists of a 4-tuple, $<\Phi, O, I, \Gamma>$, denoting sets of domain individuals, flexible operators, initial conditions consisting of flexible propositions, and flexible goal conditions, respectively. Boolean propositions are herein replaced by flexible propositions, ρ, which have the form $(\rho\ \phi_1, \phi_2, \ldots, \phi_j, \kappa_i)$, where $\phi_i \in \Phi$ and κ_i is an element of a totally ordered set, K, which denotes the subjective degree of truth of the proposition. K is composed of a finite number of membership degrees, $\kappa_\downarrow, \kappa_1, \ldots, \kappa_\uparrow$. The original Boolean proposition type is captured at the end points of K, with $\kappa_\downarrow \in K$ and $\kappa_\uparrow \in K$ indicating total falsehood and total truth, respectively. For brevity, propositions expressing a truth value of $\kappa_\downarrow$ or $\kappa_\uparrow$ will be written in the Boolean style of $\neg(\rho\ \phi_1, \phi_2, \ldots, \phi_j)$ and $(\rho\ \phi_1, \phi_2, \ldots, \phi_j)$, respectively.

A flexible proposition is described by a *fuzzy relation*, R. R is defined by a membership function $\mu(.):\Phi_1 \times \Phi_2 \times \ldots \times \Phi_j \rightarrow K$, where $\mu(.):\Phi_1 \times \Phi_2 \times \ldots \times \Phi_j$ is the cartesian product of the subsets of Φ allowable at this place in the proposition.

As an illustrative example, consider a problem in the logistics domain containing packages pkg_1, pkg_2, and pkg_3. The unary flexible proposition $(valuable\ ?pkg_i\ k_j)$ is represented by a discrete unary fuzzy relation that maps each package to an associated truth degree in K. Assume that the midpoint of K is denoted, $\kappa_\rightarrow$. We can now assert that pkg_1 is worthless with the flexible proposition $(valuable\ pkg_1\ \kappa_\downarrow)$, that pkg_2 is moderately valuable with $(valuable\ pk_2\ \kappa_\rightarrow)$, and that pkg_3 is very valuable with $(valuable\ pkg_3\ \kappa_\uparrow)$.

A flexible operator, $o \in O$, cannot simply test for the presence of a consistent set of preconditions; it must recognise *how well* its preconditions are satisfied. Flexible operators are also described by fuzzy relations that map from the precondition space to a totally ordered *satisfaction scale*, *L*, as well as a set of flexible effects propositions. As per *K*, *L* is composed of a finite number of membership degrees, $l_\downarrow, l_1, \ldots, l_\uparrow$. The endpoints, $l_\downarrow, \in L$, and, $l_\uparrow. \in L$, respectively denote a complete lack of satisfaction (in which case the operator is not to be added to the planning graph based on these preconditions) and complete satisfaction.

A flexible operator consists of a set of disjoint conditional clauses, Σ (similar to a traditional STRIPS operator (Fikes & Nilsson 1971) enhanced to support conditional effects (Pednault 1989)). Each $\sigma \in \Sigma$ is a triple $<\Theta, R, \kappa_i>$ denoting, respectively, a conjunction of flexible preconditions, a conjunction of flexible effect propositions, and the associated satisfaction degree of this operator given these preconditions. Each $\theta \in \Theta$ has the form $(\rho\ \phi_1, \phi_2, \ldots, \phi_j\ \tau\ \kappa_i)$, where τ is a precondition operator with argument set κ taken from those shown in Table 1. The general format of a flexible operator is shown in Figure 1. Each σ_i maps a subset of the space of preconditions to a particular set of effects and a satisfaction degree in L.

A flexible plan goal $\gamma \in \Gamma$ maps from the space of flexible propositions to L. Each goal is defined using a number of clauses, as shown in Figure 2. The preconditions are defined exactly as those used in the flexible operators. More than one set of mutually consistent propositions may exist that satisfy the plan goals to some extent. Hence, the satisfaction degree of the plan as a whole must take into account the goal satisfaction degrees as well as those of the flexible operators.

Precondition Operator, τ	Arguments, κ
$\equiv$	κ_i
<=	κ_i
>=	κ_i
Range	$\kappa_i\ \kappa_j$
Discrete	$\kappa_a\ \kappa_b \ldots$

Table 1: Operators Used to Defined Flexible Preconditions

(**operator** o_I (**params** $param_1$, $param_2$, …)
σ_i:{**when** (**preconds** θ_{i1}, θ_{i2}, …)
(**effects** ρ_{i1}, ρ_{i2} …) (**satisfaction** l_i)}
σ_j:{**when** (**preconds** θ_{i1}, θ_{i2}, …)
(**effects** ρ_{i1}, ρ_{i2} …) (**satisfaction** l_j)}

Figure 1: The General Format of a Flexible Operator

(**goal** γ_i
{**when** (θ_i) (**satisfaction** l_i)}
{**when** (θ_j) (**satisfaction** l_j)}

Figure 2: The General Format of a Flexible Goal

The satisfaction degree of a flexible plan is defined as the conjunctive combination of the satisfaction degrees of each flexible operator and each flexible goal used in the plan. The conjunctive combination of two fuzzy relations, $R_i \otimes R_j$, is usually interpreted as the minimum membership value assigned by either relation, although other interpretations of combination are possible (Smith & Shen 1997). The *quality* of a plan is defined in terms of its satisfaction degree combined with its length. Given two plans with an equivalent satisfaction degree, the shorter is deemed to be the better.

Flexible Blackbox

The original Blackbox framework (Kautz & Selman 1999) operates in four stages:

1. A planning problem is converted to a plan graph by using the same procedure (and code) as in Graphplan. When the sufficient conditions for plan existence are met, the system moves to step 2.
2. Translation rules are used to transform the planning graph into a CNF wff suitable for input to a SAT engine.
3. A SAT engine is applied to the CNF wff.
4. If the SAT engine finds a model then that model corresponds to a plan, which is recovered from the model, and the system terminates. If no model can be found, then the system returns to step 1.

To adapt this framework to solve the flexible planning problem, a new flexible graph expansion procedure is used in step 1 and a new subgraph extraction procedure must be added to step 2. These modifications are described in the following sections.

Flexible Graph Expansion

This process (and the code) is identical to that used by Flexible Graphplan. To describe the process, it is first necessary to define exclusivity in the context of flexible propositions. Two flexible propositions are labelled as exclusive if either they express a different truth degree for the same core proposition or (as per Graphplan) all ways of creating one are exclusive of all ways of creating the other.

Mutual-exclusion constraints between action nodes are again defined for FGP in a manner similar to that of the original Graphplan framework. They express the information that no valid plan could possibly contain both actions. Two flexible actions are mutually exclusive if any of the following hold:

- *Inconsistent Effects*: The actions have mutually exclusive effects.
- *Interference*: One action has an effect proposition that expresses a different truth degree than for a proposition required as the precondition of the other.
- *Competing Needs*: The actions have mutually exclusive preconditions.

All initial conditions are placed in the first proposition level of the graph (which will be referred to as level 0). A generic action level is generated as follows. Each clause of each flexible operator is instantiated in all possible ways to propositions of the previous level. If all preconditions are mutually consistent, an action instance with the associated satisfaction is added to the planning graph for each such instantiation. No-op actions are again added in exactly the same manner as for Graphplan. The No-op action is a special case, which always has a satisfaction of $l_\uparrow$.

The graph expansion process can be improved for FBB to ease the burden of maintaining flexible information. Consider the case where a plan of satisfaction degree l_i has been found. If $l_i < l_\uparrow$ the expansion process must continue onward from this point to look for a plan with a higher satisfaction degree. However, it can be seen that there is no point in instantiating flexible operator clauses with a satisfaction degree less than or equal to l_i: a plan with this satisfaction degree has been found already a longer plan with the same satisfaction degree is deemed to be of a lower quality. The conjunctive combination rule ensures that no plan of satisfaction degree l_j can contain an action of satisfaction l_i, where $l_i < l_j$; hence, the completeness of the search is not affected by omitting such actions in future flexible planning graph levels.

An Example

For illustration, consider the following example derived from the logistics domain. The example contains three cities connected by three roads. *$Road_1$* and *$Road_2$* are major roads and connect *$City_1$* to *$City_2$* and *$City_2$* to *$City_3$* respectively. *$Road_3$* is an unsafe track through the hills and connects *$City_1$* to *$City_3$*. The single imperative goal of this problem is to transport a package, *Pkg_1*. from *$City_1$* to *$City_3$*. The following definitions are used for K and L: $K = \{\kappa_\downarrow, \kappa_1, \kappa_2, \kappa_\uparrow\}$, $L = \{l_\downarrow, l_1, l_2, l_\uparrow\}$.

Figure 3 shows an operator which expresses the damage done to a plan by sending a truck across the dangerous dirt track whilst indicating that using the major roads has no damaging effect on the plan. Although in this simple illustrative case all effects sets within the operator are the same this is not a requirement and in general they may be totally different. Other inflexible operators are defined (by not using any of the flexible preconditions and assigning a satisfaction of $l_\uparrow$) that allow for packages to be unloaded and unloaded from a truck.

```
(operator DRIVE
   (params (?v vehicle) (?o location) (?d location)
              (?r1 major-road) (?r2 track))
   {(when (preconds (at ?v ?o) (connects ?r1 ?o ?d))
      (effects (not (at ?v ?o)) (at ?v ?d))
      (satisfaction l↑)}
   {(when (preconds (at ?v ?o) (connects ?r2 ?o ?d))
      (effects (not (at ?v ?o)) (at ?v ?d))
      (satisfaction l1)}
```

Figure 3: Flexible Operator Drive

Given the flexible Drive operator and the problem of moving a package from *$City_1$* to *$City_3$*, the flexible graph expansion will create a graph containing a plan at a satisfaction degree of l_1 after two steps and a plan at $l_\uparrow$ after three steps. The l_1 plan will use the unsafe but direct *$Road_3$* while the $l_\uparrow$ plan will take the longer but safer route of provided by *$Road_1$* and *$Road_2$*. This example illustrates the trade-off possible between the plan satisfaction degree and plan length.

Translating a Flexible Planning Graph to a CNF wff

Under the classical definition of planning that underpins plan extraction in Graphplan, and therefore Blackbox, all actions in a planning graph have the same implied degree of satisfaction and each must be considered as a candidate during plan extraction. Under the flexible definition used in FGP and FBB, in searching for a plan at a given satisfaction degree only actions that possess an individual satisfaction degree that is greater than or equal to this degree can be considered as candidates. This is a direct result of our interpretation of the conjunctive combination of two fuzzy relations, $R_i \otimes R_j$, as being the minimum membership value assigned by either relation.

As standard SAT solvers have no facility for associating a satisfaction degree with a literal in a formula, the obligation of providing a SAT solver with a wff that can lead only to a model with a given minimum satisfaction degree rests with the translation process. To achieve a CNF wff that can lead only to a plan with a given minimum satisfaction degree, we need a procedure for extracting from a flexible planning graph a *structurally-valid* subgraph containing only actions at or above the current degree of satisfaction for which we are searching. We define the procedure for building such a subgraph in three stages.

In Definition 1 we define the membership of a *candidate-subgraph* at a given degree of satisfaction. This set contains only actions with a satisfaction degree equal to or higher than the degree at which we are searching. However, it does not guarantee that the graph formed by its members will be structurally valid. Definitions 2 and 3 achieve this condition. Definition 1 assumes that the set *flexible-graph* contains the plan graph produced by the flexible plan expansion process defined in the previous section. The function *satisfaction-degree: action* $\rightarrow l_i \in K$ returns the satisfaction degree associated with an action.

Definition 1 (membership of candidate-subgraph)
$\forall a_j$: $a_i \in$ flexible-graph, $a_j \in$ candidate-subgraph$(l_i) \Leftrightarrow$ satisfaction-degree$(a_j) \geq l_i$.

Under Definition 1, the candidate-subgraph may not be structurally valid. By this we mean that it may contain actions that depend upon propositions that are not asserted by fellow members of the candidate-subgraph. In Definitions 2 and 3 we provide the necessary conditions for determining the subset of the candidate-subgraph that is also structurally valid (sv).

Definition 2: (membership of sv-subgraph)
$\forall a_i$: $a_i \in$ candidate-subgraph(l_i), $a_i \in$ sv-subgraph$(l_i) \Leftrightarrow$ supported(a_i, l_i).

The predicate *supported* defined in Definition 3 assumes the existence of the following utility functions and constants. Function *graph-level: action* $\rightarrow \mathbb{Z}^+$ provides the planning graph level occupied by an action. The set *initial-state* contains all propositions in the initial state, and the constant *first-action-level* is equal to the level in the planning graph occupied by the first action level, that is the level supported by the propositions in the initial state.

Definition 3: (membership of set supported)
supported$(a_i, l_i) \Leftrightarrow (\forall p$: precondition$(p, a_i) \Rightarrow$
$((\exists a_j$: asserts$(a_j, p) \wedge a_j \in$ candidate-subgraph$(l_i) \wedge$
(graph-level$(a_j) \equiv$ (graph-level(a_i) -1)) $\wedge$
supported(a_j, l_i)) $\vee$ ((graph-level$(a_i) \equiv$
first-action-level) $\wedge$ $p \in$ initial-state)))) .

Definitions 1 through 3 are sufficient to provide us with a structurally valid subgraph for a given degree of satisfaction. If, when converted to a CNF wff, a SAT solver can find a model for such a subgraph then a plan exists with this or a higher degree of satisfaction. The rules for translating a structurally valid subgraph to a CNF wff are the same as those given by Kautz, McAllester & Selman (1996).

1. The Initial state holds at layer 1, and the goals hold at the highest layer of the graph.
2. Operators imply their preconditions, for example,
   $(DRIVE(Truck_1, City_1, City_3, l_i, t_1) \Rightarrow ((at, truck_1, City_1, \kappa_\uparrow, t_0) \wedge (connects, R_2, City_1, City_2, \kappa_\uparrow, t_0))$
3. Each fact at graph level *a* implies the disjunction of all the operators at level a-1 that have it as an add-effect, for example,
   $(at, Truck_1, City_3, \kappa_\uparrow, t_2) \Rightarrow (no\text{-}op(at, Truck_1, City_3, l_i, t_1), t_2) \vee (DRIVE(truck_1, City_1, City_3, l_i, t_2))$
4. Conflicting actions are mutually exclusive.
   $((\neg DRIVE(truck_1, City_1, City_3, l_i, t_1) \vee (\neg DRIVE(truck_1, City_1, City_2, l_\uparrow, t_1))$

In summary, with the addition of a procedure for extracting a valid subgraph at a given degree of satisfaction, the modifications to Blackbox's plan extraction procedure are complete. The simplicity of the modifications required reflects the elegance of the Graphplan and Blackbox approaches with respect to separating graph expansion from plan extraction.

Experimental Results

FBB is first evaluated against leading Boolean solvers to establish the effect of the additional flexible machinery. Table 2 shows the results obtained when running FBB, FGP, Graphplan, STAN v4 (Long & Fox 1998), IPP v4 (Koehler *et al.* 1997), and BLACKBOX 3.6b (Kautz & Selman 1999). FBB performs strongly against all the Graphplan planners, including FGP. FGP and FBB are implemented in Java™ and the other solvers in C. The SAT solver used by FBB is implemented in C.

Problem	Len.	FBB	FGP	BBox	GP	STAN	IPP
Rocket-a	7	36	14	5	75	33	47
Rocket-b	7	54	21	8	154	2	76
Log-a	11	63	8	6	1955	1	1513
Log-b	13	175	123	9	862	2	633
Log-c	13	392	145	20	-	704	_

Table 2: Boolean Problem Comparison (in seconds) between FBB, FGP and Leading Solvers. A dash indicates no solution found in 24 hours. Hardware used: Sun Ultra 5.

On this problem set FBB's performance compares well with the Graphplan based solvers, including FGP. However, against vanilla Blackbox the performance is disappointing. Profiling of the additional flexible machinery indicated that this was contributing approximately one second to the runtime of FBB on Log-c. The bulk of the performance differential can be explained by implementation differences, particularly in the use of Java. We can conclude that on Boolean problems the additional flexible machinery does not significantly degrade performance.

Problem	L	Len	FBB	FGP
Flogs-1	l_1	3	2	< 1
	l_2	4	4	< 1
	$l_\uparrow$	9	14	3
Flogs-2	l_1	3	2	< 1
	l_2	4	3	1
	$l_\uparrow$	12	26	4
Flogs-3	l_1	3	2	< 1
	l_2	4	4	1
	$l_\uparrow$	20	172	12
Flogs-4	l_1	3	2	< 1
	l_2	6	7	2
	$l_\uparrow$	20	433	17
Flogs-5	l_1	3	2	1
	l_2	7	21	5
	$l_\uparrow$	20	816	26
Flogs-6	l_1	5	5	2
	l_2	8	14	5
	$l_\uparrow$	20	701	27
Flogs-7	l_1	5	5	2
	l_2	10	45	9
	$l_\uparrow$	20	1149	33
Flogs-8	l_1	5	5	2
	l_2	12	175	16
	$l_\uparrow$	20	1687	40
Flogs-9	l_1	7	13	5
	l_2	12	177	17
	$l_\uparrow$	20	1687	42
Flogs-10	l_1	7	16	6
	l_2	15	907	42
	$l_\uparrow$	20	3567	65
Flogs-11	l_1	9	41	11
	l_2	15	848	40
	$l_\uparrow$	20	3858	66
Flogs-12	l_1	12	114	20
	l_2	17	1589	50
	$l_\uparrow$	20	3708	67

Table 3: Flexible Problems (in seconds). Hardware used: Sun Ultra 1.

In the second experiment, the performance of FBB is compared with that of FGP on a range of flexible problems. The domain used was a variant of the flexible logistics domain introduced earlier. The test suite contained 12 problems designed to cover a range of variations in plan length and plan quality. Early in the problem set, solutions at each quality are both short and similar in length. The solution length at each satisfaction degree is then raised in turn to provide cases where the highest quality plan is significantly longer than the lower quality plans, and so forth. At the end of the test set the plans are again of a similar but longer length. The results of this experiment are shown in Table 3.

FBB's performance is poor when compared with FGP's, which is contradictory to the Boolean results reported by others. Two areas can be examined to identify the source of this performance degradation. First, do the larger graphs generated by a flexible planner exceed some threshold beyond which the translation of a planning graph to a CNF wff is more expensive than the relative performance advantage of a SAT solver in the plan extraction process? Second, is the implementation of FBB tested here inefficient? Profiling FBB's execution on Log-c revealed the following apportionment of time between the solver's subprocesses:

- 2% graph construction
- 2% SAT solver
- 0.78% determining membership of the candidate subgraphs
- 6.22% determining membership of the structurally valid subgraphs
- 95% translating plan graphs to and from CNF

As graph translation should be polynomial with respect to the size of a planning graph and given that it is taking 95% of the computational effort, the efficiency of the implemented algorithm should be considered.

It is an open question as to whether the larger graphs produced by a flexible planner favor the SAT-based or CSP-based plan extraction strategy.

Conclusion

We have shown that, with relatively simple modifications, the Blackbox framework can be adapted to solve the *flexible* planning problem. This is an interesting development as the structurally valid subgraph procedure that had to be developed to enable standard Boolean SAT solvers to be exploited identifies clearly the additional work needed to extract a plan from a flexible planning graph.

The experimental results available are inconclusive. Only by carefully tuning the CNF wff translation algorithm within FBB will it be possible to determine if the larger plan graphs generated by flexible planners favor SAT based or CSP based plan extraction strategy.

This paper completes the first item of further work identified in our FGP work reported earlier and it provides a new solution strategy for the flexible planning problem. It would be interesting to develop flexible planning in two directions. First, the inclusion of the efficient plan graph ideas from STAN (Long & Fox 1998) and IPP (Koehler *et al*. 1997) would ease the burden of building and maintaining the large plan graphs demanded by the flexible approach. Second, it would be interesting to compare the range of solutions generated by both flexible solvers with those produced by SIPE-2 under Myers and Lee's (1999) developments.

Acknowledgments

Peter Jarvis is supported by DARPA Contract F30602-97-C-0067 under the supervision of Airforce Research Laboratory - Rome. Ian Miguel is supported by the UK-EPSRC grant number 97305803. The Authors thank Karen L. Myers for insightful comments.

References

Andrews, S., Kettler, B., Erol, K., and Hendler, J. 1995. UM Translog: A planning domain for the development of benchmarking of planning systems. Technical Report, Dept. Computer Science, University of Maryland, USA.
Blum, A., and Furst, M. 1997. Fast planning through planning graph analysis. Artificial Intelligence 90(1-2):281-300.

Blum, A., and Langford, J. 1999. Probabilistic planning in the Graphplan framework. Proceedings of the 5th European Conference on Planning Systems, Durham UK 320-332.

Blythe, J., and Veloso, M. 1997. Analogical replay for efficient conditional planning. Proceedings of the 15th National Conference on AI 668-673.

Draper, D., Hanks, S., and Weld, D. 1994. Probabilistic planning with information gathering and contingent execution. Proceedings of the 2nd International Conference on AI Planning Systems 31-36.

Etzioni, O., Hanks, S., Weld, D., Draper, D., Lesh, N., and Williamson, M. 1992. An approach to planning with incomplete information. Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning 115-125.

Ferguson, G., and Allen, J. 1998. TRIPS: Towards a mixed-initiative planning assistant. Proceedings of AIPS Workshop on Interactive Collaborative Planning.

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. Artificial Intelligence 5(2).

Fox, M. 1987. Constraint-Directed Serach: A Case Study of Job-Shop Scheduling. Pitman, Morgan Kaufmann.

Golden, K., Etzioni, O., and Weld, D. 1994. Omnipotence without omniscience: Efficient sensor management for planning. Proceedings of the 12th National Conference on AI 1048-1054.

Goldman, R., and Boddy, M. 1994a. Conditional linear planning. Proceedings of the 2nd International Conference on AI Planning Systems 80-85.

Goldman. R., and Boody, M. 1994b. Epsilon-safe planning. Proceedings of the 10th Conference on Uncertainty in AI 253-261.

Kautz, H., and Selman, B. 1999. Unify Sat-based and graph-based planning. Proceedings of the 16th International Joint Conference on AI 318-325.

Kautz, H., McAllester, D., and Selman, B. 1996. Encoding plans in prepositional logic. Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning.

Koehler, J., Nebel, B., Hoffmann, J., and Dimopoulos, Y. 1997. Extending planning graphs to an ADL subset. European Conference on Planning 273-285.

Kushmerick, N., Hanks, S., and Weld, D. 1995. An algorithm for probabilistic planning. Artificial Intelligence 76(1-2):239-286.

Long, D., and Fox, M. 1998. Efficient implementation of the plan graph in STAN. Journal of AI Research 10:87-115.

Miguel, I., and Shen, Q. 1999. Extending FCSP to support dynamically changing problems. Proceedings of the 8th International Conference on Fuzzy Systems 1615-1620

Miguel, I., Jarvis, P., and Shen, Q. 2000. Flexible Graphplan. In Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000), Berlin, Germany.

Myers, K., and Lee, T. 1999. Generating qualitatively different plans through metatheoretic biases. Proceedings of the 16th National Conference on AI 570-576.

Onder, N., and Pollack, M. 1999. Conditional, probabilistic planning. Proceedings of the 16th National Conference on AI 577-584.

Pednault, E. 1989. ADL: Exploring the middle ground between STRIPS and the situation calculus. Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning.

Pedrycz, W., and Gomide, F. 1999. An Introduction to Fuzzy Sets: Analysis and Design. MIT Press.

Peot, M., and Smith, D. 1992. Conditional non-linear planning. Proceedings of the 1st International Conference on AI Planning Systems 189-197.

Pryor, L., and Collins, G. 1996. Planning for contingencies: A decision based approach. Journal of AI Research 4:287-339.

Smith, F., and Shen, Q. 1997. Choosing the right fuzzy logic controller. Proceedings of the 7th International Fuzzy Systems Association World Congress 3:342-347.

Smith, D., and Weld, D. 1998. Conformant Graphplan. Proceedings of the 15th National Conference on AI.

Tate, A., Dalton, J., and Levine, J. 1998. Generation of multiple qualitatively different plans. Proceedings of the 4th International Conference on AI Planning Systems 27-34.

Veloso, M. 1992. Learning by analogical reasoning in general problem solving. Ph.D. Dissertation, Carnigie Mellon University, USA.

Warren, D. 1976. Generating conditional plans and programs. Proceedings AISB Summer Conference 344-354.

Wilkins, D. 1988. Practical Planning. Morgan Kaufmann.