UPGRADE
The European Online Magazine for the IT Professional
http://www.upgrade-cepis.org
Vol. III, No. 5, October 2002

UPGRADE is the European Online Magazine for the Information Technology Professionals, published bimonthly at <http://www.upgrade-cepis.org/>.

Publisher
UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by NOVÁTICA <http://www.ati.es/novatica/>, journal of the Spanish CEPIS society ATI (Asociación de Técnicos de Informática <http://www.ati.es/>).
UPGRADE is also published in Spanish (full issue printed, some articles online) by NOVÁTICA, and in Italian (abstracts and some articles online) by the Italian CEPIS society ALSI <http://www.alsi.it> and the Italian IT portal Tecnoteca <http://www.tecnoteca.it/>.
UPGRADE was created in October 2000 by CEPIS and was first published by NOVÁTICA and INFORMATIK/INFORMATIQUE, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>).

Chief Editors
François Louis Nicolet, Zürich <nicolet@acm.org>
Rafael Fernández Calvo, Madrid <rfcalvo@ati.es>

Editorial Board
Prof. Wolffried Stucky, CEPIS President
Fernando Piera Gómez and
Rafael Fernández Calvo, ATI (Spain)
François Louis Nicolet, SI (Switzerland)
Roberto Carniel, ALSI – Tecnoteca (Italy)

English Editors: Mike Andersson, Richard Butchart, David Cash, Arthur Cook, Tracey Darch, Laura Davies, Nick Dunn, Rodney Fennemore, Hilary Green, Roger Harris, Michael Hird, Jim Holder, Alasdair MacLeod, Pat Moody, Adam David Moss, Phil Parkin, Brian Robson.

Cover page designed by Antonio Crespo Foix, © ATI 2002

Layout: Pascale Schürmann

E-mail addresses for editorial correspondence: <nicolet@acm.org> and <rfcalvo@ati.es>

E-mail address for advertising correspondence: <novatica@ati.es>

Copyright
© Novática. All rights reserved. Abstracting is permitted with credit to the source. For copying, reprint, or republication permission, write to the editors.

The opinions expressed by the authors are their exclusive responsibility.

ISSN 1684-5285

Coming issue:
"Security in E-Commerce/Business"

Artificial Intelligence: Technology with a Future

Guest Editors: Federico Barber, Vicente J. Botti, and Jana Koehler

Joint issue with NOVÁTICA

Progress in AI Planning Research and Applications

Derek Long and Maria Fox

Planning has made significant progress since its inception in the 1970s, in terms both of the efficiency and sophistication of its algorithms and representations and its potential for application to real problems. In this paper we sketch the foundations of planning as a sub-field of Artificial Intelligence and the history of its development over the past three decades. We will then discuss some of the recent achievements within the field and provide some experimental data demonstrating the progress that has been made in the application of general planners to realistic and complex problems. We conclude by identifying some of the open issues that remain as important challenges for future research in planning.

Keywords: AI, Planning of Actions, Plans, Scheduling and Planning

1 Introduction

Planning is a sub-field of Artificial Intelligence (AI), explored by researchers in the AI community for more than three decades. Newell and Simon's work on GPS [Newell/Simon 1963], Green's QA3 [Green 1969] and McCarthy's situation calculus [McCarthy/Hayes 1969] helped to define the *classical planning problem* and many of the basic assumptions made then still influence planning research today. This paper surveys the objectives of the research field, the progress of researchers towards meeting them and some of the current activities and themes in the area. It also considers the extent to which modern planning techniques are ready for wider exploitation and what still remains to be achieved.

In the early days of AI research scientists pursued a broad and ambitious, if somewhat ill-defined, objective of creating an intelligent machine. Reasoning capabilities were seen as central to this objective, but were expected to be based on an interlocking collection of generic mechanisms. This can be seen in the application of theorem proving as a general technology to all kinds of reasoning problems, including planning. As the subject has developed it has become clear that generic reasoning, if it can ever be achieved, can only be built on a thorough understanding of more specific examples of human problem-solving enterprises. Consequently, researchers have explored different areas of problem-solving reasoning and AI has splintered into a collection of different sub-fields. Planning emerged as a specific sub-field with the seminal work of Fikes and Nilsson [Fikes/Nilsson 1971] on the STanford Research Institute Problem Solver (STRIPS).

This paper begins, in section 2, with a description of the planning problem itself and the constraints that have been imposed to make tractable versions of it. In order to understand the foundations for the most recent developments in planning, it is helpful to review the history of research in planning – this is covered in section 4. In sections 5 and 6 the more recent developments in the field are described. Finally, in section 7, some of the many problems that remain to be solved, or solved more successfully, are considered.

2 The Planning Problem

To make the planning problem accessible, it is necessary to have a precise definition of what the problem is and what constitutes a solution to an instance of the problem. In defining the problem, several simplifications have been made that do not always characterise planning problems in general application, but make the core of what remains a more manageable starting point for research. Different researchers have adopted slightly

Derek Long is a Lecturer in Computer Science at the University of Durham (United Kingdom). His research interests lie in planning, and applications of planning systems. Dr Long co-chaired the 3rd International Planning Competition, held at AIPS 2002. He also co-developed the STAN planning system and the TIM planning domain analysis system, establishing the concept of generic types and their role in planning domain construction and decomposition. Within the Hybrid STAN extension of the original STAN system, he and his colleague Maria Fox have demonstrated the potential to automatically configure problem-solving technology to support a planning system by attacking sub-problems using specialised solvers. He is chairman of the UK Planning and Scheduling Special Interest Group, a group that holds an annual meeting attracting international participation.
<d.p.long@durham.ac.uk>

Maria Fox is Reader in Computer Science at the University of Durham (United Kingdom). Prior to joining Durham University Dr. Fox was a lecturer at University College London. Her early work in AI Planning considered the problems of generating and refining abstract plans through soundness-preserving transformations. More recently she has focused on representation and reasoning issues in temporal and metric planning. With Derek Long she co-chaired the third international planning competition which stressed planning in temporal domains. She has developed planning algorithms and domain analysis techniques for a range of planning domain description languages and is currently working in the area of autonomous planning and execution.
<maria.fox@durham.ac.uk>

© Novática

different formulations of the problem, but the following is the most widely adopted starting point:

A planning problem is described by a collection of actions, each characterised by their preconditions (what must be true in order for the action to be executed) and their postconditions (which describe the effect of execution of the action), an initial state of the world and a description of the goals to be achieved. The problem is solved by finding actions that will transform the given initial state into a state satisfying the given goals.

It can be observed that this *action-centric* view of the planning problem is influenced by the notion of *state*, or *situation*, and of *transition* between states. This view has a very strong heritage based on McCarthy's development of the situation calculus. This calculus describes how situations, described in a first order language, are affected by actions performed on them by an executive agent. *Effect axioms* describe how actions change the situations in which they are applied into new situations while *frame axioms* describe what aspects of a situation remain *un*affected as actions are applied. These axioms are specified in terms of the relations and predicates that describe configurations of objects in the world, together with situation variables that enable the facts associated with one situation to be distinguished from those associated with a successor situation. Axioms are universally quantified over situations. Given a complete set of such axioms it is possible to deduce the situation that results from the application of a chain of actions and to determine which actions to apply to obtain a desired state change.

An important shortcoming of the situation calculus is the difficulty of defining a complete set of effect and frame axioms for a non-trivial world. Effect axioms are easier to define because the number of actions that need to be described is contained and they are usually identified with only a small number of positive state changing effects. Frame axioms, on the other hand, are extremely difficult to define exhaustively because the number of properties of a situation that *do not* change when an action is applied is far greater than the number that do. Furthermore, it is not natural to think about the world in negative terms so there is a great danger of providing only a partial collection of frame axioms which would lead to unsound reasoning within the calculus. This problem is referred to as the *frame problem*.

The STRIPS system made a very important contribution to Planning research by introducing the *Strips Assumption* as a way to avoid the complexity of the frame problem for the purposes of planning within the situation calculus. The assumption is that the only changes that arise on application of an action to a situation are those that are explicitly mentioned as positive effects of the action. All other relations and predicates, associated with the situation in which the action is applied, are automatically deduced to hold in the successor situation. The STRIPS project introduced a simple syntax for defining action schemas, in terms of the *pre*conditions, *add* effects and *delete* effects of the action. An example can be seen in Figure 1. Despite the many advances that have been made in planning research over the years the STRIPS assumption continues to be a fundamental principle in the modelling of planning problems and an important influence on the way planning algorithms are designed.

```
Action LOAD ?object ?container
       ?location
    Precondition:
       at(?object,?location)
       at(?container,?location)
       empty(?container)
    Add:
       inside(?object,?location)
    Delete:
       at(?object,?location)
       empty(?container)

Action UNLOAD ?object ?container
       ?location
    Precondition:
       at(?container,?location)
       inside(?object,?location)
    Add:
       at(?object,?location)
    Delete:
       inside(?object,?container)
       empty(?container)

Action MOVE ?container ?start
       ?destination
    Precondition:
       at(?container,?start)
       link(?start,?destination)
    Add:
       at(?container,?destination)
    Delete:
       at(?container,?start)

    Initially:
       at(PickUp,Home)
       at(Box,Office)
       link(Home,Town)
       link(Town,Home)
       link(Town,Office)
       link(Office,Town)
    Goal:  at(Box,Home)
```

Figure 1: Simple domain description and problem instance. Terms marked with "?" are variables.

Planning problems are fundamentally dynamic in structure – an initial situation is presented, a goal is defined and a plan is seen as a sequence of (sets of) state changes applied over time. It is therefore natural to interpret collections of action schemas as defining the transitions in a parameterised automaton (see Figure 2) and a plan as the transitions traversed by an accepting trace through the instantiated automaton. This view is in contrast with the static view imposed by the situation calculus,

 © Novática

in which theorem-proving determines whether a particular collection of constraints is consistent and the passage of time is modelled through the use of state variables at the object level. A goal state is achievable if the collection of effect and frame axioms are satisfiable for some value of the state variable associated with the goal. Both the dynamic and static views have influenced the design of algorithms for planning, although the dynamic view has dominated approaches taken to representation of planning problems.

This dominance is apparent in the family of Planning Domain Description Languages (PDDL1.2 [McDermott 1998] and PDDL2.1 [Fox/Long 2002]) that have been proposed as standards for modelling planning problems. These languages are based on the STRIPS assumption and support the modelling of a planning problem in terms of a compact representation of the finite state automaton that describes its behaviour. As is discussed in the following sections, this style of modelling can be extended to support reasoning about continuous as well as logical change, and can provide sufficient expressive power for the modelling of very complex realistic planning problems. The question of how algorithms might be developed to enable plans to be found, given such models, is a separate one and is discussed below.

3 Foundations of Planning

The action-centred view of problem representation makes a number of simplifying assumptions which define classical planning. First, it is assumed that the evolution of action sequences, applied to a completely known initial situation, can be completely and correctly predicted as though no external influences were operating in the world. Second, planning is the task of constructing a single completed plan that achieves the goal, *prior* to the execution of any part of it. A third, related point, is that the classical planning formulation assumes that the goals are known before planning starts – planners do not set their own goals and goals do not change as execution progresses. This makes *classical* planning a poor technology for realistic problems in which goals arise continually and important things can happen outside the control of the planner. Finally, in classical planning, in which numeric reasoning is excluded, plan quality is determined solely by the number of actions in the plan. This is, of course, a very simplistic measurement of plan quality and one that is further discussed in section 6, where recent work going beyond this assumption is examined.

Even under these simplifying assumptions plan generation is computationally a very hard problem. Under the assumption that the space of reachable states is finite planning belongs to the PSPACE-hard class of problems meaning that the number of states that must be considered in attempting to find a path from the initial state to the goal is likely to be exponential in the size of the description of the planning problem. The space of reachable states is certainly exponential in the size of the problem description (the description is schematic whilst the state space is fully instantiated) so the task of a planning algorithm is to find a path between the initial situation and one satisfying the goal, whilst exploring as little as possible of the state space in the process. This is what makes plan generation different from the problem of finding a shortest path in a graph – the graph is too big to be built explicitly (see Figure 3) so a plan generation algorithm must intelligently build only that part of it in which the solution lies.

The intelligent exploration of the problem state space depends on the ability of the planning algorithm to exploit powerful heuristics or control knowledge to guide its search. Many recent strides have been made in planning because of the discovery of informative heuristics which can be very effective in directing search towards a solution.

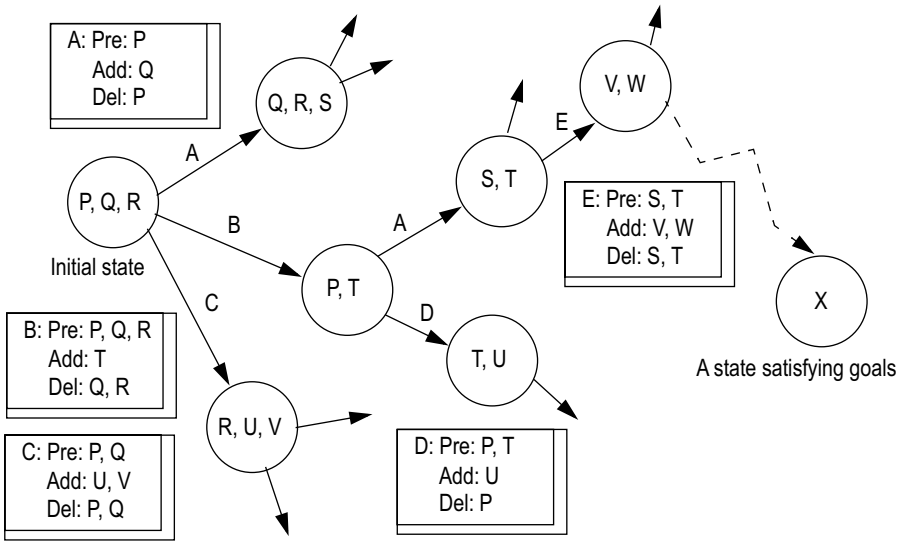

Figure 2: Illustration of a fragment of a state space for a planning problem. Note that the same action can cause transition between different pairs of states, with untouched parts of the start state preserved by the STRIPS assumption.

Planning research has traditionally relied on simple and relatively unstructured models of the problem and placed the research emphasis on the development of algorithms and powerful heuristic control methods. Although it has been recognized [Fox/Long 1998] [Gerevini/Schubert 1998] that a model typically contains hidden structure that can be exploited by a planner the tendency has been to persist with the construction and communication of traditional models and, either to supplement the traditional model with problem-specific control rules [Bacchus/Kabanza 2000] [Kvarnstrom/Doherty 2000] [Nau et al. 1999], or to use automated analysis to extract the hidden structure and make it accessible to the reasoning mechanisms of the planner [Fox/Long 2000] [Porteous 2001] [Gerevini/Schubert 1998]. The recent successes of planners using problem-specific control rules (see section

© Novática

FreeCell	4 cards per suit 13 cards per suit	1.1×10^{11} initial states 1.75×10^{64} initial states
Logistics	Largest problem in 1st IPC Largest problem in 2nd IPC (fully automated)	3×10^{23} states (solved in more than 13 minutes) 2×10^{87} states (solved in 80 seconds)

Figure 3: The table gives a very brief indication of the sizes of planning problem state-spaces. FreeCell is a problem based on the well-known solitaire card game, introduced as a planning benchtest in the 2nd International Planning Competition (IPC) [Bacchus 2000]. The Logistics problem is a commonly used benchtest for planning systems. It is not easy to estimate the size of the state space in all problems. For FreeCell the number of essentially distinct initial states gives an impression of the size of the state space. In the Logistics domain it is easier to compute the number of different reachable states. It is interesting to see the contrast in the performances between the 1st IPC (1998) and 2nd IPC (2000).

4.5) raises the question of how much modelling can impact on search efficiency. Other AI research communities have focused on exploring the extent to which modelling choices can expedite the solution of a problem [Borrett/Tsang 2001] [Westfold/Smith 2001]. Certainly, the more human expertise is embedded in the model the less discovery has to be made by the solver (whether planner, constraints problem solver, scheduling algorithm or other), but the burden on the human expert can be prohibitive. The correctness of the reasoning system depends fundamentally on the correctness of the model, so errors in the modelling can be catastrophic. The traditional approach taken in planning has been to limit this burden as far as possible, providing a standard means for modelling of action-centred behaviour and placing the problem-solving emphasis on automated techniques.

The complexity of the search problem, and the difficulty of identifying powerful and general search control methods, has tended to limit the scope for fielded applications of planning technology. A central problem is that all decision making issues are thrown into the same brute-force, often unexpected, search strategy and a planner quickly becomes unable to cope with the branching factor of the problem. This approach, has been questioned by researchers addressing real, and often highly structured, problems. Indeed, some researchers [Ingham et al. 2001] have argued that, whilst a planner must be left free to discover novel solutions to some parts of a problem there will always be other parts the solutions of which can be prescribed quite closely. Furthermore, a problem expert might be able to provide an overall flexible plan structure which would guide and constrain the search behaviour of a plan generation system whilst leaving many choices to be resolved in closer contact with an executive. The standard languages for representing planning problems do not provide this facility, but researchers in the related field of model-based reasoning have designed languages, such as RMPL [Ingham et al. 2001], which support this purpose and might provide a basis for combining the power of plan generation with the ability to exploit the knowledge of a problem expert.

4 The Development of Planning Research

A wide variety of planning algorithms has been developed since STRIPS, using different search strategies and exploring different search spaces. In the 1970s and 1980s effort was focused on puzzle-like problems, such as the blocks world, tile puzzles and the towers of Hanoi, all of which involve highly inter-dependent actions and constraints. The objective was to find optimal plans (plans that would achieve the goal state using as few action instances as possible) and planning approaches were considered flawed if they were fundamentally unable to achieve optimality in particular puzzle domains. For example, the well-known Sussman anomaly led to the development of least-commitment planning as an alternative to the sequential planning style of STRIPS. The *Sussman anomaly* (depicted in Figure 4) is a feature of the blocks world demonstrating that goals cannot always be decomposed. Similarly, the towers of Hanoi problem shows that progress cannot always be made monotonically.

Although considerable progress was made in the development of planning algorithms, and in the understanding of their formal properties, by means of consideration of these puzzles, planning did not make much progress towards practical application during these years. Indeed, the wider AI community raised questions about the utility of classical planning given its apparent inability to dispose even of puzzle problems of little or no practical interest. During this period work began to emerge on the impressive results that could be obtained in the solution of practical problems by planners equipped with problem-specific knowledge [Currie/Tate 1991] [Wilkins/desJardins 2000] [Stefik 1981]. This gave rise to a tension in the planning research community between classical, *domain independent*, and application-oriented planning. The advantages of taking a classical approach is that the techniques developed are completely general and reusable, whilst the advantages of the application-oriented approach is that excellent (though not reusable) results can be obtained in specific problem domains.

This tension has always been constructive because it has helped to drive classical planning research towards the solution of ever more realistic problems. In the last ten years incredible strides forward have been made in terms of the complexity of problems that can be addressed, the efficiency with which solutions can be generated and the quality of those solutions. In the early 1990s no classical planner could produce plans of more than, say, thirty steps. Ten years later such planners can produce plans consisting of hundreds of steps in a fraction of the time it used to take to produce plans an order of magnitude shorter. Furthermore, modern planners are capable of handling problems that are much closer to practical utility than the puzzle problems of old.

 © Novática

One of the most important forces driving these developments has been the biennial planning competition which started in 1998. The first competition was organized and run by Drew McDermott in consultation with an international committee of experts whose task it was to standardize the representation language to be used by planners and to define a range of bench mark problems. This competition led to the emergence of the PDDL family of domain description languages. The bench mark problems used in 1998 still owed much to the puzzle problem heritage, although the Logistics domain, popularized by Kautz and Selman [Kautz/Selman 1998], was used to begin to push planners towards the solution of problems with a more practical emphasis. Only five competitors participated in this competition, but it led to an enormous resurgence of interest in classical planning and its potential for application. One of the most interesting consequences of the 1998 competition was that emphasis on plan optimality began to give way to a willingness to trade off optimality for speed of plan generation. Optimality is a property of plans that is too hard to check, and because of the difficulty of producing guaranteed optimal plans, plans produced by systems that do not guarantee optimality cannot be evaluated in terms of their distance from optimal. However, the heuristic forward search planner HSP [Bonet et al. 1997] emerged as a very exciting future planning technology because of its ability to find "acceptable" plans very quickly (where a plan might be considered acceptable if it is no more than, say, ten per cent longer than competitive plans). This led to an intense period of research activity into heuristic forward search.

The two following competitions, in 2000 and 2002, introduced a range of new bench marks emphasising practical problem features. In particular, in 2002 problems involving numeric and temporal reasoning were introduced and planners were required to reason with intervals of time and the consumption and production of numeric resources. Some of the bench marks introduced in 2002 began to closely approximate problems of real practical interest. For example the *Timed Rover* domain, shown in Figure 4, closely models the planetary rover exploratory problems being researched into by NASA and other space agencies (see also section 6). Planning has moved from being a puzzle-solving technology to being the foundation of autonomous behaviour.

In the following sections we review some of the advances in planning algorithms that have supported the developments outlined above.

4.1. Graphplan

Graphplan [Blum/Furs 1995] excited a great deal of interest when it was introduced because it constitutes an approach to planning that was radically new at the time (in the early 1990s). Graphplan constructs and then searches a compact reachability analysis of the problem state space. The compactness of the representation, together with the informative nature of the data that can be accumulated during both construction and search,

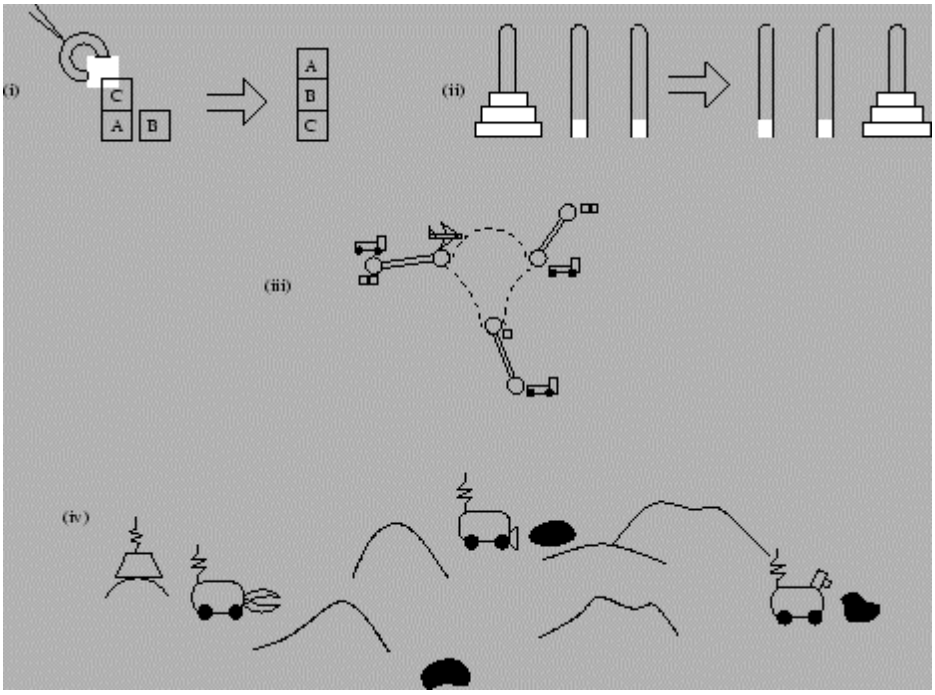

Figure 4: Examples of planning domains. (i) Simple blocks world (the problem shown is Sussman's anomaly). (ii) The towers of Hanoi problem. Constraints require that the discs move between pegs without ever placing a larger disc on a smaller one. (iii) Logistics domain: packages must be transported between locations. The airplanes are constrained to move between airports, while the trucks are restricted to local regions. (iv) Depiction of rovers domain: differently equipped rovers rove around planet surface, examining sites of interest and recording data from experiments. Data is communicated back to a lander and from there to earth. Constraints limit data storage, fuel use and recharging, rover capabilities and so on.

© Novática

leads to performance that far outstripped that of contemporary planning search strategies. The success of Graphplan led to the development of a number of Graphplan re-implementations and extensions [Koehler et al. 1997] [Long/Fox 1999] and some of its contributions remain very important today (although its planning performance has been surpassed).

Graphplan searches for a plan in two stages. The first stage is the construction of a data structure, the *plan graph*, that efficiently represents information about what the executive could possibly achieve by executing actions from the initial state. The second stage searches, backwards from the goals, for a substructure within the plan graph that represents a subset of actions that will actually achieve the goals. An important preparatory step used by Graphplan, that has become a common first step for many planning systems, is the *grounding* of all the actions. This is the process in which all the descriptions of parameterised actions are instantiated with all possible values for the parameters selected from the objects in the problem instance. Grounding for large problem instances can be a memory intensive operation, but clever compact encodings can allow tens of thousands and even hundreds of thousands of action instances to be constructed and stored efficiently.

The important information that is captured within the plan graph is the collection of propositions that *could*, individually, be made true after application of increasing numbers of actions. In addition, the graph shows which pairs of propositions are mutually incompatible. That is, pairs of facts that cannot both be made true in a reachable state of the world. This might be because the facts are simply inconsistent (such as that a door is both open and closed) or it might be because achieving both of them requires execution of more actions than the graph currently allows. As the graph is extended by adding the effects of execution of more actions, these pairs of facts will become compatible. Once all the facts that form the goal set for the planning problem appear in the graph and are pairwise compatible, the search phase commences.

Graphplan search uses several additional techniques to improve efficiency, but is essentially an exhaustive backward search from the goals, looking for achieving actions that have compatible preconditions. The search is carried out depth-first, but is limited by the number of actions that the graph data structure contains. If no plan is found, the graph is extended and a new search commences. This process makes Graphplan perform an *iterated depth-first search* for a plan. This search strategy combines the usual benefits of depth-first search (low overhead in memory consumption) with the important benefit of a breadth-first search, which is that it will find the shortest solution. A failing of this search strategy is that it repeatedly explores the same set of choices as the depth bound increases. This failing impacts on Graphplan performance, so that if a plan cannot be found in the first few iterations of the search then it is often the case that Graphplan will exceed its resource bounds before finding a plan.

4.2. Heuristics Search Planning

Following the significant success of Graphplan, interest in the planning problem was revitalised and other new ideas were explored. A very influential direction was initiated by work of McDermott in the planning system UNPOP [McDermott 1996] and Geffner and Bonet in HSP [Bonet et al. 1997]. The idea behind this work is to use a classic heuristic guided search. This is the search strategy in which the choice between alternatives is made by evaluating each alternative using a heuristic evaluation function and then selecting the best valued option. There are several ways in which the heuristic guidance can be applied, such as the well-known *A** search, hill-climbing searches, best-first searches and so on [Russell/Norvig 1995]. Using heuristic search was not new in planning – many planners, such as UCPOP [Penberthy/Weld 1992], allow a heuristic function to be used to guide the search. However, heuristic search had not previously seemed very promising. The difficulties that had been encountered by earlier planners attempting to exploit heuristic choice arose from the fact that the heuristic function was usually encoded by hand and it was often difficult to construct a function that could reliably guide the planner to make all the right choices.

The novel contribution made by McDermott and by Geffner and Bonet was to demonstrate a method by which a surprisingly informative heuristic function could be constructed *automatically*, simply by analysing the domain. Although the details of the techniques differ, the underlying approach is the same: the heuristic value of a particular choice of action is based on an estimate of how much work remains to be accomplished following the addition of that action to the plan. To estimate the outstanding work a very simple, yet very effective, measurement is made: the number of actions required to achieve all the outstanding goals *if the destructive effects of those actions are ignored*. Achieving goals using actions whose destructive effects are ignored is called *relaxed planning*. The measure of outstanding work is simply the size of a relaxed plan to achieve the goals. Unfortunately, finding an optimal relaxed plan is, perhaps surprisingly, technically as hard as finding a real plan [Erol et al. 1995]. Fortunately, however, it is relatively easy to find arbitrary relaxed plans, and even to find "good" relaxed plans. The work inspired by McDermott and Geffner and Bonet, then, uses efficient techniques to construct good relaxed plans which are treated as reasonably accurate measures of the work required to complete a plan if a given choice of action is pursued.

One of the most efficient planners based on this approach is the FF system developed by Hoffmann [Hoffmann/Nebel 2000]. This planner uses a *relaxed* plan graph (built using actions with their destructive effects ignored) and an efficient plan graph search to find good relaxed plans. Using this approach yields a slightly different heuristic estimate from that exploited in HSP, and the results seem slightly better. The heuristic exploited by HSP is *admissible* (it never over-estimates the distance to the goal) whilst the plan graph based heuristic of FF is *in*admissible because it relies on extraction of a relaxed plan from the plan graph, and there is no guarantee that the plan extracted will be the shortest one available. Geffner and Bonet have explored a wide variety of variants on the relaxed plan heuristic [Bonet/Geffner 1997]. Hoffmann has carried out a thorough exploration of the way in which the optimal relaxed

 © Novática

plan relates to the optimal real plan in a variety of bench mark domains, allowing the identification of several topological features that govern the success or failure of the relaxed planning heuristic as a guide towards real plans.

4.3. Transformation of Planning Problems

A different direction has been explored by several researchers: the reformulation of a planning problem into a form that is amenable to solution using a technology developed in a different research field. There are at least three examples of this approach: the SAT-planners, in which a planning problem is converted into an instance of a SATISFIABILITY problem and solved using an efficient SAT-solver [Kautz/Selman 1995]; the CSP approach, in which a planning problem is reformulated as a constraint satisfaction problem and solved using a CSP-solver [vanBeek/Chen 1999] [Binh/Kambhampati 2000] and the model-checking approach in which a planning problem is converted into a large logical formula that can be model-checked [Edelkamp/Helmert 2000] [Cimatti/Roveri 1999] (this is actually a variation on the SAT-planning approach, but using sufficiently distinct technology for checking the formula that it is generally seen as a direction in its own right).

The SAT-solving approach has been explored most extensively in the Blackbox system developed by Kautz and Selman [Kautz/Selman 1995]. The approach exploits the expressive power of the SATISFIABILITY problem. Briefly, this problem can be summarised as that of attempting to determine whether there is an assignment of truth values to propositional variables appearing in a given propositional formula that will make the formula true. The expressive power of propositional formulae is such that it is possible to express the structure and constraints of a planning problem. There is an interesting complication, which is that a SAT-instance can only express the possibility of their being a plan of a maximum given size. If the formula cannot be satisfied then a new formula must be constructed (usually by a straightforward extension of the existing formula) to express the possibility of there being a longer plan. Thus, the problem of finding a plan becomes not one instance of SATISFIABILITY, but a sequence of instances as the search attempts to find a plan with iteratively increasing numbers of actions. The Blackbox system encompasses several different SAT-solving techniques, each configurable with different parameters controlling their behaviour, but the underlying translation of a planning problem into a SATISFIABILITY instance is via a plangraph. Two extensions of SAT-planning have explored extended expressive power that encompasses numeric valued expressions: LPSAT combines a SAT-solver with a linear-constraint solver to handle a restricted (but very powerful) collection of numeric-valued expressions [Wolfman/Weld 1999] and constraints and a version in which an integer programming solver is combined with a SAT-solver has also been tried [Vossen et al. 1999]. Integer programming is a similar optimization of numeric valued expressions, subject to constraints, to that tackled in linear programming (linear constraint satisfaction). The difference lies in the added constraint that variables take integer values instead of real values – an apparently minor addition, but one that has surprisingly far-reaching consequences for the difficulty of the search for a solution.

The CSP approach has been explored by van Beek [vanBeek/Chen 1999], using hand-coded translations of planning problems for his CSP solver. More recently, Binh Do and Kambhampati have shown [Binh/Kambhampati 2000] that automatic translation to CSP is possible, using a Graphplan-style plan graph as an intermediate structure. CSPs consist of collections of variables, each with an associated domain of possible values, and a collection of constraints that specify how values of variables must interact. A solver searches for assignments of values to the variables that are consistent with the constraints. Described at this abstract level, it is not hard to see how CSP and SATISFIABILITY can be closely related to one another – the assignment of truth values to propositional variables is a restricted form of the assignment of domain values to variables in a CSP. The CSP community has explored a variety of techniques for efficient CSP solving [Tsang 1993] and it might be hoped that these techniques have something to offer to planning. There probably are many lessons that can be learned within the planning community from the CSP community, but it is now apparent that, as with other reformulation techniques, the sacrifice of structure from the planning problem is very hard to repay with performance benefits in the technology that is made accessible for solving the newly expressed problems.

The third approach, model-checking, was explored by several of the competitors in the competition in 2000 [Edelkamp/Helmert 2000] [Hölldobler/Störr 2000]. The idea in this approach is to reformulate the planning problem as a logical formula that can be expressed as a binary decision diagram – a particularly compact representation for manipulating such formulae. Once again, the reformulation gives access to generic technology for solving problems in the form of the reformulated problem. In this case, an efficient model-checking approach can be used to determine the status of the formula. There remains a very active application of the approach in the solution of planning problems that involve actions with uncertain outcomes (see, for example, [Cimatti/Roveri 1999]).

4.4. Hybrid Planning

Experimental results demonstrate that there is no single planning strategy suited to all planning problems. All of the approaches discussed above display a wide range of performance over different problems and the extent to which a particular strategy is suited to a particular problem is determined by the structure and organization of the problem. For this reason *hybrid* planning approaches can be very successful. However, determining automatically which of the available strategies in a hybrid to invoke remains a difficult research problem.

Edelkamp and Helmert's MIPS system [Edelkamp/Helmert 2000] combined a Graphplan strategy with a model-checking approach, using time bounds to determine which of the strategies to apply to a problem. The Graphplan strategy was tried first, and if no solution was found within the time bound the problem would be reformulated as a model-checking problem. Similarly, the BLACKBOX system [Kautz/Selman 1995] is

© Novática

able to invoke a range of different SAT-solving techniques and these are tried successively until a solution is found (or resource bounds are exceeded).

Sometimes no generic search strategy is suited to the solution of a problem and, instead, some specialised solver should be invoked. A direction that has been explored in some recent work in planning [Fox/Long 2001] [Long/Fox 2000], is the idea of harnessing specialised solvers capable of efficiently tackling particular combinatorial problems, such as the well-known Travelling Salesman Problem (TSP), to support a general planning engine. The motivation for this is that many hard specialised problems have themselves been the subject of research and good solutions exist for them, while a generic problem-solving technology is very unlikely to challenge these specialised solutions when applied to these problems. It is often the case that these specialised problems appear as sub-problems within a larger planning problem. For example, a problem that involves transporting components between locations, constructing complex artifacts with the components and then delivering them to customers can be seen to contain route-planning, resource allocation, job-shop scheduling and construction planning sub-problems. Each of these can, to some extent (although not entirely) be decoupled from the others and solved using specialised technology.

Several examples have now been demonstrated in which a successful automatic identification of sub-problems is combined with automatic configuration of specialised problem-solving technology and solution of planning problems using these specialised solvers [Long/Fox 2001] [Clark 2001] [Long et al. 2000] [Long/Fox 2002]. Another planner exploiting a similar architecture, although without automatic problem decomposition, is REALPLAN [Srivastava 2000], which factors out resource-scheduling sub-problems for specialised treatment. The decomposition approach is of interest because it offers a way in which advances in particular problem-solving technologies might be exploited without impacting on the task of encoding the planning problem – automatic identification of sub-problems relieves the domain-engineer of the requirement to understand the characteristics of the sub-problems and also of the obligation to understand the restrictions on the capabilities of the sub-solvers or of the ways in which collections of solvers might interact in deployment.

4.5. Planners Exploiting Hand-Coded Control Knowledge

Although the development of fully-automated planning systems, which do not rely on advice from human domain-experts to guide their search, has been the focus of most of the survey so far, progress in planners exploiting hand-coded control knowledge has been equally dramatic in the past few years. The third IPC highlighted the performance of three such systems: TALplanner [Kvarnstrom/Doherty 2000], TLplan [Bacchus/Kabanza 2000] and SHOP2 [Nau et al. 1999]. The first two of these share significant common ground in their planning strategy, which is essentially a forward search, exploring the states that can be reached by the execution of actions from each state as it is visited, and selecting the next state to visit using carefully crafted advice provided by a human domain-engineer. The third system represents a class of planners known as *hierarchical task network* (HTN) planners. In these systems the domain must be described in terms of hierarchically structured actions, from an abstract level down to a concrete level of detail. An abstract action connects to one or more methods, which are possible "recipes" for accomplishing the abstract task using less abstract tasks. The planner uses a process of refinement to construct a plan, replacing abstract actions that achieve its initial goals with their less abstract method expansions and then, iteratively, refining the components of these expansions. We will now consider each of these two families of planners in a little more detail.

4.6. Hand-Coded Control Rules

TLplan and TALplanner both rely on control rules which guide the choices made by the planner as it considers which action to select at each step in the plan construction. Both of these systems use a temporal logic to express the rules, which allows the control rules to explicitly refer to logical relationships between properties of the states before and after application of an action, to the goal state, the initial state and also to collections of states such as all future states or all past states. Control rules can express advice such as "do not load an object onto a transporter if it is already at the place it needs to reach", or "if an unloaded object and a transporter are at the same place and the object needs to go somewhere that the transporter is scheduled to visit, then in the next state the transporter must still be at the same location and the object must either be loaded into the transporter or still waiting to be loaded". It can be seen from these examples that control rules can become quite cumbersome to write, even if they express relatively intuitive advice. In particular, the second of these examples is only valid if there is a single suitable transporter. If there is more than one transporter that could be used to transport the object then the rule becomes even more cumbersome to express. Some research has been conducted on automatically constructing rules in certain cases [Kvarnstrom et al. 2000] [Muñoz-Avila 2002], but the encoding of a good set of advice for a domain remains a demanding task.

One of the most significant disadvantages of planners using hand-coded control knowledge is that the task of writing requires expertise in both the domain and knowledge of the way that the planner will respond to the advice. Constructing collections of advice that are both consistent and allow a planner to find all plans that are sought is a complex task, not unlike programming. The benefits are considerable, however: TLplan and TALplanner produced plans of a consistently high quality, very fast, for a huge range of problems presented to it in the third IPC. TLplan can plan with durative actions, exploiting concurrency, and with numbers. TLplan is regularly used by graduate students, who have constructed representations of a wide variety of problem domains and achieved successful planning performance with the system, demonstrating that domain construction is not a skill restricted to a small elite. Nevertheless, the burden of modelling domains for TLplan has not been properly explored and it is interesting to observe that Bacchus

 © Novática

and Ady were unable to complete an encoding of the final domain for the third IPC due to time constraints.

4.7. HTN Planning

HTN planning has been an active area of research for more than a decade, with influential systems including Tate's O-Plan system [Currie/Tate 1991] [Drabble/Tate 1994] and Wilkin's SIPE [Wilkins 1988]. SHOP2 [Nau et al. 1999] [Nau et al. 2001] represents a particularly straightforward implementation of an HTN system. In contrast to the approach adopted in the planners that use hand-coded control rules, HTN planners cannot rely on the primitive actions available in the domain, alone, but must have these supplemented with constructions – abstract actions – that represent sequences of these primitives. The use of a hierarchy of abstractions allows alternative choices about the precise sequence of primitive actions that is used to realise a given abstract action, by offering different expansions for abstract tasks. Thus, the search problem confronted by HTN planners is not the typical search between alternative primitive actions that is managed by other planners, but a choice between the particular refinements used to realise abstract tasks.

HTN planning is a confounding mixture: on the one hand it is highly intuitive to exploit abstraction in planning, simplifying a planning problem by identifying strategic structures to solutions before attempting to refine them into detailed activity. On the other hand, it is a common criticism of HTN systems that it seems that most of the planning problem is solved by the human domain engineer and that the "planner" is relegated to a purely administrative role of coordinating the recovery of plan components from a database. This is a simplistic view – domain constructions for SHOP are demanding to construct, requiring knowledge of both the planner and the domain, but nevertheless, the domain encodings represent specialised *programs* and the planner is certainly required to execute more complex activity than pure database look-up.

What is frustrating to many researchers is that the intuitive appeal of abstraction as a route to efficient planning seems to be submerged under a significant amount of problem-specific specialised techniques, some of which are not at all intuitive and do not fit easily into the HTN machinery. For example, managing the order in which goals are considered is an important element of the efficient heuristic control of all planners, including HTN planners. The mechanism by which this is handled in SHOP is by defined artificial extensions to the purely physical elements of the original problem that encode goal agenda management actions and the organisation of the agenda is handled by use of these problem-specific actions.

The exploitation of abstraction remains a controversial and tantalising theme in planning research: it seems very reasonable that some form of abstraction should help to make planning more efficient, but there remains work to be done to identify mechanisms for doing so that are flexible and convenient to exploit while realising their promise of efficiency gains.

5 Modern Planning Systems

In this section we will review some of the modern planning approaches that currently seem particularly promising and that exhibit particularly interesting features. The enriched expressive power of PDDL2.1 [Fox/Long 2002], used in the 2002 planning competition (the third IPC), makes the range of problems that can be modelled far more interesting and far closer to real applications than previously. Some of the recently emerging planning approaches are capable of handling problems requiring at least the expressive power of PDDL2.1.

A search strategy that has recently entered the planning repertoire is local search. The planner PbR [Ambite/Knoblock 1997] was one of the first planners to demonstrate that plans could be obtained by means of iterative repair performed on a flawed initial plan, using rewriting rules selected by efficient local search techniques. ASPEN [Chien et al. 1999] [Rabideau et al. 1999] also constructs plans by means of local search-based iterative repair. This idea is explored in LPG [Gerevini/Serina 2002] using locally extending plan graphs. LPG does not handle arbitrary use of numbers, but manages actions with duration and concurrency. LPG replaces the Graphplan search with a far more efficient and more powerful local search technique. This involves identifying an initial candidate "plan" (which might, in fact, not even be executable) and refining it by generating alternative possible repairs or modifications to the candidate. Evaluation of alternative refinements is carried out by a heuristic evaluation function that is partially based on the relaxed plan heuristics used in HSP and FF. LPG demonstrates that there is significant potential in this technique coupled with techniques capable of exploiting the reachability structure of planning problems.

Heuristic search planning remains a powerful strategy. An extended version of FF, metric-FF [Hoffmann 2002], can handle numeric expressions and typically generates good quality solutions while maintaining its speed. The extension of the heuristic evaluation function to handle numbers essentially involves estimating how actions can contribute towards moving a number-valued expression towards its target threshold value. In the same way that the destructive propositional effects of actions are ignored in generating the relaxed plan heuristic estimates of distances to goals, so in the metric version of FF the "destructive" effects of actions that move numeric values in the "wrong" direction are ignored in making the heuristic estimates of numbers of actions required to complete a plan. This approach relies on numeric expressions being linear forms. Although it performs extremely well in many bench mark problems it does not perform well in problems in which the numeric expressions represent complex interlocked resource management behaviour and it seems likely that the relaxation of propositional parts of a plan is more widely applicable than the relaxation of the numeric parts. FF does not handle actions with duration and its current architecture does not obviously generalise to support construction of plans with concurrency.

An interesting system, which extends the notion of forward search to planning in temporal and resource intensive domains, is SAPA [Binh/Kambhampati 2001]. SAPA uses a heuristic evaluation function based on a relaxed temporal plan, together

© Novática

with a technique for estimating the implied cost associated with supplying necessary resources for the relaxed plan. The language SAPA uses allows the representation of actions with duration, the effects of which can happen at arbitrary points within the durative interval. The current implementation of SAPA allows effects to be associated only with the end points of these intervals, because computing interactions between arbitrary time points is computationally expensive and complicates the reasoning mechanisms of the planner.

Most of the fully automatic planning systems that have been developed in the past five years rely on grounding of actions. As previously mentioned, although this might seem excessively memory-intensive, in practice, using compression techniques, it has proved possible to manage large numbers of actions. However, there is clearly a question about the scalability of this technique in the face of problems with large numbers of objects. Although such problems might not seem likely to arise in practice, in fact, where a domain requires ranges of numbers as objects there can be very large collections of objects in a problem instance. One planner in the third IPC, VHPOP [Younes/Simmons 2002], demonstrated an interesting approach to compromising between grounding, and the potential for efficiency it offers, and leaving parameters unbound, which offers the benefits of far more compact representation.

6 Bridging the Gap between Research and Application

Planning technology has been applied to a wide variety of problems. For example manufacturing processes [Nau et al. 1995] [Gupta 1998], satellite and spacecraft operations planning [Muscettola 1994] [Muscettola et al. 1998] [Smith et al. 1999], bridge play [Smith et al. 1998], chemical plant start-up planning [Aylett et al. 1998], elevator scheduling [Koehler 1998], evacuation planning [Muñoz-Avila et al. 2001] represent only some of problems to which planning has been successfully applied. However, applied planning systems are knowledge-intensive systems, requiring significant technical input from both domain-experts and, perhaps primarily, from planning experts. To make planning a more accessible and widely used technology, domain-independent systems offer a route past the bottleneck of planning system expertise. In this section we consider the extent to which domain-independent planning technology is reaching a maturity sufficient for broader and realistic application.

Simplifying assumptions of classical planning, described in section 3, have been progressively relaxed in modern systems. One of the stated goals of the third IPC was the encouragement of a broader commitment to more sophisticated planning capabilities. The development of a much richer expressive power for logical preconditions than pure STRIPS was considered from relatively early days [Pednault 1989] and achieved in least-commitment planners [Blum/Furs 1995], as well as more recent systems such as the Graphplan-based IPP [Koehler et al. 1997] and heuristic-forward-search planner FF. Preprocessing of domain encodings has proved a powerful and successful approach to managing this form of extended expressiveness. This power was tested in a range of systems in the first and, particularly, the second IPCs. In the second IPC, a domain based on the scheduling of elevator use (Miconics-10 [Koehler 1998]) exercised a wide range of the expressive power of this fragment of PDDL.

In the most recent competition, the expressive power of PDDL has been extended with the explicit intention of breaching some of the traditional restrictions of classical planning. In particular, domains used in the third IPC make use of numerically measured resources and other numeric values. They also model the temporal structure of actions in the domains, including the duration of actions that can be executed. The inclusion of actions with duration implicitly introduces the need for planners that can manage and exploit concurrency, including recognising harmful interactions between concurrent activities even if they simply overlap, rather than synchronise their start or end points. Furthermore, plan-metrics have been added to the language, so that it is possible to identify how plans should be evaluated. The impact of this extension should not be underestimated: it offers the power to harness planning for practical use in a way that is simply impossible if the only measure of plan quality is the number of steps in the plan. In almost every real planning application the cost of resource consumption, including the actual time over which the plan is executed, possibly offset by the profit generated by the plan, is an essential measure of plan quality, while number of steps is of limited interest.

These extensions in expressive power made it possible to introduce several domains in the third IPC that make an inter-

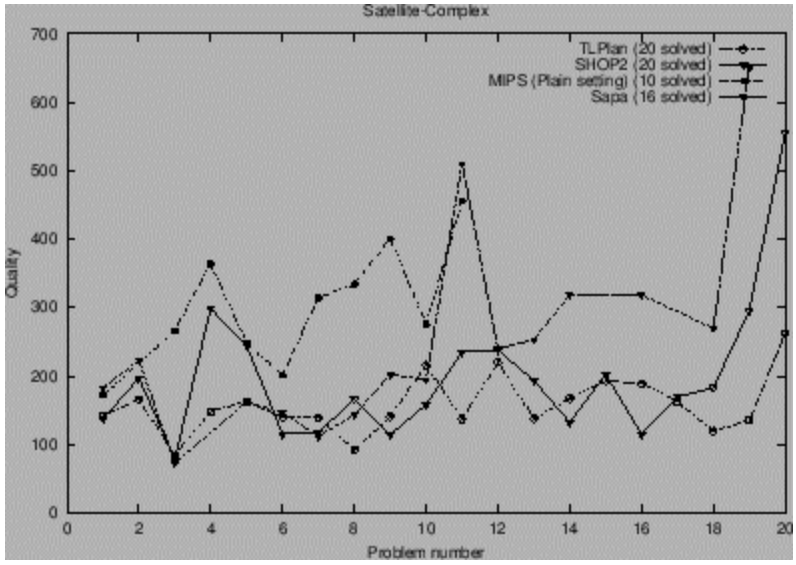

Figure 5: Complex Satellite variant, plan quality: smaller values are better plans.

 © Novática

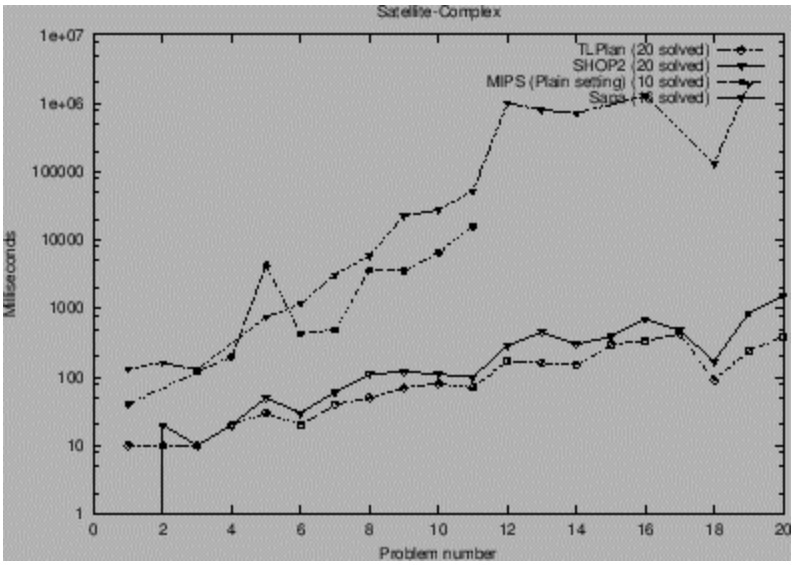

Figure 6: Complex Satellite variant, planning speed: note log-scale.

esting and convincing step towards real application domains. One domain extends the logistics domain by the addition of time for travel between locations, time for loading and unloading and fuel consumption. With these additions, plan quality can be measured as a tradeoff between the total time required to complete all the deliveries and the fuel consumed in doing so. The extended domain also includes alternative means of travel, offering a choice between slow, fuel-efficient travel and faster, fuel-hungry travel. This choice is an important one when the plan is to be evaluated using combined costs of duration and fuel consumption.

A second realistic domain is that of planetary exploratory-rovers, based on the Mars exploratory rovers mission due for 2003 launch. In the competition model of this domain rovers are equipped with different equipment and have different capabilities for traversal of the terrain. The rovers must collect data and downlink it to a lander (for subsequent retransmission to earth). Terrain blocks communication between certain locations, making it necessary to plan how to efficiently visit experiment sites, store data and communicate it after moving to a site from which transmission to the lander is possible. Rovers consume energy during their activities and must recharge to remain active. Plan costs are measured in terms of the time and energy spent acquiring the data, making efficient concurrent use of rovers important for plan quality.

A third domain introduced is modelled after the problem of satellite observation scheduling [Smith et al. 2000]. This problem involves using a collection of differently equipped satellites to record observations, storing the data in a finite capacity on-board store for subsequent downlink. Careful choice of calibration targets to be used in setting up instruments is an important aspect of this problem, as well as efficient management of the data store and of fuel in slewing the satellite between observation and calibration targets. One version of this domain used in the competition required planners to solve the problem of maximising acquired data, without being presented with any logical goals at all. This is a dramatic contrast to the classical planning problem formulation and demonstrates one of the ways in which plan metrics stretch the scope of the planning problem.

Although there remain aspects of the real problems on which these domains are modelled that are not yet captured, or are even impossible to capture within the PDDL language as it currently stands, the performance of planners on these problems and the sophistication of the models demonstrates the distance that planning has progressed in recent years. Some of the outstanding issues are discussed in section 7. We end this section by considering data demonstrating the performance of planners in the third IPC on some of the problems in various versions of the satellite domain.

Figure 5 shows plans produced for a variant of the satellite domain using both numeric and temporal features. Here the objective was to minimize the time for execution of the plan. Figure 6 shows how fast these plans were produced (on a 1800MHz Athlon CPU PC, with 1Gb RAM), while Figure 7 shows that these plans contain well over 100 steps in some cases. Note that the planners using hand-coded controls produced longer plans, reflecting the exploitation, by the domain-engineers, of certain sequences of actions that do not obey a "triangle inequality": for some actions A, B and C,

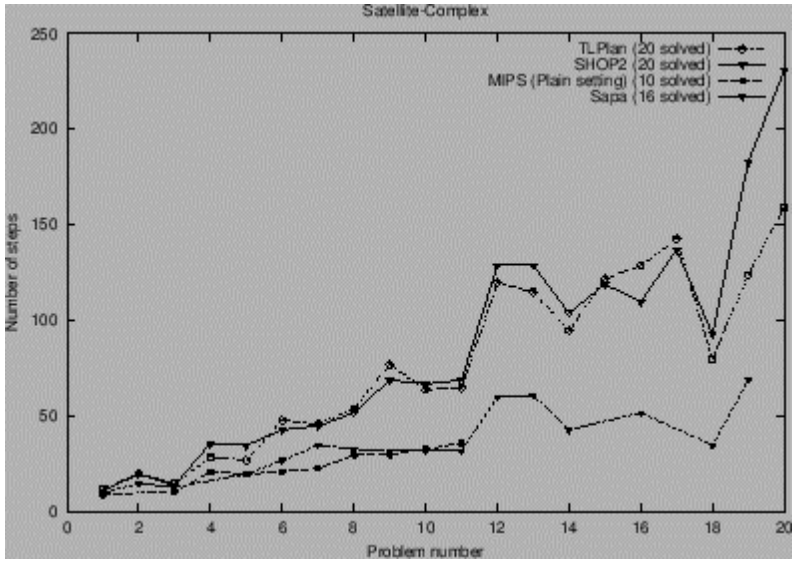

Figure 7: Complex Satellite variant, plan length. This plot shows the size of the plans produced, placing the planning times in context. Note that the fully-automated planners typically produce plans with fewer steps.

© Novática

actions A and B achieve a goal that is otherwise achieved by action C, but the sum of the lengths of actions A and B is smaller than the length of action C. The automatic planners tend to favour selecting single action solutions to goals when they can, even though this is not always optimal.

7 Issues for Further Research

As we have seen, planning has moved on a long way from its STRIPS roots. Planners can now handle problems with time and numbers, allowing the expression of complex mixed scheduling and planning problems, demanding concurrency and resource management. In addition, the modelling language allows expression of plan metrics, so that planners can seek to optimise the plans they produce against a more useful measure of value than simple plan length. In the 2002 planning competition the use of this metric confirmed that several planners can tailor their performance towards production of plans of higher quality. Using the plan metric it is even possible to construct planning problems in which the objective is not to achieve any particular logical goal state, but to optimise the value of some measure of utility subject to the selection of actions forming the plan. For example, in the competition a problem was posed in which satellite observations should be collected into finite capacity stores on board satellites. No specific observation targets were required to be stored, but the value of the plan was determined by the total amount of stored data achieved by the plan. This kind of problem represents a radical new challenge for the planning community. MIPS [Edelkamp 2002] was the only fully automated planner that managed to produce plans that acquired data. The hand-coded systems, TLplan and SHOP2, both produced very high-quality plans, apparently close to optimal in the majority of problems; but this performance relied on a human expert encoding the problem so that numeric rather than logical effects would be prioritized in the search process. The use of plan metrics is clearly an important challenge for the further development of fully automated planning systems, perhaps being used to supplement the guidance heuristic evaluation functions used by heuristic search planners.

The existing PDDL standards do not support the modelling of *exogenous events*. As discussed earlier, classical planning makes the assumption that no change can occur that is not under the direct control of the planner. However, in many realistic situations it is necessary to plan around the fact that uncontrollable changes will occur. For example, in the full satellite observation planning problem, it is vital to represent the fact that opportunities for making observations and for downlinking data both arise in time windows that are not under the control of the executive (and, therefore, cannot be planned by the planner), since they arise as a consequence of the process of orbiting the earth and the events of transition above or below the horizon for each opportunity location that are caused by this process. Some planners in the application-oriented tradition [Muscettola 1994] [Laborie/Ghallab 1995] are capable of planning with foreknowledge of such events. When such events can be anticipated they can be encoded as constraints which can be thrown in with all the other constraints that describe a problem and then any solution that emerges will respect the restrictions that they impose. However, planning strategies not based on reformulation of the problem into a CSP have not yet successfully tackled planning with predictable exogenous events.

When exogenous events are not entirely predictable they give rise to uncertainty in the planning problem. Although many researchers have considered the problem of planning under uncertainty in various forms, there remain many questions to resolve. There is not yet a clear consensus even on the form of a plan when managing uncertainty. It seems that uncertainty arises in several different forms. Unpredictable exogenous events give rise to a form which can be difficult to plan with – if unexpected changes are occurring frequently the resulting uncertainty can undermine any planning effort. More benign forms of uncertainty also exist. For example, there is uncertainty about the precise duration of actions, or consumption of resources such as fuel by those actions. This uncertainty typically can be described by continuous probability distributions, often normal or close to normal. This form of uncertainty might be considered benign in that plans can be made increasingly robust to the uncertainty by allowing more resources for their execution. Uncertainty about the successful outcome of the execution of an action might be best described by a discrete probability distribution. This form of uncertainty is more difficult to manage because it leads to a significant branching in the possible states of the world after execution, making it very hard to produce a robust plan without introducing contingent actions.

In addition to uncertainty, many real domains demand that a planner should manage complete ignorance. In this case, the executive will typically have access to some form of information gathering actions and the planner can plan to acquire information and react to the results. In this situation, and also in the case of handling uncertainty, it is often the case that to plan for long sequences of activity is either impossible or else a poor investment of effort. A more useful approach to problem solving in this case is to interleave planning and execution, using execution-monitoring, failure diagnosis and plan repair techniques to resolve the problems that arise during this process. Continuous planning [Knight et al. 2001], in which new goals can arise as a consequence of discoveries made at execution time, is an important development of planning taking it further in the direction of providing the basis for autonomous behaviour. These are all areas of active research, but there is no commonly accepted empirical framework for evaluating such systems, or even for describing problems that could be shared across the community. Putting problems such as these onto the agenda for the whole planning community is an important role that can be played by combinations of the competition series, the reporting of high-profile application areas demanding these kinds of functionality and the continual striving of the community as a whole to extend and develop the technology at its core.

8 Conclusion

Planning is a hard problem, even when reduced to the bare form of the classical planning formulation. The hardness of this problem has confounded rapid development in the planning

 © Novática

research community for a long time, but over recent years the field has seen significant advances. This progress has allowed the restrictions of classical planning to be relaxed or overcome. In turn, this has brought extended expressive power and domain-independent planning is now a technology capable of application to realistic domains.

Over the past decade several new techniques have emerged as promising for future development of classical planning towards application. In particular, these include heuristic forward search based on informative heuristics that can be automatically generated and that exploit the structure of the problem; local search techniques that can be applied to efficiently repair a rapidly generated, flawed plan; hybrid systems that exploit structure and systems that go beyond the simple notion of batch planning and can anticipate execution-time discoveries. In parallel there has been increasing interest in planners that can reason about expert knowledge of a problem, expressed in terms of search control rules or in terms of skeletal plan structure or HTN-style action decompositions. The gap between classical planning and application-oriented planning is narrowing and the scope for exploitation of planning technology in industry, commerce and scientific application contexts is continuously increasing.

References

[Ambite/Knoblock 1997]
J.L. Ambite and C.A. Knoblock. Planning by rewriting: efficiently generating high-quality plans. In Proceedings of 14th National Conference on AI (AAAI), 1997.

[Aylett et al. 1998]
R. Aylett, J. Soutter, G. Petley, P.W.H. Chung, and A. Rushton. AI planning in a chemical plant domain. In Proceedings of European Conference on AI ECAI 98, 1998.

[Bacchus 2000]
F. Bacchus. The 2nd International Planning Competition home page. <http://www.cs.toronto.edu/aips2000/>, 2000.

[Bacchus/Kabanza 2000]
F. Bacchus and F. Kabanza. Using temporal logic to express search control knowledge for planning. Artificial Intelligence, 116(1–2):123–191, 2000.

[Binh/Kambhampati 2000]
M. Binh Do and S. Kambhampati. Solving planning graph by compiling it into a CSP. In Proc. of 5th Conference on AI Planning Systems, pages 82–91. AAAI Press, 2000.

[Binh/Kambhampati 2001]
M. Binh Do and S. Kambhampati. Sapa: a domain-independent heuristic metric temporal planner. In Proc. ECP-01, 2001.

[Blum/Furs 1995]
A. Blum and M. Furst. Fast Planning through Plan-graph Analysis. In Proc. of 14th Internation Joint Conference on AI, pages 1636–1642. Morgan Kaufmann, 1995.

[Bonet/Geffner 1997]
B. Bonet and H. Geffner. Planning as heuristic search: new results. In Proc. of 4th European Conference on Planning (ECP). Springer-Verlag, 1997.

[Bonet et al. 1997]
B. Bonet, G. Loerincs, and H. Geffner. A robust and fast action selection mechanism for planning. In Proc. of 14th National Conference on AI, pages 714–719. AAAI/MIT Press, 1997.

[Borrett/Tsang 2001]
J. Borrett and E.P.K. Tsang. A context for constraint satisfaction problems formulation selection. Constraints, 6, No.4:299–327, 2001.

[Chien et al. 1999]
S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau. Integrated planning and execution for autonomous spacecraft. In Proceedings of the IEEE Aerospace Conference (IAC), 1999.

[Cimatti/Roveri 1999]
A. Cimatti and M. Roveri. Conformant planning via model checking. In Proceedings of the European Conference on Planning (ECP99), 1999.

[Clark 2001]
M. Clark. Construction domains: a generic type solved. In Proc. of 20th UK Planning and Scheduling Workshop, Edinburgh, 2001.

[Currie/Tate 1991]
K. Currie and A. Tate. O-plan: the open planning architecture. Artificial Intelligence, 52(1):49–86, 1991.

[Drabble/Tate 1994]
B. Drabble and A. Tate. The use of optimistic and pessimistic resource profiles to inform search in an activity based planner. In Proc. of 2nd Conference on AI Planning Systems (AIPS). AAAI Press, 1994.

[Edelkamp 2002]
S. Edelkamp. Mixed propositional and numeric planning in the model checking integrated planning system. In M. Fox and A. Coddington, editors, Planning for Temporal Domains: AIPS'02 Workshop, 2002.

[Edelkamp/Helmert 2000]
S. Edelkamp and M. Helmert. On the implementation of mips. In Proceedings of Workshop on Decision-Theoretic Planning, Artificial Intelligence Planning and Scheduling (AIPS), pages 18–25. AAAI-Press, 2000.

[Erol et al. 1995]
K. Erol, D. Nau, and V.S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning. Artificial Intelligence, 76(1–2):75–88, 1995.

[Fikes/Nilsson 1971]
R.E. Fikes and N.J. Nilsson. STRIPS: A New Approach to the Application of Theorem-Proving to Problem-Solving. Artificial Intelligence, 2(3):189–208, 1971.

[Fox/Long 1998]
M. Fox and D. Long. The automatic inference of state invariants in TIM. Journal of AI Research, 9:367–421, 1998.

[Fox/Long 2000]
M. Fox and D. Long. Utilizing automatically inferred invariants in graph construction and search. In Proc. of 5th Conference on Artificial Intelligence Planning Systems (AIPS), Breckenridge, Colorado. AAAI Press, 2000.

[Fox/Long 2001]
M. Fox and D. Long. Hybrid STAN: Identifying and Managing Combinatorial Sub-problems in Planning. In Proc. of 17th International Joint Conference on AI, pages 445–452. Morgan Kaufmann, 2001.

[Fox/Long 2002]
M. Fox and D. Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. Technical Report Department of Computer Science, 20/02, Durham University, UK. Available at: <http://www.dur.ac.uk/d.p.long/competition.html>. Forthcoming in revised form as a JAIR article, 2002.

[Gerevini/Schubert 1998]
A. Gerevini and L. Schubert. Inferring state constraints for domain-independent planning. In Proc. of 16th National Conference on AI, pages 905–912. AAAI/MIT Press, 1998.

[Gerevini/Serina 2002]
A. Gerevini and I. Serina. LPG: A planner based on local search for planning graphs. In Proc. of 6th International Conference on AI Planning Systems (AIPS'02). AAAI Press, 2002.

[Green 1969]
C. Green. Theorem proving by resolution as a basis for question-answering systems. In B. Meltezer, D. Michie, and M. Swann,

© Novática

editors, Machine Intelligence, volume 4. Edinburgh University Press, 1969.

[Gupta 1998]
S.K. Gupta, D.S. Nau, and W.C. Regli. IMACS: A case study in real-world planning. IEEE Expert and Intelligent Systems, 13(3):49–60, 1998.

[Hoffmann 2002]
J. Hoffmann. Extending FF to numerical state variables. In Proceedings of European Conference on AI (ECAI'02), pages 571–575, 2002.

[Hoffmann/Nebel 2000]
J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. Journal of AI Research, 14:253–302, 2000.

[Hölldobler/Störr 2000]
S. Hölldobler and H-P. Störr. Solving the entailment problem in the fluent calculus using binary decision diagrams. In Workshop on Model-Theoretic Approaches to Planning at AIPS2000. Beckenridge, 2000.

[Ingham et al. 2001]
M. Ingham, R. Ragno, and B.C. Williams. A reactive model-based programming language for robotic space explorers. In Int. Symp. on Artificial Intelligence, Robotics and Automation in Space, 2001.

[Kautz/Selman 1995]
H. Kautz and B. Selman. Unifying SAT-based and graph-based planning. In Proc. of 14th Internation Joint Conference on AI, pages 318–325. Morgan Kaufmann, 1995.

[Kautz/Selman 1998]
H. Kautz and B. Selman. The role of domain-specific axioms in the planning as satisfiability framework. In Proc. of 4th Conference on AI Planning Systems, Pittsburgh, PA, pages 181–189. AAAI Press, 1998.

[Knight et al. 2001]
R. Knight, G. Rabideau, S. Chien, B. Engelhardt, and R. Sherwood. Casper: Space exploration through continuous planning. IEEE: Intelligent Systems, 16(5):70–75, 2001.

[Koehler 1998]
J. Koehler. Planning under resource constraints. In Proc. of 13th European Conference on AI, pages 489–493, 1998.

[Koehler et al. 1997]
J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos. Extending planning graphs to an ADL subset. In Proc. of 4th European Conference on Planning, Toulouse, pages 273–285, 1997.

[Kvarnstrom/Doherty 2000]
J. Kvarnstrom and P. Doherty. TALplanner: A temporal logic based forward chaining planner. Annals of Mathematics and Artificial Intelligence, 30(1–4):119–169, 2000.

[Kvarnstrom et al. 2000]
J. Kvarnström, P. Doherty, and P. Hasslum. Extending TALplanner with concurrency and resources. In Proceedings ECAI-00, Berlin, Germany, August, 2000.

[Laborie/Ghallab 1995]
P. Laborie and M. Ghallab. Planning with sharable resource constraints. In Proc. of 14th International Joint Conference on AI. Morgan Kaufmann, 1995.

[Long/Fox 1999]
D. Long and M. Fox. The efficient implementation of the plan-graph in STAN. Journal of AI Research, 10, 1999.

[Long/Fox 2000]
D. Long and M. Fox. Automatic synthesis and use of generic types in planning. In Proc. of 5th Conference on Artificial Intelligence Planning Systems (AIPS), pages 196–205. AAAI Press, 2000.

[Long/Fox 2001]
D. Long and M. Fox. Multi-processor scheduling problems in planning. In Proc. of International Conference on AI (IC-AI), Las Vegas, 2001.

[Long/Fox 2002]
D. Long and M. Fox. Reformulation in planning. In Proceedings of Symposium on Abstraction, Reformulation and Approximation. LNAI, Springer-Verlag, 2002.

[Long et al. 2000]
D. Long, M. Fox, L. Sebastia, and A. Coddington. An examination of resources in planning. In Proc. of 19th UK Planning and Scheduling Workshop, Milton Keynes, 2000.

[McCarthy/Hayes 1969]
J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, Machine Intelligence 4, pages 463–502. Edinburgh University Press, 1969. reprinted in McC90.

[McDermott 1998]
D. McDermott. PDDL – the planning domain definition language. Technical report, Yale University, <http://www.cs.yale.edu/users/mcdermott.html>, 1998.

[McDermott 1996]
Drew McDermott. A heuristic estimator for means ends analysis in planning. In B. Drabble, editor, Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-96), pages 142–149. AAAI Press, 1996.

[Muñoz-Avila et al. 2001]
H. Muñoz-Avila, D.W. Aha, D.S. Nau, R. Weber, L. Breslow, and F. Yaman. SiN: Integrating case-based reasoning with task decomposition. In Proceedings of International Joint Conference on AI (IJCAI-2001), 2001.

[Muñoz-Avila 2002]
L. Murray. Reuse of control knowledge in planning domains. In L. McCluskey, editor, Knowledge Engineering Tools and Techniques for AI Planning: AIPS'02 Workshop, 2002.

[Muscettola 1994]
N. Muscettola. HSTS: Integrating planning and scheduling. In M. Zweben and M.S. Fox, editors, Intelligent Scheduling, pages 169–212. Morgan Kaufmann, San Mateo, CA, 1994.

[Muscettola et al. 1998]
N. Muscettola, P. Pandurang Nayak, B. Pell, and B.C. Williams. Remote agent: To boldly go where no AI system has gone before. Artificial Intelligence, 103(1–2):5–47, 1998.

[Nau et al. 1999]
D. Nau, Y. Cao, A. Lotem, and H. Muñoz-Avila. SHOP: Simple hierarchical ordered planner. In Proc. of 16th International Joint Conference on AI, pages 968–975. Morgan Kaufmann, 1999.

[Nau et al. 1995]
D. Nau, S. Gupta, and W. Regli. Artificial intelligence planning versus manufacturing-operation planning: a case study. In Proc. of 14th International Joint Conference on AI, pages 1670–1676. Morgan Kaufmann, 1995.

[Nau et al. 2001]
D. Nau, H. Muñoz-Avila, Y. Cao, A. Lotem, and S. Mitchell. Total-order planning with partially ordered subtasks. In Proceedings of International Joint Conference on AI (IJCAI-2001), 2001.

[Newell/Simon 1963]
A. Newell and H. Simon. GPS, a program that simulates human thought. In E.A. Feigenbaum and J. Feldman, editors, Computers and Thought. McGraw Hill, NY, 1963.

[Pednault 1989]
E. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In Proc. of 1st International Conference on Principles of Knowledge Representation and Reasoning, pages 324–332. San Francisco, CA, Morgan Kaufmann, 1989.

[Penberthy/Weld 1992]
J. Penberthy and D.S. Weld. UCPOP: a sound, complete, partial-order planner for ADL. In Proc. Int. Conf. On Principles of Knowledge Representation and Reasoning, pages 103–114, Los Altos, CA, 1992. Kaufmann.

 © Novática

[Porteous 2001]
J. Porteous, L. Sebastia, and J. Hoffmann. On the extraction, ordering, and usage of landmarks in planning. In Proceedings of European Conference on Planning ECP'01, 2001.

[Rabideau et al. 1999]
G. Rabideau, R. Knight, S. Chien, A. Fukunaga, and A. Govindjee. Iterative repair planning for spacecraft operations in the ASPEN system. In International Symposium on Artificial Intelligence Robotics and Automation in Space (ISAIRAS), 1999.

[Russell/Norvig 1995]
S. Russell and P. Norvig. Artificial Intelligence: a Modern Approach. Prentice Hall, 1995.

[Smith et al. 1999]
B. Smith, W. Millar, J. Dunphy, Y. wen, P. Nayak, E. Jr, and M. Clark. Validation and verification of the remote agent for spacecraft autonomy. In Proceedings of the IEEE Aerospace Conference, Snowmass, CO., 1999.

[Smith et al. 2000]
D.E. Smith, J. Frank, and A.K. Jónsson. Bridging the gap between planning and scheduling. Knowledge Engineering Review, 15(1), 2000.

[Smith et al. 1998]
S.J.J. Smith, D.S. Nau, and T. Throop. Computer bridge: A big win for AI planning. AI Magazine, 19(2):93–105, 1998.

[Srivastava 2000]
B. Srivastava. RealPlan: Decoupling causal and resource reasoning in planning. In Proc. of 17th National Conference on AI, pages 812–818. AAAI/MIT Press, 2000.

[Stefik 1981]
M. Stefik. Planning with constraints (MOLGEN: Parts 1 and 2). Artificial Intelligence, 16(2):111–170, 1981.

[Tsang 1993]
E.P.K. Tsang. Foundations of Constraint Satisfaction. Academic Press, London and San Diego, 1993.

[vanBeek/Chen 1999]
P. van Beek and X. Chen. CPlan: A constraint programming approach to planning. In Proc. of 16th National Conference on Artificial Intelligence, pages 585–590. AAAI/MIT Press, 1999.

[Vossen et al. 1999]
T. Vossen, M. Ball, A. Lotem, and D. Nau. On the use of integer programming models in AI planning. In Proceedings of International Joint Conference on AI, (IJCAI-99), pages 304–309, 1999.

[Westfold/Smith 2001]
S.J. Westfold and D.R. Smith. Synthesis of efficient constraint satisfaction programs. Knowledge Engineering Reviews: Special Issue on AI and OR, 2001.

[Wilkins/desJardins 2000]
D. Wilkins and M. desJardins. A call for knowledge-based planning. In Proc. of AI Planning and Scheduling (AIPS) Workshop on Analyzing and Exploiting Domain Knowledge for Efficient Planning, 2000.

[Wilkins 1988]
D.E. Wilkins. Practical Planning: Extending the Classical AI Planning Paradigm. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1988.

[Wolfman/Weld 1999]
S. Wolfman and D. Weld. The LPSAT engine and its application to resource planning. In Proc. of 16th International Joint Conference on AI, pages 310–317. Morgan Kaufmann, 1999.

[Younes/Simmons 2002]
H.L.S. Younes and R.G. Simmons. On the role of ground actions in refinement planning. In Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling Systems, pages 90–97, 2002.

© Novática