The AIPS-98 Planning Competition

Edited by Derek Long, with Henry Kautz and Bart Selman (BLACKBOX team); Blai Bonet and Hector Geffner (HSP team); Jana Koehler, Michael Brenner, Joerg Hoffmann, and Frank Rittinger (IPP team); Corin R. Anderson, Daniel S. Weld, and David E. Smith (sGP team); and Maria Fox and Derek Long (sTAN team)

In 1998, the international planning community was invited to take part in the first planning competition, hosted by the Artificial Intelligence Planning Systems Conference, to provide a new impetus for empirical evaluation and direct comparison of automatic domain-independent planning systems. This article describes the systems that competed in the event, examines the results, and considers some of the implications for the future of the field.

The International Artificial Intelligence Planning Systems Conference (AIPS-98), L held at Carnegie Mellon University in Pittsburgh in June 1998, played host to the first world planning competition. Competitors were invited to come and compete on a collection of domains and associated problems, sight unseen, using whatever planning technology they wanted. Tracks were offered for STRIPS, ADL, and HTN planning, but in the event, only STRIPS and ADL were entered. Indeed, ADL saw only two competitors: IPP (Koehler et al. 1997) and SGP (Anderson and Weld 1998). IPP gave a convincingly superior performance over SGP, the only Lisp-based planner in the competition, in a single round playoff. The STRIPS track originally attracted some 9 or 10 declarations of intent to take part. This article gives an account of the competition as seen by the competitors who actually arrived in Pittsburgh and took part in the two tracks.

Competitions as a way to evaluate and promote progress in various fields have precedents, such as the Message Understanding Competition (MUC) series, sponsored by the Defense Advanced Research Projects Agency (DARPA); the Text Retrieval Competition (TREC) series, sponsored by National Institute of Standards and Technology and DARPA; and the Turing Competition. These competitions have stimulated work, but they also represent a serious investment in effort for the competition organizers and competitors. It is a formidable task to create a collection of tasks that is realistically within the reach of the existing technology and that represents an adequate challenge and points the way for the field to develop. Administrative problems represent a huge overhead to this task: A common language must be developed that allows problems to be specified and results evaluated, scoring mechanisms must be determined, and the environment must be selected and competitors forewarned. Tribute should be paid to Drew McDermott for the role he played in almost single-handedly executing all these tasks, with support from the competition committee.

For the competitors, the competition represents a challenge to the robustness of their software, and demands work in meeting the specifications for both input and output formats, while they continue to develop and enhance the basic functions of their systems. The development of PDDL (McDermott and AIPS 1998) as the common language for the competition problem specifications was an important step in the progress of the competition. A challenge to the competitors was to the collection of planners that are sufficiently robust for release into the general research *community*, requiring no additional specialpurpose domain encoding beyond STRIPS or ADL, and that can tackle significant instances in reasonable time is still small.

adapt to the many minor changes in this language as it steadily stabilized. Another important problem was anticipating the demands of the competition domains. All the planners that eventually competed are domain-independent planners that require little or no manual guidance in selecting run-time behavior; however, the performance of all the planners can be affected dramatically by the design of the encoding of the domain and problem specifications. The criteria by which success was to be judged were also volatile: Trade-offs between the planning time and the optimality of the plan produced were a controversial balancing act. Optimality was measured purely by the number of steps in the plan for the STRIPS track; so, planners producing optimal parallel plans might well find themselves significantly outperformed by planners concentrating on sequential plan optimality. Furthermore, the selection of domains to be used in the competition was hard. All the competitors had some collection of favored domains that showcased the characteristics of their planners. In principle, all competitors had the opportunity to propose domains for use, but the constraints on the time available for the competition made it impossible to use all of them.

These various challenges thinned the field so that the competition eventually hosted four STRIPS planners and two ADL planners (IPP took part in both tracks). It was apparent that several others would liked to have participated but were unable to meet the input and output criteria within the time scales that were imposed by the organization of the competition.

This article presents the competitors' impressions of the competition and a summary of the planning technology with which they competed. The Competing Planners includes a series of short subsections that highlight features of the individual planners that took part and briefly explores the issues that have led to the differences in the systems that competed. Review of Results examines the results of the competition, analyzing the issues they raise and pointing out some of the questions that they leave unanswered. Finally, The Future considers the future of the competition and the role it might play in pushing forward planning research.

The Competing Planners

The competition highlighted several facets of the current state of the planning research field. First, even within the last couple of years, the technology has advanced dramatically in terms of the size of problems in standard benchmark domains that planners can realistically be expected to handle. All the planners in the competition were solving some instances of problems in times measured in milliseconds, including nontrivial instances. Second, the collection of planners that are sufficiently robust for release into the general research community, requiring no additional special-purpose domain encoding beyond STRIPS or ADL, and that can tackle significant instances in reasonable time is still small. The qualifications are important, however, because there are clearly many more planners being actively worked on that are either still insufficiently robust to be entered into a competition event or require too much careful encoding of their domains to be able to compete automatically across large numbers of problems, sight unseen. It is hoped that this situation changes as the competition is repeated in future years, with a widening of the field and a broadening of the range of technology being exposed. This latter point is particularly poignant because at AIPS-98, three of the five planners involved were directly built ON GRAPHPLAN (Blum and Furst 1997) (IPP, SGP, and STAN), one exploited GRAPHPLAN technology (BLACKBOX), and only one was independent of GRAPHPLAN in every way (HSP). The role of GRAPH-PLAN in shaping the domain-independentplanning research field at the time of the competition is so significant that a short summary of the features of the system is included in this section.

Interestingly, all five planners used a full instantiation of actions as a prelude to search, explaining in part the difficulties all the planners experienced with problems that involved huge numbers of ground-operator instances. This point is discussed further in the next section. It is unfortunate that no constraint-posting planner, avoiding the initial grounding of action schemas, took part to compare performance. Some informal experiments with PRODI-GY (Fink and Veloso 1996) during the competition suggested that it might have performed significantly better than the other planners in some domains, but it was completely outperformed in others. This pattern hints at one of the most interesting issues that the competition raised (discussed in more detail in the next section): Different planning technologies, despite being domain independent, are actually highly sensitive to both domain and problem instance in determining actual performance.

This section proceeds with a summary of GRAPHPLAN before proceeding with an examination of each of the competing systems in turn. The organization of this sequence reflects the relationship between each system and GRAPH-PLAN. IPP, SGP, and STAN are essentially extensions of GRAPHPLAN, each exploring different directions. BLACKBOX exploits the core GRAPH-PLAN data structure but carries out its search for a plan in an entirely different way. Finally, HSP is an entirely different approach to planning.

GRAPHPLAN

GRAPHPLAN is an exceptionally influential planning system devised by Blum and Furst (1997). Since it was first developed, there have been several insightful papers that seek to explore its relationship with other planning approaches and with constraint solving (Kambhampati 1999; Kambhampati, Lambrecht, and Parker 1997). There has also been considerable activity devoted to extending and improving the underlying algorithms.

GRAPHPLAN constructs plans in three phases: First, the action schemas are fully instantiated to identify the complete set of ground actions for the planning problem. Second, a fundamental data structure, called the *plan graph*, is constructed. Third, the plan graph is searched in an attempt to identify a plan structure within it. If this process fails, the graph is extended, and a new search is initiated. This process iterates until a plan is found, or certain terminating conditions are encountered, proving that there is no plan. The interleaved graph-construction and -search processes amount to an iterated depth-first search for a plan, ensuring a systematic search guaranteed to find the shortest plan structure.

The key to understanding GRAPHPLAN then is to understand the plan-graph data structure. This structure is built as a series of layers of nodes. Nodes in alternate layers represent ground facts and ground actions. Arcs are created between nodes that represent important relationships. Fact nodes are connected to action nodes in the immediately succeeding layer when they represent preconditions of the actions. Actions are connected to the facts in the immediately succeeding layer that they achieve as positive effects. They are also connected (by differently labeled edges) to their negative effects (facts they delete). GRAPHPLAN also inserts special actions, called noops, at each action layer, which have the role of causing facts at the preceding layer to persist into the next layer. Using a noop to achieve a goal is equivalent to achieving the goal early and allowing it to persist while other actions occur. Finally, and most significantly, pairs of actions in the same layer, and pairs of facts in the same layer, are connected if they are mutually inconsistent (or mutually exclusive-"mutex"). Two

actions are considered mutually exclusive if the negative effects of one intersects with the preconditions or positive effects of the other or if they have mutually exclusive preconditions. Facts are mutually exclusive if all pairs of actions that achieve the two facts are themselves mutually exclusive. The graph is seeded with an initial layer representing the facts true in the initial state. The graph is constructed until a layer of facts includes all the plan goals as a collection of nonpairwise mutually exclusive nodes in a layer.

Search is goal directed, starting with the last layer of the graph. The goals are identified in the final layer, and a collection of nonpairwise mutually exclusive actions in the preceding action layer is found that includes all the goals as positive effects. The preconditions of these actions are then used as the seed for a recursive search from the immediately preceding layer of facts. The process successfully identifies a plan if the search reaches a collection of facts in the initial layer. It should be noted that because more than one action can be used in a given layer of the graph, the plan can include parallel actions. Therefore, GRAPHPLAN produces paralleloptimal plans that are not always sequentially optimal. If the search fails to produce a plan (including the recursively initiated searches in preceding layers), the collection of goals that cannot be solved is recorded as unsolvable at the corresponding layer. During subsequent searches, goals are checked against previously recorded unsolvable sets, and if they contain such a set as a subset, then no search is initiated, but a failure is reported immediately. This approach can save a huge amount of wasted search.

If the search fails to find a plan for a top-level goal set, then the graph is extended by another action and fact layer, and the search recommences. It has been proved (Blum and Furst 1997) that certain signals in the data structure will infallibly demonstrate a problem to be unsolvable, so that the algorithm is not only sound and complete but also terminating.

GRAPHPLAN has had a dramatic impact on the planning community, effectively displacing the earlier nonlinear planning technology as the foundation of fast and effective planners. Nevertheless, it suffers from significant problems. The need to instantiate actions, the storage of the plan graph itself, and the records of unsolvable goal sets together represent a formidable demand on memory. It is not uncommon to require over 100 megabytes of randomaccess memory (RAM) to solve significant problems, and memory constraints represent one of the barriers to solving much larger planThe key to understanding GRAPHPLAN then is to understand the plan-graph data structure.



Figure 1. IPP Architecture.

ning problems with this technology. Further, the backward search through ground actions forces the planner to make decisions about parts of a plan long before the information is available to determine how to make the decision intelligently. These issues, among others, remain to be resolved if GRAPHPLAN is to have longevity as a planning solution.

IPP

The IPP (1998) planning system extends the GRAPHPLAN system in several significant ways. In particular, IPP was developed to demonstrate the extension of the GRAPHPLAN approach to allow it to handle a larger subset of ADL (Pednault 1989). These extensions include the use of negative preconditions and conditional effects, allowing IPP to enter the ADL track of the competition. To help in handling the ADL extensions in PDDL, IPP uses a preprocessing phase based on the proposal first documented by Gazen and Knoblock (1997) that translates

PDDL first-order formulas into the disjunctive normal form required by IPP but, in contrast to that approach, does not expand out conditional action effects. Instead, IPP uses a special extension of the GRAPHPLAN machinery to handle conditional effects directly. In addition, IPP represented a first complete reimplementation of GRAPHPLAN and incorporated many major improvements in the underlying data structures. These efforts give IPP a dramatically better time and space performance than the original GRAPHPLAN implementation. A particular improvement is that IPP exploits a new data structure to support the recording of goal sets for which search has failed (Hoffmann and Koehler 1999). This data structure makes it possible for a full subset test to be used efficiently when checking to see if a new goal set includes a subset already recorded as unsolvable, allowing IPP to make more effective use of these records.

A further innovation introduced in the IPP system is the use of a filter to remove facts and actions from the plan graph that are considered irrelevant to the planning problem. This process has the potential to improve the performance of the planning system dramatically, reducing both storage requirements and search time. This system, RIFO (Nebel, Dimopoulos, and Koehler 1997), is heuristic and can be controlled to prune the graph more or less aggressively. The competition version of IPP introduces a control harness that uses various parameters measured from the domain and problem instance to determine how aggressively to apply the RIFO mechanism.

The RIFO Metastrategy The competition version of IPP exploits a metastrategy, "wrapped" around the planner (figure 1), that is activated when two threshold parameters—(1) the number of objects and (2) the number of ground actions—are exceeded by a planning problem. These parameters can be set by hand or can be preset. The strategy configures the RIFO subsystem within IPP.

RIFO was developed to cope with the problem of irrelevant information in plan graphs. Usually, when building a plan graph, a GRAPHPLANbased planner will add a great many facts and actions that are totally irrelevant to solving the planning problem at hand. By building a tree of possible actions backwards from the goal set, ignoring negative effects of actions, RIFO can identify what information is or might be relevant to a problem. This identification allows the system to exclude irrelevant information. The pruning can be applied at various degrees of strength but at the price of increasing danger of incompleteness: Overaggressive pruning can lead to too many actions being ruled out, making the planning problem unsolvable. The metastrategy runs RIFO on the problem input using the strongest reduction heuristic, which selects a minimal set of relevant initial facts and objects and keeps only those ground actions that were used in the tree of actions built from the goals to the initial state. The remainder of the initial state facts, objects (and facts and actions including them), and other actions are pruned from the graph to yield a reduced problem to be exposed to search.

If IPP determines the reduced problem to be unsolvable, which it usually does quickly, because of the effective termination test inherent to GRAPHPLAN, the search space is enlarged by applying RIFO again using a weaker pruning heuristic, restoring all the previously excluded actions that use objects that are newly judged to be potentially relevant. If IPP still cannot solve the problem, the original search space is fully restored to retain completeness of the planner, in the worst case leading to some overhead if both RIFO heuristics turn a solvable planning problem into an unsolvable one or if a completely unsolvable problem is investigated.

Handling of Conditional Effects in IPP Probably the most important ADL extension handled by IPP is the use of conditional effects. Gazen and Knoblock (1997) observe that all the other commonly handled ADL extensions can be converted into STRIPS in reasonable time but that conditional effects lead to a potentially crippling combinatorial cost if an attempt is made to deal with them in this way.

IPP represents conditional effects directly in planning graphs and also propagates mutual exclusion relationships between facts occurring in conditional effects and their effect conditions. However, it does not extend the notion of mutual exclusion between actions, as defined in GRAPHPLAN. This fact contrasts with the approach taken in SGP, discussed later, where a propagation of mutual exclusion relations over actions is proposed.

Actions with conditional effects are still treated as atomic units in IPP, but SGP creates components similar to GRAPHPLAN'S STRIPS actions that represent each conditional effect in the graph. Thus, there is the possibility of marking component nodes as exclusive as well as, it is claimed, a performance benefit. A detailed discussion of the relative merits of the two approaches is beyond the scope of this article but remains an interesting area of debate in the GRAPHPLAN community.

Interestingly, the movie domain, which was designed to explore the differences in the approaches to conditional effects, failed to achieve this objective because of an additional feature of IPP: the exploitation of *inertia*, or unchanging facts in the domain, to carry out constant folding among the grounded actions and reduce them to simpler forms. This process represents an important element of the collection of techniques being developed to control the size of the plan graph for GRAPHPLAN-based approaches to planning.

SGP

SENSORY GRAPHPLAN (SGP) is a sound, complete planner also built following GRAPHPLAN. SGP includes support for conditional effects, universal and existential quantification, uncertainty, and sensing actions. SGP only entered the ADL track of the competition to demonstrate these extensions to the underlying GRAPHPLAN machinery. SGP was the only planner written in Lisp to enter the competition and, despite the performance limitations from the language, competed well.

SGP has a number of features that set it apart from the other GRAPHPLAN-based planners. These unique features, described in the following subsections, are factored expansion for handling conditional effects and the ability to handle uncertainty and sensing actions. SGP is written in well-documented Lisp code and can be downloaded from the SGP home page (SGP 1998). SGP continues to be supported by the University of Washington, with new bug fixes and new releases being produced as appropriate.

Factored Expansion of Conditional Effects Many of the expressive features of ADL are easy to implement in GRAPHPLAN, but handling conditional effects is surprisingly tricky. Conditional effects allow the description of a single action with context-dependent effects. The basic idea is simple: A special when clause is introduced into the syntax of action effects. When takes two arguments: (1) an antecedent and (2) a consequent. Execution of the action will have the consequent's effect just in case that the antecedent is true immediately before execution (much like the preconditions of the action determine if execution itself is legal), which is exactly why handling actions with conditional effects is tricky: The outcomes of the actions depend on the state of the world when the action is executed.

Factored expansion was first described in Anderson and Weld (1998), where it was compared to full expansion (Gazen and Knoblock 1997) and the conditional-effects handling method of IPP. The idea behind factored expansion is twofold: First, all actions that contain conditional effects are factored into components, with a single effect (conditional or unconditional) in each component. Each component can then be treated, more or less, just like a STRIPS action. Second, when the components are being examined for graph expansion and backtracking search, the interaction between components from the same action is considered. In particular, there is the new concept of *component induction:* Component C_n induces C_k at level *i* if it is impossible to execute C_n without executing C_k .

Component induction leads to changes in both the GRAPHPLAN mutual-exclusion rules and in the backtracking search for a solution. The change to the mutual-exclusion rules is the addition of a new rule that can identify more mutual-exclusion relations: Two components C_n and C_m at level *i* are mutually exclusive if there exists a third component C_k that is mutually exclusive with C_m , and C_k is induced by C_n at level *i*. The change to the backtracking search requires that when a component is selected to establish a goal, all other components from the same action instance must also be considered.

An example of a domain where the factored expansion approach shows its strength is the movie domain. The goal of the movie domain is to collect snacks and then watch a movie. Before the movie can be "watched," the movie must be rewound, and the video cassette recorder's (VCR) counter must be set to zero. Consider what the planning graph looks like after one level has been built: The VCR's counter can be reset, the tape can be rewound, and snacks can be fetched. The key thing, however, is that the actions to rewind the movie and to reset the VCR's counter are mutually exclusive only if their conditional effects are considered. The IPP method of handling conditional effects does not find this mutual-exclusion relationship, and thus, IPP must perform an exhaustive search of the planning graph to determine that no plan yet exists.

SGP, however, can identify the induced mutual-exclusion relationship between the two actions. Thus, SGP immediately knows that no plan yet exists and can thus proceed to the next planning graph level.

Uncertainty and Sensing Actions The second novel feature in sGP is its ability to deal with uncertainty and sensing actions (this work is detailed in Smith and Weld [1998a] and Weld, Anderson, and Smith [1998]). Although this part of SGP was not used at the AIPS-98 planning contest, it is discussed here in some detail because it was the main motivation for work on SGP and is the characteristic that distinguishes SGP from the other planners.

Classical planners assume complete knowl-

edge of the world at plan time and assume that the outcome of actions is certain. Although this assumption simplifies the planning process, it is an unrealistic one. SGP relaxes the assumption that the agent knows the state of the entire world a priori.

The first step in relaxing this assumption is to allow the initial state to include propositions whose truth values are uncertain (either true or false). Internally, SGP represents this sort of uncertainty by keeping track of each possible world. For example, if both propositions P and *Q* are uncertain, then there are four possible worlds: (1) P and Q, (2) P and not Q, (3) not Pand Q, and (4) not P and not Q. For each possible world, SGP builds and maintains a separate planning graph. This task is complicated by the fact that because of conditional effects, the outcome of an action might be different depending on the world it is executed in. It should be noted here that SGP makes the assumption that an uncertain proposition might be true or false but does not make any assumptions about probabilities (SGP simply records that a proposition is uncertain and does not associate a numeric weight with each possibility).

The second step in relaxing the complete knowledge assumption is to limit the agent's observational powers to explicit sensing actions. *Sensing actions* are actions in the domain whose effects include a special sense statement. *Sense statements* are used to let the agent query the world for the truth value of a proposition. Based on the sensed truth value, the agent can then refine the subset of possible worlds it is in (for example, if a proposition *P* is sensed to be true, then the agent knows that it is in one of the possible worlds in which *P* is true).

Given the set of planning graphs (one for each possible world) and the set of actions, including sensing actions, SGP builds a plan that will achieve the goals in all possible worlds. To guarantee success in each possible world, the plan can include actions that are to be executed only in some of the possible worlds. Thus, the plan might have branch points contingent on the consequences of sensory actions. The plan is a directed acyclic graph, with the possibility of branches rejoining with each other. The contingencies in the plan are based on which possible world the agent is in at plan-execution time. To resolve which possible world the agent is in, the results of the sensing actions are used. An important feature of SGP is that the planner determines exactly which sensing actions need to be taken to resolve the uncertainty. There are two important outcomes here: First, the planner does not assume that the agent has full observational power. Instead, SGP allows only

the terms sensed by the sensing actions to be used in resolving uncertainty. Second, the agent does not have to resolve exactly which possible world it is in but, rather, which of a set of possible worlds it is in. This distinction is important because with SGP, the agent need not necessarily resolve all the uncertainty but simply "enough."

STAN

STAN (state-analyzing planner) (1998) is a third system based on GRAPHPLAN that extends its planning technology in several ways: First, a more sophisticated data structure is used to store the graph and aid in its construction than was used in earlier versions of GRAPHPLAN. Second, more analysis is carried out on the graph and the problem it encodes to reduce search branches, including goal-ordering analysis, symmetry analysis, and resource analysis. Third, a wave front is exploited at the fix point of the plan graph to avoid unnecessary additional work. Fourth, state-analysis techniques, implemented as a module of STAN that can be exploited as a planner-independent system, TIM, are used to acquire information about a domain and problem encoding that is exploited in instantiation and filtering of the plan graph.

The implementation of the plan graph in STAN is based on a careful reuse of much of the central graph structure, to avoid repeatedly copying layer-independent information, together with bit-level operations to support graph construction and careful filtering of the layer-dependent structures to reduce the retesting carried out at subsequent layers. This implementation, the most recent version of which is described in full in Long and Fox (1999), is efficient and relatively compact, although the competition version of STAN (STAN 1.0) still allowed unnecessary wastage of space. A similar approach, although introduced with a slightly different purpose (to extend GRAPH-PLAN to handle actions with duration), is discussed in Smith and Weld (1999, 1998b).

The plan graph is carefully analyzed during construction to produce several auxiliary relationships between goals and between actions at each layer. In particular, STAN identifies ordering relations between pairs of goals reflecting the order in which they must be satisfied so that both are true at a given level. Thus, a considerable reduction in the search space is allowed by automatically selecting noops for goals that must be satisfied earlier than the current layer during search. Furthermore, certain chains of ordered goals can prevent an entire goal set from being achieved at a given layer without any search at all.

STAN also identifies some goal sets as unachievable when they exceed certain resource limitations imposed by the domain. Resources include numbers of objects in certain configurations, the rate at which certain objects can be put into key configurations during the plan execution, and other factors as well as the physical resources available to the planning agent (such as grippers, fuel, and containers). Limits on the availability of abstract resources, such as the rate at which objects can be configured, arise from the limits imposed by the physical resources of the agent and are expressed, for example, in terms of constraints on the number of plan-graph levels that must have been built before a certain goal configuration can be achieved. The resource analysis performed by STAN can dramatically reduce search, by identifying the minimum number of graph layers that must be built, in some domains including the traveling salesman problem discussed later.

Finally, STAN exploits structural features of the problem domain, such as symmetry, to reduce search. The competition version of STAN performed only a preliminary symmetry analysis in which symmetric objects (those that are indistinguishable and, hence, do not form interestingly different action instantiations) were identified to reduce the number of action instantiations produced. Although STAN was able to detect certain forms of symmetry, the competition version was not able to exploit it fully. A more advanced treatment has been developed since the competition (Fox and Long 1999).

One of the most important features of STAN that distinguishes it from other GRAPHPLANbased planners is its use of a highly efficient implicit representation of the graph beyond the fix point. STAN avoids both construction and search of the plan graph beyond the fix point, where the graph is static. Instead, search is built on a collection of candidate goal sets that are generated as failure sets at the fix point and that are pushed one layer forward to be retried. This process in which failed goal sets are promoted forward forms a kind of rolling collection of goal sets at the fix point, which is called a wave front (Long and Fox 1999). The efficiency gains this mechanism offers can be dramatic in certain problems and can also significantly reduce the memory demands during the process of constructing a solution.

The exploitation of the results of state analyses of various kinds is an important feature of STAN. State analyses can be done in a preprocessing stage, using techniques that are planner independent, and the results fed into the planning process and used to reduce, and even eliminate, some of the more resource-intensive aspects of planning. Symmetry analysis, in which symmetric objects and actions are identified to prune redundant search, is one such form of state analysis. Another technique, currently being integrated with STAN 3, is the types and state-invariant inference analysis performed by the TIM module (Fox and Long 1998). The competition version of STAN was not fully integrated with TIM and therefore could not make use of the inferred state invariants. It could exploit the type structure inferred by TIM, but the competition domains had types supplied, hence STAN did not have as large an advantage over the other competitors as can be achieved with some domain encodings. More recently, TIM has been extended to allow the automatic identification of certain generic domain behaviors such as objects that traverse a map of locations and objects that can be carried around such a network, together with the objects responsible for carrying them. This analysis can be exploited in a variety of ways, but an initial use is in filtering useless action instances from the initial action-instantiation phase common to GRAPHPLAN-based planners.

Development of STAN During the development of the competition version of STAN, a pattern was established of exploring the behavior of the planner on a family of problems to understand what made them hard followed by constructing techniques to tackle the source of the difficulty in a domain-independent way. An important lesson learned from this process, and from the competition itself, is that problems are hard for a wide variety of reasons and that these different sources of difficulty can lead to the development of techniques that are powerful in certain contexts but are simply useless overhead in others. Of particular interest were problems that appeared hard for STAN yet were easy for other planners or are intuitively easy.

One of the reasons problems can be hard, affecting GRAPHPLAN-style planners in particular, is that domains can contain huge collections of instantiated actions that have no useful role in the plan. Thus, you have both a large cost in the construction phase and, often, a large cost in the search phase when many branches must be explored. These branches often express the same fundamental planning decisions but, because of the search in grounded actions, differ in details that are insignificant from the planning point of view.

STAN attempts to deal with this problem in several ways: Type inference leads to a potential reduction in the numbers of instantiated actions; the use of static conditions in instantiation; and some preliminary work on filtering to remove some objects in some domains. Much work remains to be done in this area: RIFO offers IPP a huge benefit in many problems by filtering out many irrelevant objects, actions, and facts but at the price of possible incompleteness. As can be seen in Review of Results, all the planners were confounded when confronted with problem instances containing large numbers of ground-operator instances. The ability to reduce these numbers by intelligent filtering would appear to be a critical element to successful planning.

In contrast, some problems are hard not because of the cost of construction of the graph but because of the multiplicity of search paths in the problem. This multiplicity is particularly problematic in cases in which the problem appears to be solvable long before it actually is (the graph contains the nonpairwise, mutually exclusive goals long before the solution layer). An example of such a problem is the completegraph traveling salesman problem (TSP) in which the graph is completely connected so that the traveler is unconstrained in the order of visits he makes. In this case, the problem is that all the destinations could be visited within a single step, and any pair could be visited after two steps. Thus, the problem appears solvable after two steps but is not actually solvable until *n* steps have passed, where *n* is the number of sites to be visited. However, as *n* grows, the number of possible paths the traveler might have taken explodes exponentially, so that the search problem quickly becomes intractable. STAN uses resource analysis to determine that only one destination can be visited at each step, so it does not search for a plan until *n* layers are constructed and can also ensure that every layer is used for a visit to a hitherto unvisited site.

Unfortunately, there are other problems of similar character that are not yet adequately tackled by STAN. The gripper problem used in the competition is an example: STAN correctly determines that no more than two balls can be deposited at each time step. However, it does not allow for the fact that the two balls must also be picked up and transported, requiring an additional two steps (and a third to return to the source for another load), so the resource analysis does not help as much as one might hope.

GRAPHPLAN-style planners also suffer during search from a problem of premature commitment. This problem occurs because the use of ground-action instances forces selection of objects to play particular roles in a plan often before constraints that would govern their choice become apparent. Constraint-solving planners can benefit with these problems by making choices in the highly constrained parts of the plan and propagating them into the less constrained parts of the plan. Forward- and backward-chaining planners do not direct their planning strategies by exploiting the most highly constrained parts of a plan structure, which can lead to costly mistaken choices that must be retracted.

Some problems are hard because they have an inherent combinatorial cost, and others are, in principle, easy yet prove hard with current planning technology or at least with some current planning strategies. The gripper domain is a good example of the latter-a domain in which problems are trivial for human problem solvers, yet an optimal plan for this domain eluded all the planners in the competition for instances larger than a dozen balls. More recent work on STAN has explored the exploitation of symmetry in problems such as the gripper that can reduce the difficulty dramatically (Fox and Long 1999). Nevertheless, clearly much work remains to be done in recognizing and exploiting features of problems that humans appear to identify with ease. Furthermore, the representation of solutions to problems such as these remains an issue: No human would represent the solution to problems in the gripper domain as an explicit sequence of steps but would as a method for generating these steps during execution ("Transport the first two balls from room *a* to room *b*, then return and repeat until all balls are transported." The fact that the first two balls are not identified by name is an indication of the role of symmetry for human problem solvers).

BLACKBOX

It has often been observed that the classical AI planning problem (that is, planning with complete and certain information) is a form of logical deduction. Because early attempts to use general theorem provers to solve planning problems proved impractical, research became focused on specialized planning algorithms. However, the belief that planning required such specialized reasoning algorithms was challenged by the work of Kautz and Selman (1996, 1992) on planning as propositional satisfiability testing. SATPLAN showed that a general propositional theorem prover could be competitive with some of the best specialized planning systems. The success of SATPLAN can be attributed to two factors: (1) the use of a logical representation that has good computational properties and (2) the use of powerful new general reasoning algorithms such as WALKSAT (Selman, Kautz, and Cohen 1994). Both the fact that SATPLAN uses propositional logic instead of first-order logic and that we suggested the particular conventions for representing time and actions are significant. Differently declarative representations that are semantically equivalent can have distinct computational profiles. The use of general reasoning algorithms offers an important benefit because many researchers in different areas of computer science devise new algorithms and implementations for SAT testing each year and freely share ideas and source code. SAT is the general problem of determining whether there is a way to set a collection of identified variables to each be either true or false so that a given logical expression involving these variables is made true. As a result of this shared work and the size of the interested community, at any point in time, the best general SAT engines tend to be faster (in terms of raw inferences a second) than the best specialized planning engines.

Interestingly, the GRAPHPLAN approach to planning shares a number of features with the SATPLAN strategy. Comparisons with SATPLAN show that neither algorithm is strictly superior. For example, SATPLAN is faster with a complex logistics domain; they are comparable on the blocks world; and with several other domains, GRAPHPLAN is faster.

GRAPHPLAN bears an important similarity to SATPLAN: Both systems work in two phases, first creating a propositional structure (in GRAPHPLAN, a plan graph, in SATPLAN, a formula in conjunctive normal form [CNF]) and, second, searching this structure. The propositional structure corresponds to a fixed plan length, and the search reveals whether a plan of this length exists. Furthermore, in Kautz and Selman (1996), it is shown that the plan graph has a direct translation to a CNF formula and that the form of the resulting formula is close to the original conventions for SATPLAN. It is hypothesized that the differences in performance between the two systems can be explained by the fact that GRAPH-PLAN uses a better algorithm for instantiating the propositional structure, and SATPLAN uses more powerful search algorithms.

SATPLAN fully instantiates a problem instance before passing it to a simplifier and a solver. By contrast, GRAPHPLAN interleaves plan-graph instantiation and simplification. Furthermore, GRAPHPLAN uses a powerful planning-specific simplification algorithm: the computation of the mutual-exclusion relations between pairs of actions and facts.

These observations have led to the creation of a new system that combines the best features of GRAPHPLAN and SATPLAN. This system, called BLACKBOX, works in three phases. First, a planning problem (specified in a standard STRIPS notation) is converted to a plan graph. Second, the plan graph is converted into a CNF formula. Third, the formula is solved by any of a variety of fast SAT engines.

The formula generated from the plan graph can be considerably smaller than one generated by translating STRIPS operators to axioms in the most direct way, as was done by the earlier MEDIC system of Ernst, Millstein, and Weld (1997). Furthermore, the mutual-exclusion relations computed in the plan graph can be translated directly into negative binary clauses, which can make the formula easier to solve for many kinds of SAT engines.

The competition version of BLACKBOX included the local-search SAT solver WALKSAT and the systematic SAT solver SATZ (Li and Anbulagan 1997) as well as the original GRAPHPLAN engine (that searches the plan graph instead of the CNF form). To have robust coverage over a variety of domains, the system can use a schedule of different solvers. For example, it can run GRAPHPLAN for 30 seconds; then WALKSAT for 2 minutes; and, if still no solution is found, SATZ for 5 minutes.

The BLACKBOX system introduces new SAT technology as well, namely, the use of randomized complete search methods. As shown in Gomes, Selman, and Kautz (1998), systematic solvers in combinatorial domains often exhibit a "heavy tail" behavior, whereby they get "stuck" on particular instances. Adding a small amount of randomization to the search heuristic and rapidly restarting the algorithm after a fixed number of backtracks can dramatically decrease the average solution time, often from hours to seconds.

This randomization-restart technique was applied to the version of SATZ used by BLACKBOX. The variable-choice heuristic for SATZ chooses to split on a variable that maximizes a particular function of the number of unit propagations that would be performed if that variable were chosen (see Li and Anbulagan [1997] for details). The BLACKBOX version, SATZ-RAND, randomly selects among the set of variables whose scores are within 40 percent of the best score. The solver schedule used in the competition was to run the GRAPHPLAN engine for 2 seconds, then to convert the problem to CNF and run SATZ-RAND for 10 restarts with a cutoff of 100 backtracks. If one of the solvers showed that no solution existed, the plan graph was extended by one layer and the schedule repeated. Note that no one cutoff value is ideal for all domains. One way to address the problem is to specify a sequence of increasing cutoff values in the solver schedule.

The newest version of BLACKBOX includes an additional solver, RELSAT (Bayardo and Schrag 1997), based on dependency-directed backtrack-

ing as well as a technique for reducing the size of the CNF encodings by suppressing the generation of clauses that are logically redundant.

BLACKBOX was designed to improve the *planextraction phase* of GRAPHPLAN (that is, the search within the plan graph for a solution). In the majority of the problems solved in the competition, plan extraction was comparatively easy once the plan graph was constructed. Systems that reduce the plan-graph structure and thereby reduce the cost of constructing it enjoyed a relative advantage over BLACKBOX in this context. Problems that have a relatively much harder plan-extraction phase are included in the BLACKBOX (1998) distribution.

HSP: Heuristic Search Planner

HSP is a planner based on the well-established idea of heuristic search. Heuristic search algorithms perform forward search from an initial state to a goal state using a heuristic function that provides an estimate of the distance to the goal. The 8-puzzle is the standard example of heuristic search and is treated in most AI textbooks (Pearl 1983; Nilsson 1980). The main difference between the 8-puzzle and the approach to planning adopted in HSP is in the heuristic function. Although in domain-specific tasks such as the 8-puzzle, the heuristic function is given (for example, as the sum of the Manhattan distances), in domain-independent planning, it has to be derived from the high-level representation of the actions and goals.

A common way to derive a heuristic function h(s) for a problem P is by relaxing P into a simpler problem P' whose optimal solution can be computed efficiently. Then, the optimal cost for solving P' can be used as a heuristic for solving P (Pearl 1983). For example, if P is the 8-puzzle, P' can be obtained from P by allowing the tiles to move into any neighboring position. The optimal cost function of the relaxed problem is precisely the Manhattan-distance heuristic.

In STRIPS planning, the heuristic values for a planning problem *P* can be obtained by considering the *relaxed* planning problem *P'* in which all delete lists are ignored. In other words, *P'* is like *P* except that delete lists are assumed empty. As a result, actions can add new atoms but not remove existing ones, and a sequence of actions solves *P'* when all goal atoms have been generated. As with all the other competition planners, action schemas are first grounded in an initial instantiation phase, so variables do not occur in actions.

It is not difficult to show that for any initial state *s*, the optimal cost h'(s) to reach a goal in *P*' is a lower bound on the optimal cost $h^*(s)$ to reach a goal in the original problem *P*. The

heuristic function h(s) could therefore be set to h'(s) and obtain an informative and admissible (nonoverestimating) heuristic. The problem, however, is that computing h'(s) is still NP hard (first observed by Bernhard Nebel). Therefore, an approximation is used: The heuristic values h(s) are set to an estimate of the optimal values h'(s) of the relaxed problem. These estimates are computed as follows:

Starting with $s_0 = s$ and i = 0, s_i is expanded into a (possibly) larger set of atoms s_{i+1} by combining the atoms in s_i with the atoms that can be generated by the actions whose preconditions hold in s_i . Every time an action that asserts an atom p is applied, a measure $g_s(p)$ is updated that is intended to estimate the difficulty (number of steps) involved in achieving pfrom s. For atoms p in s, this measure is initialized to 0, but for all other atoms, $g_s(p)$ is initialized to infinity. Then, when an action with preconditions $C = r_1, r_2, ..., r_n$ that asserts p is applied, $g_s(p)$ is updated as

$$g_s(p) := \min[g_s(p), 1 + \sum_{i=1,n} g_s(r_i)]$$

The expansions and updates continue until these measures do not change. When all preconditions involve a single atom, it is a Bellman-Ford procedure for computing costs in a graph from a given set of sources. The nodes are the atoms, the sources are the atoms *p* such that $g_s(p) = 0$, and edges *p* to *q* with cost 1 exist when an action with precondition *p* asserts *q*.

The heuristic function h(s) used by HSP is defined then as

$$h(s) \stackrel{def}{=} \sum_{p \in G} g_s(p)$$

where *G* stands for the set of goal atoms. This definition assumes, like decompositional planners, that subgoals are independent. The added value of the heuristic approach is that subgoals are weighted by a *difficulty measure* that makes it possible to regard certain decompositions as better than others. A result of this assumption is that the heuristic function h(s) is not admissible. However, it is often quite informative and can be computed reasonably fast.

The Search Algorithm The heuristic function defined previously allows us to deal with any STRIPS planning problem as a problem of heuristic search. Thus, planning could be carried out using algorithms such as A*. A*, however, might take exponential memory and approaches the goal too cautiously. In HSP, where the heuristic is recomputed from scratch at every node, it is necessary to use algorithms that can get to the goal with as few evaluations as possible. For this reason, HSP uses a form of hill-climbing search. Surprisingly, hill climbing works well in many problems and often produces good plans fast. Sometimes, however, it gets stuck in local minima. To tackle this problem, search proceeds until a fixed number of impasses have been encountered, restarting the search if necessary, to some specified maximum number of times. The algorithm used in the competition is a variation of this idea that also uses memory to keep track of the states that are visited. Current effort is directed toward identifying ways to speed up the evaluation of the heuristic so that more systematic search algorithms could be used.

HSP is implemented in C. In contrast to all the other competition systems, a preprocessor is used to convert a STRIPS problem in PDDL into a c program that is then compiled, linked, assembled, and executed. This process usually means a time overhead on the order of a second or two in small planning problems but pays off in larger ones.

Related Work HSP is based on the planner reported in Bonet, Loerincs, and Geffner (1997). This planner, called ASP, uses the same heuristic function but a different search algorithm based on Korf's (1990) LRTA, which was designed for real-time planning. An independent proposal that also formulates planning as heuristic search was developed by McDermott (1996).

The bottleneck in HSP is the computation of the heuristic values that are obtained afresh in every new state. Work carried out since the competition has led to a solution to this problem, reported in Bonet and Geffner (1999). A related proposal is owed to Refanidis and Vlahavas (1999). The work reported in Bonet and Geffner (1999) also suggests that there is a close relation between heuristic search planning and GRAPHPLAN planning that might be worth further investigation.

Further details on HSP and code can be found at the HSP (1998) web site.

Review of Results

One of the tasks the competition committee faced was to determine a strategy for evaluating planner performance. Before the competition, a formula was proposed that combined weighted values for plan length and planning time, adjusted to reflect the relative performance of different planners on the same problem (to give due credit to planners that quickly solved problems that defeated many of the others). In the event, this formula was judged to give counterintuitive results, and it was essentially abandoned, leaving a void in the final evaluation of performance; in the sTRIPS track, it

It is worth emphasizing that three of the planners running in the STRIPS track (BLACKBOX, IPP, and STAN) all produce parallel optimal plans (although IPP does not guarantee to do so when using its RIFO *machinery*), which. when linearized. will not always lead to optimal sequential plans.

remains a difficult task to assess the relative performances of the planners. A summary of the results was presented at the event, but it was crude in that it failed to differentiate between good performance on simple problems and good performance on hard problems. This masking is amplified by the fact that each of the planners faced minor problems because of program bugs that made what would have otherwise been simple problems appear hard for these planners. In addition, in the mystery domain, several problems were unsolvable (and proven such by some of the planners), but these problems were ignored in the results summary. Overall, the results of the individual planners all showed strengths and weaknesses, and it is not surprising that a simple direct comparison proved unsatisfactory in the competition. This challenge remains unsolved for future competition events, and the community must be wary of setting up targets (in whatever form) that oversimplify the objectives of planners. The problem that evaluation represented suggests that these objectives remain a complex and clouded issue.

It is worth emphasizing that three of the planners running in the STRIPS track (BLACKBOX, IPP, and STAN) all produce parallel optimal plans (although IPP does not guarantee to do so when using its RIFO machinery), which, when linearized, will not always lead to optimal sequential plans. This can explain the discrepancies in plan lengths discovered by these planners. In general, the length of the linearized plan is difficult to control when using a mechanism that produces optimal parallel plans. HSP produces linearized plans but does not support claims for optimality. An interesting example to consider is the seventh logistics problem in the first round of the competition, where HSP produced the only plan found by any of the planners. This plan was 112 (sequential) steps long. STAN has subsequently demonstrated that there is a (sequential) 37-step plan!

The First Round: STRIPS Track

The competition involved the use of five domains in the first round and three in the second. The first round used the gripper, the movie, the logistics, the mystery, and the mystery prime domains. The last two are variations on transportation domains with limited fuel. In the last domain, the fuel can be piped between nodes in the transportation network, but in the mystery domain, the fuel is held at its starting depots. In the first round, 30 problems were presented of each type, except for gripper, in which just 20 problems were presented.

The characteristics of the last three domains

are similar in that they are all transportation problems that involve moving objects around a network of locations as efficiently as possible. Interestingly, these domains all had a similar performance limit for all the planners: No planner could solve a logistics problem with more than 10,000 ground-action instances, and all the problems with fewer than 10,000 ground-action instances were solved by at least 1 of the planners. Where reported, the numbers of ground-action instances have been computed using STAN, with all filtering mechanisms turned off. STAN uses the TIM subsystem to generate a type structure for each domain, which can result in fewer ground-action instances being generated than would be the case with raw instantiation. The mystery prime domain proved more tractable, with problems including as many as 24,290 ground-action instances being solved by some planners (although the problem instance with 24,290 actions involved producing a plan with only 4 steps). However, problems with over 10,000 ground-action instances still proved hard in general, with several planners failing on these large problems and inconsistent performance being demonstrated between the threshold of 10,000 ground-action instances and the largest solvable problems. More than half the mystery problems presented in the competition were under 10,000 ground-action instances in size. Of the 13 problems that exceeded this size, 5 were unsolvable, and 3 were solved in the competition by at least 1 of the planners. The other five problems were all solvable, at least with STAN, although buffer sizes were set too small in the competition configuration for it to solve them. STAN uses an object-filtering mechanism that worked successfully in both the mystery and mystery prime domains to cut the numbers of ground-action instances so that in the mystery domain, none of the problems presented actually produced more than 8,000 ground-action instances.

Interestingly, of the 30 mystery-domain problems presented in the competition, 11 were proved unsolvable by at least 1 planner rather than simply proposed unsolvable because of a lack of resources. This distinction was not used in the competition (presumably because of the difficulty in distinguishing an accurate claim that a problem is unsolvable from a lucky guess when resources run out), but the three GRAPHPLAN-based planners used in the competition are capable of identifying problems of this kind (at least in principle); however, BLACKBOX can identify some unsolvable problems (those in which some of the goals are unreachable or are pairwise unreachable). STAN was fastest in demonstrating 10 of the 11 problems to be unsolvable (on average, it was 15 times faster than its nearest rival at showing these problems unsolvable), and IPP was fastest on the remaining problem.

The movie domain presented no difficulty to any of the planners, and performance times were so small that they cannot really be usefully compared. This domain was included to consider its effect in the ADL track, as was discussed in SGP. The gripper domain is peculiar in that it is a domain that is intuitively easy to solve-the problems present no difficulty for a human planner—yet only HSP was able to solve instances involving more than 12 balls. Performance of all the other planners deteriorated exponentially with the increasing problem size. IPP used RIFO in this problem, allowing it to solve more problems than would otherwise have been possible, by excluding one gripper from consideration. HSP produced similarly suboptimal solutions by carrying only one ball on each trip. This problem is so hard because there are so many ways in which the actions can almost solve the problem with a shorter sequence than is actually required to completely solve it, and these garden-path sequences increase exponentially with two grippers despite the fact that even the hardest problem instance presented in this domain contains only 340 ground-action instances! One of the reasons this problem appears to be so simple for a human planner is that a human can see the essential symmetry to the problem and can exploit it to simplify the problem to the extent that it becomes trivial. This problem is one that highlights a critical weakness of the current fast-planning technology.

In round one, of the 140 problems presented, 98 problems were either solved or proved unsolvable by at least 1 of the planners. At least eight of the remaining problems have been solved by one or more of the planners since the competition. The problem domain that proved difficult for all the planners was the logistics domain, accounting for 25 of the unsolved problems.

The Second Round: STRIPS Track

In the second round, a new domain was introduced: the grid domain. IPP managed to solve three of the five problems presented, using strong RIFO pruning and producing suboptimal plans. The other planners managed only one problem in this domain. The problem sizes ranged from 2,609 ground-action instances for the simplest through 16,239 (unsolved by any planner). IPP solved an instance with 11,150 ground-action instances.

The other domains used were logistics and mystery prime. All but one of the problems presented in these domains were solved by at least one planner. All the logistics problems were under 5,000 ground-action instances (although the 2 hardest of these involved few planes, and in 1, many trucks made for a big search space). All but one of the mystery prime problems were under 6,000 ground-action instances, and the exception contained 19,730 ground-action instances. This instance defeated all the planners in the competition, although at least one of the planners has since generated a 33-step plan that solves it. With the exception of a single instance (traced to a trivial program bug), all the other problems were solved by all the planners. Thus, of the 15 problems presented in round 2, 12 were solved in the competition, and at least 1 further problem has been solved since. The grid domain proved the least tractable, with only IPP making much headway and even then only producing suboptimal plans. A critical problem in this domain appears to be that the plans are comparatively long with no parallel steps. Thus, graph construction is an expensive process if there is no pruning. Failure to complete the graph-construction phase to achieve a single search was the reason for failing to solve these problems in all the GRAPHPLAN-related planners.

In light of the observations already made about the sizes of the problems measured in terms of the numbers of ground-action instances, it is interesting to consider the behavior of IPP using the RIFO subsystem, which filters some objects and action instances from domains prior to planning. RIFO was not used by IPP in the first round of the competition, except in the gripper domain in which it was turned on by hand. In this domain, RIFO identifies one gripper hand as irrelevant, so that only one ball is carried at a time, and plans become much longer.

In the second round, IPP was run using the RIFO metastrategy, discussed earlier, which significantly reduced the search space for the planner. As table 1 shows, only eight problems can be solved without RIFO, but with the metastrategy, three more solutions are found. The metastrategy combines only a small subset of possible RIFO pruning heuristics. Discovering which combinations of heuristics work for which domains and problems is still a matter of experimentation. In the competition, no such experimentation was possible; therefore, the selection of the heuristics was done long before the competition, following the intuition that one should try the strongest-possible heuristic first because it leads to the smallest search

The problem domain that proved difficult for all the planners was the logistics domain, accounting for 25 of the unsolved problems.

001	571/25 (13	_	
og2	502/21 (20)	_	_
og3	958/26 (27)	_	_
og4	3561/42 (—)	189/30 (—)	_
og5	4985/53 (—)	119/26 (31)	_
nprime1	7809/36 (5)	7/9 (⊥)	11/9 (⊥)
nprime2	3281/32 (8)	_	_
nprime3	97259/68 (—)	_	_
nprime4	8485/42 (5)	7/10 (4)	_
nprime5	6773/22 (6)	_	_
grid1	2610/38 (14)	_	80/11 (20)
grid2	4501/50 (—)	69/19 ()	260/19 (31)
grid3	7256/64 (—)	315/21 (⊥)	557/21 (—)
grid4	11151/80 (—)	135/24 (47)	_
grid5	16240/98 ()	1481/49(_

Table 1. This Table Shows the Number of Ground Actions and Objects in the Original Problem Descriptions and, in Brackets, the Length of the Plan (If IPP Could Find One Given a 10-Minute Central Processing Unit [CPU] Time Limit and a 120-Megabyte Memory Limit on a SPARC 1.170).

It also shows the number of selected ground actions and objects after the stronger and weaker RIFO pruning processes. \perp shows the planning problem became unsolvable, — means RIFO was inactive, and (—) means no plan was found because the planner either exceeded the CPU time or memory limit.

space and then relax it by allowing more actions if no solution is found. The threshold parameters that decide whether RIFO is activated at all were set to 3500 actions and 35 objects after a few trials on some of the competition problems.

The ADL Track

In the ADL track, the same collection of domains and problems was used as in the first round of the STRIPS track. Of course, the domain encodings were reconstructed to exploit the ADL features. Only SGP and IPP competed in this track.

IPP solved all problems from the movie domain, the first 5 problems from the gripper domain containing 20 test problems, 13 of 30 problems in the mystery and mystery prime domains, and 3 of 30 problems in the logistics domain. In total, it solved 69 problems in approximately 20 minutes, including all the 38 problems SGP solved. RIFO would not have improved the performance of IPP in this round because on most of the problems, it makes the planner incomplete; so, the planner would have had to find the solution in the original search space after all reduction attempts had failed. In the movie domain, however, the metastrategy using the weaker heuristic succeeds in determining that only a handful (between five and nine) of the initial facts in each problem are relevant.

Commentary

Throughout the competition, no planner successfully solved any problem instance that involved more than 60,000 ground actions. In fact, without filtering techniques to reduce the number of ground actions, no planner solved problems with more than 25,000 ground actions, and reliable performance was restricted to problems with fewer than 10,000, or so, ground-action instances. In domains with harder search-space growth problems, even fewer action instances could be handled. Although the number of ground-action instances is not an infallible guide to the difficulty of problems, it is clearly an important indicator and strongly suggests that at least for planners that work with ground actions during plan construction, there is much work to be done on the filtering process that could remove unnecessary actions from the problem space. It is interesting to observe that problem 15 of the first round in the logistics domain lies beyond the scope of the competition planners even with a manual filtering of the domain objects, removing irrelevant packages and trucks, leaving as few as 3,006 ground actions. The difficulty of this problem is not easy to understand because there appears to be no pressure on the aircraft resources (with six aircraft available in just three cities), although the fact that the cities each have six locations could well be significant.

Although the number of ground actions represents an important element in determining planner performance, the number of domain states is also a factor. Indeed, for planning systems that do not instantiate operators before planning, the number of states might be a more important feature of the problems. It is not easy to compute the numbers of states for some problems (particularly mystery and mystery prime domains) because reachable states are nontrivial to determine. However, for the logistics and gripper domains, it is straightforward. In the logistics problems that were solved in round 1, the state spaces contained between 10¹⁰ states (problem 5) and 8 x 10²⁵ states (problem 11). These are clearly large state spaces. By contrast, gripper problems define state spaces containing a mere 256 states (problem 1) to 68,608 states (problem 4) to more than $4 \ge 10^{15}$ states in the largest (problem 20), solved only by HSP. The 376,832-state problem (problem 5) was beyond all the planners but HSP, and none solved it optimally. This analysis gives an indication of the dramatic contrast between the problems in which the planning technology is performing well and the problems where it demonstrates fundamental weaknesses.

As a final summary of the results from the strips track, figures 2, 3, 4, and 5 attempt to give a broad indication of the relative performances of the planners. Figure 2 shows the cumulative numbers of problems solved with increasing time (in milliseconds). It should be noted that HSP solved more problems than any other planner (91 problems solved), but the graph has been drawn with a 15-second cutoff to allow a clearer view of the important data: Twentythree of the problems solved by HSP took it between 15 seconds and 14 minutes. Similarly, IPP required more than 15 seconds on 8 additional problems, and STAN required more than 15 seconds on an additional 3 problems. The graph demonstrates that the planners had remarkably similar performances, solving the bulk of problems in less than 10 seconds. The first 30 problems, or so, are from the movie domain, where it can be seen that the compilation overhead paid by HSP gives it comparatively poorer performance. Figure 3 shows a similar plot, indicating cumulative numbers of problems solved in increasing numbers of plan steps. In this case, no cutoff has been applied, and HSP is alone in solving the last 17 problems. Once again, the graph emphasizes the similarities in performance, although BLACKBOX appears to generate slightly better-quality plans. The significant step at seven steps is the result of the movie domain, where all plans are seven steps long. Figure 4 shows plan lengths plotted against times taken to produce them, revealing that plan length has surprisingly limited impact on the time taken to solve a problem, with comparatively short plans often proving as challenging to produce as longer plans. The trail of points curving above the main cluster is the sequence of results for HSP in the gripper domain.

Although there is no strong correlation between plan length and planner performance, there is a more suggestive correlation (valued at 0.69 across all the planners) between problem size and planner performance, as can be seen in figure 5. Of course, the specific domain has a significant impact on the difficulty of a problem (as shown by the gripper problems particularly), but the size of a problem file measured purely by byte count is a good indicator of difficulty. For example, the logistics problems that were solved were under 4,500 bytes in size, with the hardest logistics problem that was solved the only one over 4,000 bytes (round 1, problem 7, solved only by HSP). Similarly, mystery prime problems under 5,000 bytes were all solved, but problems between 5,000 and 6,000 bytes proved hard, and those above 6,000 bytes were unsolvable for these planners. Because problem size measured in this way is a good indicator of the number of objects in a problem and, therefore, of the number of ground-action instances, these data add further evidence to support the view that the number of ground actions is a key factor in determining the performance of these planners.

Other Challenges

Although the size of domains, particularly measured by numbers of instantiated ground actions, represents a critical challenge to planning technology, the domains in the competition and others explored independently by the competitors have revealed other important problems that must be addressed.

For example, the gripper domain highlights the combinatorial costs of exploring a large (and largely redundant) search space. The search must be reduced when possible, and in the gripper domain in particular, there is a huge potential for reduction in search costs. HSP



Figure 2. Cumulative Numbers of Problems Solved against Time (Data Cut Off at 15 Seconds).

demonstrated that heuristic search in this space can offer dramatic benefits. HSP solved all the gripper problems, where other planners managed at most four or five. This domain alone accounts for three-quarters of HSP's significant lead over the other competitors in the number of problems solved. HSP's heuristic effectively ignored the possibility of transporting the balls in pairs and solved the problems by transporting one ball at a time between the rooms. IPP, which succeeded in solving five of these problems, used its RIFO machinery, which caused it to ignore one of the grippers, also leading to solutions in which only one ball was transported at a time. None of the planners could exploit the incredible degree of symmetry in the problem to cut the search space from its exponential size to reflect the trivial underlying problem.

The logistics domain has the interesting property that the hardest part of the problem instances usually lies in the middle of the plan. The problems of transporting packages to and from airports, which sandwich the problem of flying packages between cities, are relatively easy but often generate large collections of redundant search paths. A planner that can tackle the core problem, the flying of packages between cities, and propagate necessary constraints outward to the simpler ends of the plan will have the advantage. This phenomenon represents a single instance of a more general issue: Many planning problems are not uniformly hard. A planner that can identify the hardest parts of a planning problem and concentrate on solving those parts first, propagating constraints toward the easier, initially less constrained, parts of the problem, will perform far better than a planner that always tackles the problems from the same place.

The mystery domain is a fascinating variation on the transportation theme, introducing resource limits on carrier capacity and fuel as well as an underlying route-planning problem. This domain (and the mystery prime variation) has the potential to produce problems that are



Figure 3. Cumulative Numbers of Problems Solved against Plan Length.

hard for a wide variety of reasons. The lack of resources can make the route-planning problem dramatically more complex because it interacts with the transportation of multiple packages. Similarly, the capacity limits can interact with fuel shortages to make it necessary to carefully coordinate the actions of carriers to cooperate in the transportation of objects. By varying the size of fuel dumps, the problems can range from simple route planning (with abundant fuel) to complex scheduling of interacting carrier actions (with limited fuel).

The grid domain, used in the second round of the competition, represents a further transportation domain on a grid-shaped network but with constraints on the access to certain locations based on keys of appropriate shapes for the corresponding locks. This domain represents a difficult search domain, primarily because the domain forces the planner to use only one useful action at each layer. Tackling this problem requires an effective filter to remove ground actions, partly to reduce the cost of manipulating the domain itself but mainly to reduce the number of redundant search paths, corresponding to multiple actual paths through the grid itself.

The Future

The first planning competition proved an extremely stimulating event for the planning community. It has brought into sharp focus the state of the art in domain-independent planning and has offered the opportunity to identify several essential issues for the planning community to address. The first of these issues is the development of a common planning domain- description language, currently taking the form of PDDL. Although PDDL must be considered still under development, the effort already invested in its development is an important step toward allowing planners to be usefully compared and in constructing a generally useful repository of planning-domain problems. Perhaps the closest the community



Figure 4. Plan Length against Planning Time (Movie Domain Ignored and Data Cut Off at 100 Seconds).

has come to having such a resource in the past is the collections of problems included with particular planner releases, where those planners were widely used (UCPOP being an obvious example [Penberthy and Weld 1992]).

PDDL has not yet been put to the test in its provision of HTN expressiveness, and questions remain about the ADL components of the language. In particular, it has been proposed that nested conditional effects are an unnecessary element of the language. Provision for the expression of resource-constrained planning problems also remains untested.

These observations highlight a second issue for the community to confront: It remains difficult to compare planners on an equal footing. Planners can differ widely in terms of the expressiveness of the domain description language they handle, the expressiveness of the plans they produce, the speed of planning, and the range of domains they can successfully tackle. To make coherent progress in the field, it is necessary to be able to compare potential advances, attempt to coalesce different but compatible approaches, and avoid repeated redevelopment of the same basic planning software tools at dozens of different sites. A common domain description language is only one step in addressing this issue. It also requires that components of planning systems be made available to the community, particularly in stable and adaptable forms. PDDL parsers are already being made available, and some of the code used in the competition is available as source to be modified, extended, or adapted. These steps are essential in supporting the efforts of the community to advance beyond the current state of the art.

A third issue arises from the hope to push the boundaries of the current state of the art in planning: The benchmark domains that are used to establish the current levels of performance and set targets for the next generation of planners must be chosen with care. Many of the standard benchmark domains were designed with specific agendas. Often, they



Figure 5. Problem File Size against Time (Plot Only Shows Problems Solved by All the Planners).

were designed to showcase specific expressive features of particular languages and are uninteresting when expressed in STRIPS (the movie domain is one example and Pednault's [1989] BRIEFCASE WORLD another). Others are designed to showcase particular planning strategies (the rocket domain used for GRAPHPLAN, for example [Blum and Furst 1997]) or demonstrate flaws in certain planning strategies (for example, the gripper domain). Although these domains retain some interest for these reasons, it is important for the planning community to look beyond these "simple" problems and identify more significant benchmarks that represent tasks that demonstrate planning's coming of age to the wider research and applications community. The greatest challenge for the community, then, is to take the lessons learned from the competition and from the research that is current and show how planning can move on from these problem domains.

References

Anderson, C., and Weld D. 1998. Conditional Effects in GRAPHPLAN. In *Proceedings of Artificial Intelligence Planning Systems*, 44–53. Menlo Park, Calif.: AAAI Press.

Bayardo, R. J., Jr., and Schrag, R. C. 1997. Using CSP Look-Back Techniques to Solve Real-World SAT Instances. In Proceedings of the Fourteenth National Conference on Artificial Intelligence, 203–208. Menlo Park, Calif.: American Association for Artificial Intelligence.

BLACKBOX. 1998. BLACKBOX Web Site. AT&T Research Laboratories, Florham Park, New Jersey. Available at www.research.att.com/~kautz/blackbox.

Blum A., and Furst, M. 1997. Fast Planning through Planning Graph Analysis. *Artificial Intelligence* 90(1–2): 279–298.

Bonet B., and Geffner, H. 2000. Planning as Heuristic Search: New Results. In Proceedings of the Fifth European Conference on Planning (ECP'99). New York: Springer-Verlag. Forthcoming.

Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A Robust and Fast Action Selection Mechanism for Planning. In Proceedings of the Fourteenth National Conference on Artificial Intelligence, 714–719. Menlo Park, Calif.: American Association for Artificial Intelligence.

Ernst, M. D.; Millstein, T. D.; and Weld, D. S. 1997. Automatic SAT Compilation of Planning Problems. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 1169–1177. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Fink, E., and Veloso, M. 1996. Formalizing the PRODICY Planning Algorithm. In *New Directions in Artificial Intelligence Planning*, eds. M. Ghallab and A. Milani, 261–272. Amsterdam, The Netherlands: IOS.

Fox, M., and Long, D. 1999. The Detection and Exploitation of Symmetry in Planning Domains. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 956–961. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Fox, M., and Long, D. 1998. The Automatic Inference of State Invariants in TIM. *Journal* of Artificial Intelligence Research 9:367–422.

Gazen, B., and Knoblock, C. 1997. Combining the Expressivity of UCPOP with the Efficiency of GRAPHPLAN. In *Proceedings of the Fourth European Conference on Planning (ECP-97)*, 221–233. Berlin: Springer-Verlag. Gomes, C. P.; Selman, B.; and Kautz, H. 1998. Boosting Combinatorial Search through Randomization. Paper presented at the Fifteenth National Conference on Artificial Intelligence, 26–30 July, Madison, Wisconsin.

Hoffmann, J., and Koehler, J. 1999. A New Method to Index and Query Sets. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 462–467. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence. HSP. 1998. HSP Web Site. Universidad Simon Bolivar, Caracas, Venezuela. Available at www.ldc.usb.ve/~hector.

IPP. 1998. IPP Web Site. University of Freiburg, Freiburg, Germany. Available at www.informatik.uni-freiburg.de/~koehler/ipp.html.

Kambhampati, S. 1999. Improving GRAPHPLAN's search with EBL and DDB Techniques. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 982–987. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Kambhampati, S.; Lambrecht, E.; and Parker, E. 1997. Understanding and Extending GRAPHPLAN. In *Proceedings of the Fourth European Conference on Planning (ECP-97)*, 260–272. Berlin: Springer-Verlag.

Kautz, H., and Selman, B. 1996. Pushing

the Envelope: Planning, Propositional Logic, and Stochastic Search. Paper presented at the Fourteenth National Conference on Artificial Intelligence, 27–31 July, Providence, Rhode Island.

Kautz, H., and Selman, B. 1992. Planning as Satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92)*, 359–363. Vienna: Wiley.

Koehler, J.; Nebel, B.; Hoffmann, J.; and Dimopoulos, Y. 1997. Extending Planning Graphs to an ADL Subset. In *Proceedings of the Fourth European Conference on Planning (ECP-97)*, 273–285. Berlin: Springer-Verlag. Korf, R. 1990. Real-Time Heuristic Search.

Artificial Intelligence 42(2–3): 189–211.

Li, C. M., and Anbulagan. 1997. Heuristics Based on Unit Propagation for Satisfiability Problems. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 366–371. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Long, D., and Fox, M. 1999. The Efficient Implementation of the Plan Graph in STAN. *Journal of Artificial Intelligence Research* 10:87–115.

McDermott, D. 1996. A Heuristic Estimator for Means-Ends Analysis in Planning. In Proceedings of the Third International Conference on AI Planning Systems (AIPS-96), 142–149. Menlo Park, Calif.: AAAI Press.

McDermott, D., and the AIPS Planning Competition Committee. 1998. PDDL—The Planning Domain Definition Language. Available at ftp.cs.yale.edu/pub/mcdermott/software/pddl.bar.gz.

Nebel, B.; Dimopoulos, Y.; and Koehler, J. 1997. Ignoring Irrelevant Facts and Operators in Plan Generation. In *Proceedings of the Fourth European Conference on Planning (ECP-97)*, 338–350. Berlin: Springer-Verlag. Nilsson N. 1980. *Principles of Artificial Intelligence*. San Francisco, Calif.: Morgan Kaufmann.

Pearl, J. 1983. *Heuristics.* San Francisco, Calif.: Morgan Kaufmann.

Pednault, E. 1989. ADL: Exploring the Middle Ground between STRIPS and the Situation Calculus. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 324–332. San Francisco, Calif.: Morgan Kaufmann.

Penberthy, J., and Weld, D. S. 1992. UCPOP: A Sound and Complete Partial-Order Planner for ADL. Paper presented at the Third International Conference on Principle of Knowledge Representation and Reasoning (KR-92), October, Cambridge, Massachusetts.

Refanidis, I., and Vlahavas, I. 2000. GRT: A Domain-Independent Heuristic for STRIPS

Worlds Based on Greedy Regression Tables. In Proceedings of the Fifth European Conference on Planning (ECP'99). Berlin: Springer-Verlag. Forthcoming.

Selman, B.; Kautz, H.; and Cohen, B. 1994. Noise Strategies for Local Search. In Proceedings of the Twelfth National Conference on Artificial Intelligence, 337–343. Menlo Park, Calif.: American Association for Artificial Intelligence.

sGP. 1998. sGP Web Site. University of Washington, Seattle, Washington. Available at www.cs.washington.edu/research/projects/ai/www/sgp.html.

Smith, D., and Weld, D. 1999. Temporal GRAPHPLAN with Mutual Exclusion Reasoning. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 326–337. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Smith, D. E., and Weld, D. S. 1998a. Conformant GRAPHPLAN. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, 889–896. Menlo Park, Calif.: American Association for Artificial Intelligence.

Smith, D. E., and Weld, D. S. 1998b. Incremental GRAPHPLAN. Technical Report, TR 98-09-06, University of Washington.

stan. 1998. stan Web Site. University of Durham, Durham, United Kingdom. Available at www.dur.ac.uk/CompSci/research/ stanstuff/planpage.html.

Weld, D. S.; Anderson, C. R.; and Smith, D. E. 1998. Extending GRAPHPLAN to Handle Uncertainty and Sensing Actions. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, 897–904. Menlo Park, Calif.: American Association for Artificial Intelligence.



Derek Long is a lecturer in computer science at Durham University. He joined the department in 1995 after lecturing at University College London for six years. He obtained his doctorate at the Programming Re-

search Group, Oxford University, in 1989, with work exploring reasoning by analogy. Since then, he has pursued interests in reasoning techniques in general and planning in particular. He has worked in close collaboration with Maria Fox for the last 10 years. He is currently exploring automatic domain analysis techniques and the automatic extraction of planning domain features. His e-mail address is D.P.Long@dur.ac.uk.

Henry Kautz is an associate professor with

the Department of Computer Science and Engineering at the University of Washington. He was previously department head of the AI Principles Research Group at AT&T Bell Labs. He is a fellow of the American Association for Artificial Intelligence (AAAI) and a member of the AAAI Executive Council and has received the International Joint Conferences on Artificial Intelligence Computers and Thought Award. He is known for his work on planning and plan recognition, efficient deduction and search, and the logical foundations of AI.

Bart Selman is an associate professor of computer science at Cornell University. He previously was a principal scientist at AT&T Bell Laboratories. He holds a Ph.D. and an M.Sc. in computer science from the University of Toronto and an M.Sc. in physics from Delft University of Technology. His research has covered many areas in AI and computer science, including tractable inference, knowledge representation, natural language understanding, stochastic search methods, theory approximation, knowledge compilation, planning, default reasoning, and the connections between computer science and physics (phase-transition phenomena). He has received four best paper awards at the American and Canadian national AI conferences and at the International Conference on Knowledge Representation. He holds a National Science Foundation Career Award and is an Alfred P. Sloan research fellow.

Blai Bonet received an engineering and master degree in computer science from Universidad Simon Bolivar in Venezuela. Among his interests are planning and scheduling with complete and incomplete information. He is currently studying for his Ph.D. at the University of California at Los Angeles. His e-mail address is bonet@cs.ucla.edu.

Hector Geffner received his Ph.D. from the University of California at Los Angeles with a dissertation that was co-winner of the 1990 Association of Computing Machinery Dissertation Award. He then worked as staff research member at the IBM T. J. Watson Research Center in New York for two years before returning to the Universidad Simon Bolivar in Caracas, Venezuela, where he currently teaches. He is interested in models of reasoning, action, planning, and learning.

Jana Koehler is a project manager at the Schindler Lifts Ltd. Technology Management Center in Switzerland. She is currently developing elevator control software based on AI planning techniques. Before joining Schindler, she worked at the German Research Center for AI and held an assistant professorship at the University of Freiburg, Germany, where she developed the IPP planning system. Currently, her interests are centered on real-time planning and scheduling and software verification.

Michael Brenner worked on IPP from spring 1997 until fall 1998 when he went to Paris, France, where he received a Master's in cognitive science. He is now back at the University of Freiburg as a member of the Graduate School on Human and Machine Intelligence, working on multiagent planning and plan execution in dynamic environments.

Joerg Hoffmann was part of the IPP team during the entire project period, from February 1997 until August 1999. He received a Master's in computer science in March 1999. Currently, he is a member of the Graduate School on Human and Machine Intelligence at the University of Freiburg, where he is developing a new planning system based on heuristic forward search.

Frank Rittinger was a student member of the IPP team for the entire project period. Currently, he is working on his Master's thesis at the chair for software engineering at the University of Freiburg, where he is investigating the security aspects of CORBA with formal methods.

Corin Anderson is a fourth-year Ph.D. candidate in the Computer Science and Engineering Department at the University of Washington. He earned Bachelors of Science in computer science and mathematics from the University of Washington in 1996, and he received a Master's in computer science from the same university in 1998. Anderson's primary interests include web site management, planning and scheduling algorithms, and intelligent systems.

Daniel S. Weld received Bachelor's in computer science and biochemistry at Yale University in 1982. He received a Ph.D. from the Massachusetts Institute of Technology Artificial Intelligence Lab in 1988 and immediately joined the Department of Computer Science and Engineering at the University of Washington, where he is now professor. Weld received a Presidential Young Investigator's Award in 1989 and an Office of Naval Research Young Investigator's Award in 1990 and is a fellow of the American Association for Artificial Intelligence. Weld is on the advisory board of the Journal of AI Research, has been guest editor for Computational Intelligence and Artificial Intelligence, and was program chair of the 1996 National Conference on Artificial Intelligence. Weld founded Netbot, Inc., which developed the JANGO comparison shopping agent (now part of the Excite Shopping Channel); AdRelevance, Inc., an online competitive monitoring service for internet advertisements; and Nimble.com, which develops XML query-processing technology. Weld has published about 100 technical papers on AI, planning, data integration, and software agents.

David E. Smith is a head of the planning and scheduling group in the Computational Sciences Division at NASA Ames Research Center, where he is involved in contingency planning for rover operations and research on temporal planning techniques. Prior to joining NASA, he was a senior scientist at the Rockwell Science Center, where his work led to a commercially fielded system for newspaper imposition planning and to the DESIGN SHEET conceptual design system in use throughout Rockwell and Boeing. He received his Ph.D. from Stanford University in 1985. Current research interests include temporal planning, planning under uncertainty, constraint-satisfaction approaches to planning, and preprocessing techniques for planning.

Maria Fox is a reader in computer science at Durham University. She joined the university in 1995 after 6 years at University College London. She obtained her doctorate in 1989 and has worked in aspects of planning, both theoretical and algorithmic, since that time. Currently, her primary interests lie in the development of automatic domain analysis techniques and their exploitation in efficient planning systems.

Complete Your AI Planning Library with these Volumes from AAAI Press

Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems

Edited by Steve Chien, Subbarao Kambhampati, and Craig A. Knoblock

The International Conference on AI Planning and Scheduling has evolved into the premier forum for researchers and practitioners in planning and scheduling. In recent years, artificial intelligence planning and scheduling have emerged as technologies critical to production management, space systems, the Internet, and military applications. These proceedings contain the papers presented at the conference. While a majority deal with algorithms for planning and scheduling in a variety of environments—be they static or dynamic, deterministic or stochastic, or completely or partially observable—there are also papers on planner implementation and applications of planning and scheduling research.

ISBN 1-57735-111-8, 412 pp., index, \$50.00 softcover

Advanced Planning Technology: Technological Achievements of the ARPA/ Rome Laboratory Planning Initiative

Edited by Austin Tate

This volume presents the range of technological results that have been achieved with the ARPA/Rome Laboratory Planning Initiative. Five lead articles introduce the program and its structure and explain how the more mature results of individual projects are transferred through technology integration experiments to fielded applications. The main body of this volume comprises one paper from each group or project within ARPI. Each of these papers seek to introduce the technological contribution of the group's work and provide a pointer to other work of that group.

ISBN 0-929280-98-9, 233 pp., index, \$55.00 softcover

Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems

Edited by Reid Simmons, Manuela Veloso, and Stephen Smith

The 1998 proceedings includes the work of AI researchers in all aspects of problems in planning, scheduling, planning and learning, and plan execution, for dealing with complex problems. Papers in this proceedings range from new theoretical frameworks and algorithms for planning to practical implemented applications in a variety of domains.

ISBN 1-57735-052-9, 244 pp., index, \$50.00 softcover

Proceedings of the Third International Conference on Artificial Intelligence Planning Systems

Edited by Brian Drabble

The 1996 organizers have tried to bring together a diverse group of researchers representing the various aspects and threads of the planning community. As with all previous AIPS conferences, the papers have been selected on technical merit. They include practical algorithms for achieving efficiency in planning, formal results on the completeness and complexity of planning domains, classical planning, formal specification of planning knowledge and domains, constraint satisfaction techniques and their application, reactive planning, and repair and consistency checking in schedules.

ISBN 0-929280-97-0, 292 pp., index, \$55.00 softcover

Proceedings of the Second International Conference on Artificial Intelligence Planning Systems

Edited by Kristian Hammond

The papers in this work present current state-of-the-art research in AI planning systems. Reviewed papers present research on how to generate plans to succeed in uncertain environments, improving robot plans during their execution, managing dynamic temporal constraint networks, and solving time-critical decision-making problems.

ISBN 0-929280-56-3, 360 pp., index, \$45.00 softcover

To order, call 650-328-3123 or send e-mail to orders@aaai.org, or visit our website at www.aaai.org/Press/ AAAI MEMBERS: DEDUCT 20%!

The AAAI Press ■ 445 Burgess Drive ■ Menlo Park, California 94025