Flexible Graphplan

Ian Miguel[1], **Peter Jarvis**[2], **Qiang Shen**[1]

Abstract. Traditionally, planning problems are cast in terms of imperative constraints that are either wholly satisfied or wholly violated. It is argued herein that this framework is too rigid to capture the full subtlety of many real problems. A new flexible planning problem is defined which supports the soft constraints often found in reality. A solution strategy using the Graphplan framework is described and it is shown how flexible plan extraction can be cast as the solution of a sequence of linked Dynamic Flexible Constraint Satisfaction Problems (DFCSPs). A recently developed DFCSP algorithm, Flexible Local Changes, is exploited to solve this sequence. For a given flexible problem, this framework can synthesise a range of plans that trade the compromises made in a plan versus plan length. The proposed technique is evaluated on a range of flexible problems and against leading boolean solvers on benchmark problems.

1 Introduction

In the Constraint Satisfaction (CSP) community it has become increasingly clear that the classical CSP definition is insufficient to capture the full subtlety of many real world problems. In particular, classical hard constraints (which are imperative and are either fully satisfied or fully violated) prove too restrictive. Recent approaches extend classical CSP to create flexible constraint satisfaction techniques [6, 15] that support the soft constraints often found in reality. This paper explores the benefits of applying the same approach to planning problems, many of which exhibit the need for soft constraints. Consider the UM-Translog domain [2] where a valuable package must be carried on an armoured truck and loading/unloading must be accompanied by a guard. The preconditions of the `LOAD-TRUCK` action state that (i) the truck and the package must be co-located, (ii) the truck must be armoured, and (iii) that a guard must be present. Precondition (i) is imperative, but preconditions (ii) and (iii) are soft constraints [15] and can be relaxed with an associated damage to the resultant plan. The *flexible planning problem*, described herein, is a framework for reasoning about the relaxation of soft constraints and the resultant damage to the plan synthesised.

In order to create a solution technique for flexible planning problems, the recent paradigm of casting planning itself as a classical CSP in the Graphplan framework [3] is exploited. This framework is extended to support flexible planning via the utilisation of dynamic-flexible CSP (DFCSP) [16] methods. The reader is assumed to be familiar with the basic operation of Graphplan.

[1] Div. of Informatics, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, UK, email: {ianm,qiangs}@dai.ed.ac.uk
[2] Artificial Intelligence Center, SRI International, 333 Ravenswood Ave. Menlo Park, CA 94024, USA, email: jarvis@ai.sri.com

© 2000 I. Miguel, P. Jarvis and Q. Shen
ECAI 2000. *14th European Conference on Artificial Intelligence*
Edited by Werner Horn
Published in 2000 by IOS Press

2 Background

The assumption that an agent possesses complete and correct information has received much attention in contingent and conformant AI planning research. In the former, a planner is equipped with sensing actions that can be used to determine the state of the world during plan execution [1, 22]. In the latter, a planner is provided with knowledge about the possible initial states the world can be and the possible outcomes of executing each action [13, 4]. These approaches are complementary and their integration has been explored [5].

The proposed flexible approach is distinct from but complementary to contingent and conformant techniques. With respect to contingent approaches, 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 has been exploited in the area of fuzzy control [21]. Conformant planners model uncertainty as a distribution over the effects of a given operator yet the propositions themselves are exclusively true or false. Within the proposed approach, flexible propositions are assigned subjective truth degrees and operators map deterministically from the space of flexible preconditions to a set of flexible effects and a satisfaction degree. Therefore, subjective knowledge can be utilised to entail inferences over preferred combinations of actions.

It might be conjectured that a probabilistic formalism could be overloaded so that probabilities attached to effects signify operator satisfaction degrees. Invoking a probabilistic planner on such an overloaded domain description highlights the different semantics of satisfaction degrees and probabilities. For example, PGraphplan [4] produces a contingent plan that recommends repeated application of an operator in event of it not producing the desired effects on its first application. This is nonsense in the flexible context as repeating an action will not affect its satisfaction degree. While a probabilistic planner will not correctly solve flexible planning problems, the approaches are complementary and could be combined.

A mixed-initiative paradigm has been explored where the user is supported in exploring the range of plans that can be used to achieve a task. In TRIPS [7] and O-Plan [24], the user is expected to interact extensively with the planner to explore the solution space. However, a user may not understand the solution space and vital solution areas may be missed. The present work is in line with that of Myers and Lee's [19] development of the SIPE planner [26] where the system automatically produces a number of courses of action. The important difference is that FGP can automatically find a range of plans given a flexible planning problem using a single run, as opposed to the multiple runs required by SIPE.

Other recent work [11, 25] has incorporated numerically weighted constraints into planning to give a quantitative means of differentiating between different potential plans, as opposed to the qualitative methods in this paper. However, if there is actual numerical knowledge available regarding the truth and/or satisfaction degrees, the present approach can be readily adapted to make use of such knowledge without changing the underlying representation mechanism. It may be beneficial to investigate the combination of both approaches, though this remains an important item of future work.

CSP approaches used for plan extraction in the Graphplan framework [9] generally utilise conditional CSP techniques [17] which consider the plan extraction process as a single problem whose structure changes based on existing (tentative) assignments. FGP takes the novel approach of using the dynamic-flexible CSP definition proposed in [16]. DFCSPs support changes to the problem structure via the addition and/or removal of flexible constraints and are used herein to exploit the layered structure of the planning graph.

3 A Flexible Planning Problem

A flexible planning problem, Ψ, consists of a 4-tuple, $<$ Φ, O, I, Γ $>$, denoting sets of plan objects, flexible operators, initial conditions consisting of flexible propositions, and flexible goal conditions, respectively.

Boolean propositions are herein replaced by flexible propositions, ρ, of the form $(\rho\ \phi_1, \phi_2, ...\phi_j\ k_i)$, where $\phi_i \in \Phi$ and k_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, $k_\perp, k_1, ..., k_\top$. Boolean propositions are captured at the end points of K, with $k_\perp \in K$ and $k_\top \in K$ indicating total falsehood and total truth respectively. When dealing with propositions which only ever take a truth value of $k_\perp$ or $k_\top$, the boolean style of $\neg(\rho\ \phi_1, \phi_2, ..., \phi_j)$ and $(\rho\ \phi_1, \phi_2, ..., \phi_j)$ is adopted respectively. A flexible proposition is described by a *fuzzy relation* [21], R, which is defined by a membership function $\mu_R(.) : \Phi_1 \times \Phi_2 \times ... \times \Phi_j \rightarrow K$, where $\Phi_1 \times \Phi_2 \times ... \times \Phi_j$ is the cartesian product of the subsets of Φ allowable at this place in the proposition.

(**operator** o
 (**params** $param_1$, $param_2$, ...)
 σ_i: {**when** (**preconds** θ_{i_1} θ_{i_2} ...)
 (**effects** ρ_{i_1} ρ_{i_2} ...)
 (**satisfaction** l_i)} *etc*

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

Figure 1. General Formats of Flexible Operators and Goals

A flexible operator, $o \in$ O, must recognise *how well* its preconditions are satisfied. Flexible operators are described by fuzzy relations which map from the precondition space to a totally ordered *satisfaction scale*, L and a set of flexible effects propositions. L is also composed of a finite number of membership degrees, $l_\perp, l_1, ..., l_\top$. The endpoints, $l_\perp \in L$ and $l_\top \in L$ respectively denote a complete lack of satisfaction (in which case the operator is not added to the planning graph based on these preconditions) and complete satisfaction. The Noop action is a special case which has a satisfaction of $l_\top$.

A flexible operator (figure 1) consists of a set of disjoint conditional clauses, Σ (similar to conditional effects [20]). Each $\sigma \in \Sigma$ is a triple $<$ Θ, R, k_i $>$ respectively denoting a conjunction of flexible preconditions, a conjunction of flexible effect propositions and the satisfaction degree of this operator given these preconditions. Each $\theta \in \Theta$ has the form $(\rho\ \phi_1, \phi_2, ...\phi_j\ \tau\ \kappa\ )$, where τ is a precondition operator with argument set κ. The allowed precondition operators encompass equality, inequality, ranges of truth degrees and sets of discrete truth degrees. 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 1. Preconditions are defined exactly as those used in the flexible operators. More than one set of mutually-consistent propositions may exist which 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.

The satisfaction degree of a flexible plan is defined as the conjunctive combination of the satisfaction degrees of each operator and each 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. The *min* operator has the desirable property of idempotency $(\forall u, u \oplus u = u)$ which enables classical CSP k-consistency enforcing techniques to be straightforwardly extended to support flexible CSP [6] and is central to the computational efficiency of FGP. The *quality* of a plan is its satisfaction degree combined with its length, where the shorter of two plans with equivalent satisfaction degrees is better.

4 Flexible Graph Expansion

Exclusivity is first defined 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 express that no valid plan could 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 which expresses a different truth degree than for a proposition required as the precondition of the other; *Competing Needs*: The actions have mutually exclusive preconditions.

Initial conditions are placed in the first proposition layer of the graph (layer$_0$). A generic action layer is generated as follows. Each clause of each flexible operator is instantiated in all possible ways to propositions of the previous layer. Provided that all preconditions are mutually consistent, an action instance with the associated satisfaction is added to the planning graph for each such instantiation. Noop actions are added in the same manner as for Graphplan.

4.1 Limited Flexible Graph Expansion

The cost of graph expansion can be reduced as follows. Consider the case where a plan of satisfaction degree l_i has been found. If $l_i < l_\top$ FGP searches onwards in order to look for a plan with a higher satisfaction degree. However, there is no point in instantiating flexible operator clauses with a satisfaction degree less 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 implemented via the *min* operator 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 layers.

4.2 An Example

An illustrative example follows, derived from the logistics domain (see figure 2). The c_i are three cities, r_1 and r_2 are major roads and r_3 is a very unsafe track through the hills. The single boolean goal of this problem is to transport pkg_1 to c_3. The following definitions are used for K and L: $K = \{k_\perp, k_1, k_2, k_\top\}$, $L = \{l_\perp, l_1, l_2, l_\top\}$.

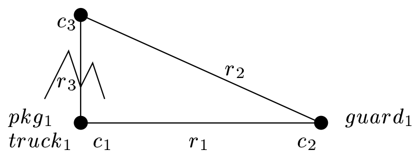


Figure 2. An Example Flexible Problem

Figure 3 shows the flexible operator `Load-Truck` which considers the value of the package and the presence of a guard. If the package is not very valuable (according to the truth degree in K of the proposition concerning the value of the package), the guard's presence makes no difference (a satisfaction degree of $l_\top$ is assigned). If the package is quite valuable, loading it onto the truck without a guard damages the plan, resulting in a lower satisfaction degree, l_2. Figure 3 also 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.

```
(operator LOAD-TRUCK
 (params (?t truck) (?p package) (?l location)
         (?g guard))
 {when (preconds (at ?t ?l) (at ?p ?l) (valuable ?p <= k1))
  (effects (on ?p ?t) (at ?p ?l))
  (satisfaction l⊤)}
 {when (preconds (at ?t ?l) (at ?p ?l) (on ?g ?t)
         (valuable ?p >= k2))
  (effects (on ?p ?t) (at ?p ?l))
  (satisfaction l⊤)}
 {when (preconds (at ?t ?l) (at ?p ?l) (not (on ?g ?t))
         (valuable ?p >= k2))
  (effects (on ?p ?t) (at ?p ?l))
  (satisfaction l2)})

(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. The Flexible Operators `Load-Truck` and `Drive`

Although in this simple case all sets of effects within the two flexible operators described are the same this is not a requirement and in general they may be different. Other inflexible operators are defined (i.e. using inflexible preconditions and assigning a satisfaction of $l_\top$) which allow the guard to get on the truck and for packages to be unloaded.

Figure 4 shows the initial conditions and first three layers of the flexible planning graph generated from this problem definition. Only propositions and actions (annotated with truth and satisfaction degrees respectively) sufficient to synthesise a short plan with a low satisfaction degree (l_1) are shown: the truck is loaded with the quite valuable package without the presence of a guard and used to carry it to the destination city via the unsafe track. By using the main roads to avoid the unsafe track, the package may be carried to c_3 in 4 steps - but still the plan is not regarded as having the highest satisfaction degree because the guard was not present when the package was loaded. The 'no compromise' plan uses the truck to fetch the guard from c_2 before returning to load the package. The package is then transported to c_3 via the major roads, resulting in a plan length of 7 steps.

5 Flexible Plan Extraction

Plan extraction from a flexible planning graph may be viewed as solving a CSP. Propositions are regarded as CSP variables whose domain elements are the actions which produce the associated proposition as an effect, as per Graphplan [3]. In addition, unary *preference* constraints [15] are constructed from the satisfaction degrees associated with each action, specifying a preference degree for each potential assignment. Boolean binary constraints are generated from the mutual exclusion relations in the planning graph.

Each variable instantiation represents the choice of an action node to support the proposition node represented by the variable. The preconditions of each chosen action must then also be supported, and so on. If solution extraction is viewed as solving a single large problem, the fact that every variable instantiation causes a change to the problem structure makes it very unstable. An alternative method is proposed where each layer is treated as a single (dynamic) sub-problem to be solved. Solving a sub-problem instance at layer$_i$ defines a sub-problem instance at layer$_{i-1}$ (figure 5). Multiple solutions define a problem sequence at layer$_{i-1}$ which can be seen as a DFCSP that can be efficiently solved using the Flexible Local Changes (FLC) algorithm [16].

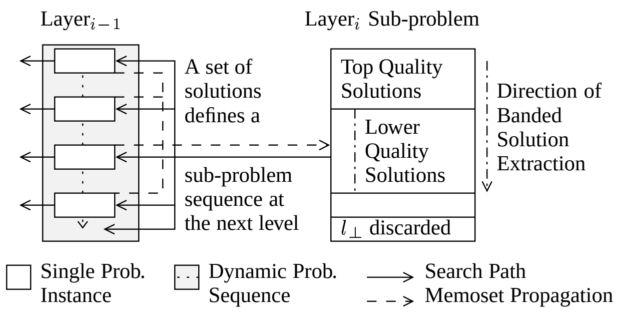


Figure 5. Partitioning of the Flexible Planning Graph.

If a sub-problem instance is completely insoluble, even taking flexibility into account, backtracking is necessary to the layer above

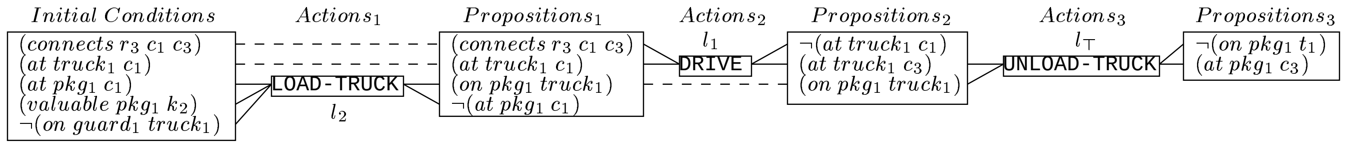


Figure 4. Partial Flexible Planning Graph for the Problem of Figure 2. Note that r_3 is the unsafe track.

in order to find the next solution. A branch and bound technique is used to control the search for the set of problems which may be optimally solved conjunctively. Plan extraction starts when a set of mutually-consistent flexible propositions exist in the current layer which, to some extent, satisfy the flexible plan goals. All goal combinations that might lead to a better plan than is currently known must be tested. Since it is desirable to find plans with a higher satisfaction degree first, a 'banded' solution extraction procedure has been adopted (figure 5). The FLC algorithm concentrates on a particular band of solutions to a sub-problem instance, starting with the highest. The satisfaction degree of the plans found increases along with the plan length, providing a mechanism for trading plan length versus the compromises made in the plan.

Solution extraction can be ameliorated using a similar basis to that for limiting graph expansion. Once a plan of satisfaction degree l_i has been found, all search paths which necessarily lead to a plan of lower or equal satisfaction can be pruned. Hence, the banded extraction of solutions need only extend to l_j such that $l_j > l_i$, guaranteed by the use of the *min* operator.

5.1 Memoisation

A fundamental aspect of Graphplan's search is the use of memoisation to store sets of subgoals determined to be unsupportable at a particular graph layer. This technique is also adopted by FGP. Unsupportable subgoal sets (*memosets*, composed of CSP variables) are identified when a problem instance is over-constrained via value-based learning [8], which enables the removal of irrelevant variables from a candidate memoset at minimal cost. Memosets are stored so that no further attempt is made to solve problems involving these subgoals (or a superset) at this layer. Memoset information is also propagated up the layer hierarchy such that memosets for each problem at layer$_{i-1}$ resulting from a solution to a problem instance at layer$_i$ are combined to produce a memoset for the problem instance at layer$_i$ (figure 5).

6 Experimental Results

FGP is first evaluated against leading boolean solvers to establish the efficacy of the novel DFCSP approach used. Table 1 shows the results obtained when running FGP, Graphplan, STAN v4 [14], IPP v4 [12] and BLACKBOX 3.6b (SATZ-RAND) [10] on benchmark problems from the logistics domain. FGP is implemented in Java and the other solvers in C. FGP performs strongly across this range of benchmarks, especially on more difficult problems. STAN and IPP contain enhancements to the Graphplan framework not included in FGP. It can be assumed that adding such enhancements FGP would provide similar performance improvements. The results from BLACKBOX suggest the exploration of the compilation of the flexible planning graph to a propositional satisfiability problem in future work. The flexible machinery does not appear to cause performance degradation on boolean problems.

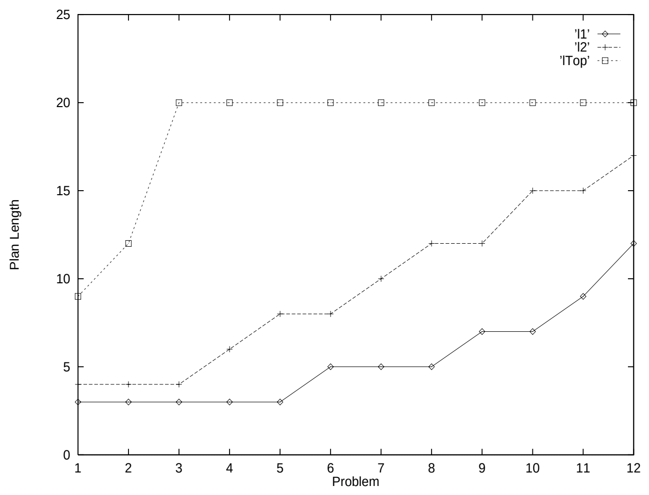
Problem	Length	FGP	GP	STAN	IPP	BBOX
Rocketa	7	14	75	33	47	5
Rocketb	7	21	154	2	76	8
Log-a	11	8	1955	1	1513	7
Log-b	13	99	862	2	633	24
Log-c	13	123	-	704	-	46
Log-d	14	171	-	7096	-	108

Table 1. Boolean problem comparison (in seconds) between FGP and leading solvers. A dash indicates no solution found in 24 hours. Hardware used: Sun Ultra 5.

In the second series of experiments, the overhead of looking for a range of solutions is evaluated by comparing the search process on a flexible planning problem against a boolean version of the same problem constructed using the endpoints in the scales K and L. K and L are defined as follows: $K = \{k_\perp, k_1, k_2, k_\top\}$, $L = \{l_\perp, l_1, l_2, l_\top\}$. The domain used was a variant of the flexible logistics domain introduced in section 4.2. The test suite contained 12 problems, with plan lengths at the satisfaction bands as shown in figure 6, which also shows the resultant run-times. As expected, it always takes longer to find a compromise-free plan when also searching for shorter 'compromise' plans than it does to simply solve the boolean problem. However, the time taken to produce shorter compromise plans is significantly lower than to solve the boolean problem, providing an attractive 'anytime' behaviour. This advantage is only reduced when the lengths of compromise plans and the plan with satisfaction degree $l_\top$ are similar, reducing the effects of limited graph expansion. However, the user still has the luxury of choosing from three alternative plans given slightly more run-time.

7 Conclusion

This paper introduced the flexible planning problem, a representation designed to capture the inherent 'softness' found in many real problems. This is achieved by assigning subjective truth degrees to propositions which are exploited by flexible operators that express one of a range of satisfaction degrees depending on how well their preconditions are satisfied. This general representation is independent of the technique used to synthesise flexible plans. A solution strategy which extends the Graphplan framework was introduced and a method was described for limiting graph expansion to those operator clauses which can lead to a better plan than the best one currently known. Solution extraction consists of solving a linked sequence of dynamic flexible constraint satisfaction problems (DFCSP). The DFCSP algorithm Flexible Local Changes (FLC) was employed to solve this sequence. Experimental results demonstrate that FGP ef-



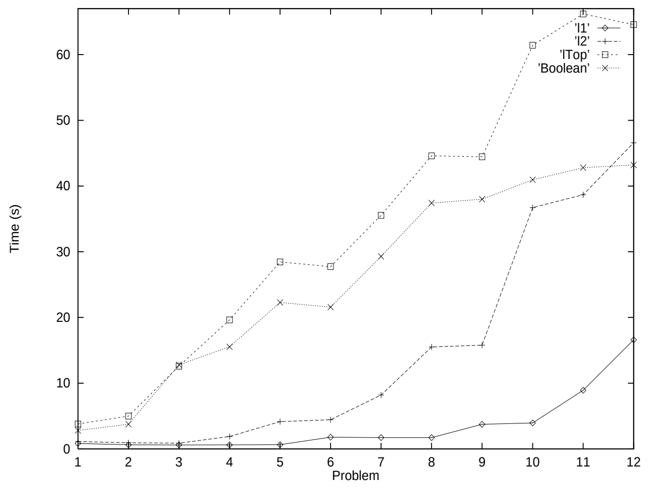


Figure 6. Flexible Problems: Composition and Run-times.

ficiently produces a range of solutions to a given flexible planning problem.

The disadvantage of using an idempotent operator is the so-called *drowning effect* [23], where a low satisfaction degree resulting from one assignment 'drowns' several others whose satisfaction degrees, whether optimal or sub-optimal, are not reflected in the overall satisfaction degree. Leximin CSP [18] combines additive and possibilistic approaches to avoid this problem, although at a significantly higher computational cost. Making use of such an approach in the context of the flexible planning problem is an active research topic.

ACKNOWLEDGEMENTS

Ian Miguel is supported by UK-EPSRC grant number 97305803. Peter Jarvis is supported by DARPA contract F30602-97-C-0067 under the supervision of the AFRL-Rome.

REFERENCES

[1] J.A. Ambros-Ingerson and S. Steel, 'Integrating planning, execution and monitoring', *Proceedings of the 9th American National Conference on Artificial Intelligence*, 83–88, (1988).

[2] S. Andrews, B. Kettler, K. Erol, and J. Hendler, 'UM translog: A planning domain for the development and benchmarking of planning systems', Tech rpt., Univ. of Maryland, (1995).

[3] A. Blum and M. Furst, 'Fast planning through planning graph analysis', *Artificial Intelligence*, **90**(1-2), 281–300, (1997).

[4] A. Blum and J. Langford, 'Probabilistic planning in the graphplan framework', *Proceedings of the 5th European conference on Planning Systems, Durham, UK*, 320–332, (1999).

[5] D. Draper, S. Hanks, and D. Weld, 'Probabilistic planning with information gathering and contingent execution', *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems*, 31–36, (1994).

[6] D. Dubois, H. Fargier, and H. Prade, 'Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty', *Applied Intelligence*, **6**, 287–309, (1996).

[7] G. Ferguson and J. Allen, 'TRIPS: An intelligent integrated problem-solving assistant', *Proceedings of the 15th American National Conference on Artificial Intelligence*, 567–573, (1998).

[8] D. Frost and R. Dechter, 'Dead-end driven learning', *Proceedings of the 12th American National Conference on Artificial Intelligence*, 294–300, (1994).

[9] S. Kambhampati, 'Planning graph as (dynamic) CSP: Exploiting EBL, DDB and other CSP techniques in graphplan', *Journal of AI Research*, **12**, 1–34, (2000).

[10] H. Kautz and B. Selman, 'Unifying SAT-based and graph-based planning', *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 318–325, (1999).

[11] H. Kautz and J. Walser, 'State-space planning by integer optimization', *Proceedings of the 16th American National Conference on Artificial Intelligence*, 526–533, (1999).

[12] J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos, 'Extending planning graphs to an adl subset', *ECP-97*, 273–285, (1997).

[13] N. Kushmerick, S. Hanks, and D. Weld, 'An algorithm for probabilistic planning', *Artificial Intelligence*, **76**(1-2), 239–286, (1995).

[14] D. Long and M. Fox, 'Efficient implementation of the plan graph in STAN', *Journal of AI Research*, **10**, 87–115, (1998).

[15] I. Miguel and Q. Shen, 'Hard, flexible and dynamic constraint satisfaction', *Knowledge Engineering Review*, **14**(3), 199–220, (1999).

[16] I. Miguel and Q. Shen, 'Dynamic flexible constraint satisfacton', *Applied Intelligence* (in press), (2000).

[17] S. Mittal and B. Falkenhainer, 'Dynamic constraint satisfaction problems', *Proceedings of the 8th American National Conference on Artificial Intelligence*, 25–32, (1990).

[18] J. Moura Pires, F. Moura Pires, and R. Almeida Ribeiro, 'Structure and properties of leximin FCSP and its influence on optimisation problems', *Proceedings of the 7th Conference on Information Processing and Management of Uncertainty*, 188–194, (1998).

[19] K. Myers and T. Lee, 'Generating qualitatively different plans through metatheoretic biases', *Proceedings of the 16th American National Conference on Artificial Intelligence*, 570–576, (1999).

[20] E. Pednault, 'ADL: Exploring the middle ground between strips and the situation calculus', *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, 324–332, (1989).

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

[22] M. Peot and D. Smith, 'Conditional non-linear planning', *Proceedings of the 1st International Conference on Artificial Intelligence Planning Systems*, 189–197, (1992).

[23] T. Schiex, H. Fargier, and G. Verfaillie, 'Valued constraint satisfaction problems: Hard and easy problems', *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 631–637, (1995).

[24] A. Tate, J. Dalton, and J. Levine, 'Generation of multiple qualitatively different plans', *Proceedings of the 4th International Conference on Artificial Intelligence Planning Systems*, 27–34, (1998).

[25] T. Vossen, M. Ball, A. Lotem, and D. Nau, 'Applying integer programming to ai planning', *Knowledge Engineering Review* (to appear), (2000).

[26] D. Wilkins, *Practical Planning*, Morgan Kaufmann, 1988.