Proceedings of the 2007 IEEE Symposium on
Computational Intelligence and Data Mining (CIDM 2007)

Using Data Mining to Enhance Automated Planning and Scheduling

Jeremy Frank
NASA Ames Research Center
Mail Stop N269-3
Moffett Field, CA 94035-1000
frank@email.arc.nasa.gov

Abstract

Automated planning is a combinatorial problem that is important to many NASA endeavors, including ground operations and control applications for unmanned and manned space flight. There is significant value to integrating planning and data mining to create better planners. We describe current work in this area, covering uses of data mining to speed up planners, improve the quality of plans returned by planners, and learn domain models for automated planners. The central contribution of this paper is a snap shot of the state of the art in integrating these technologies and a summary of challenges and open research issues.

1. Introduction

Automated planning and scheduling has been applied to a wide variety of endeavors across NASA, including the scheduling of ground-based, aircraft-based and space-based telescopes; on-board scheduling of deep space craft; and ground-based planning of deep space missions such as the Mars Exploration Rovers Spirit and Opportunity. Plans and schedules are generated in the face of constraints on activity time and resource needs. Automated planning and scheduling systems are designed to search for plans or schedules that achieve a set of goals and also obey a set of constraints. These constraints are provided as inputs to the automated planning system in the form of a declarative model. The goal of the planner can be to produce short plans, plans that minimize resource use, or plans that maximize an objective, e.g. collect the maximum number of science targets. The intent is to build general purpose automated planners and schedulers that can solve problems for any model; that is, the algorithms are designed to accept as input any model described using a specific language. While there are important distinctions to make between planning and scheduling, we will refer to such systems as planners throughout. Planning problems can be computationally easy to solve, but most planning problems of interest are either NP-complete or PSPACE-complete.
A complete discussion of techniques for a variety of automated planning problems can be found in [Gh04].

Data mining is sorting through data to identify patterns and establish relationships. Data mining is considered "inductive learning" (data intensive process) vs "deductive learning" (knowledge intensive.) Data mining parameters include:

* Association - looking for patterns where one event is connected to another event.
* Sequence or path analysis - looking for patterns where one event leads to another later event.
* Classification - looking for new patterns.
* Clustering - finding and visually documenting groups of facts not previously known.
* Forecasting - discovering patterns in data that can lead to reasonable predictions about the future.

In this paper we will focus on how data mining techniques have been used to create better automated planners. We first describe a variety of ways in which data mining can improve various types of planning, then describe current work in this area, and finally describe some outstanding challenges. Those interested in planning and learning in general should refer to the International Conference on Planning and Scheduling 2004 tutorial by Borrajo and Veloso.

2. Automated Planning and Scheduling

The simplest formulation of planning employs the STRIPS formalism [Fi71]. In this formalism, a planning domain model consists of a set of propositions P and a set of actions A. The propositions in P can be true or false, and implicitly describe a state space S such that the set of true propositions s in 2^P. The actions A form a mapping from S to S. Each action a in A is concisely described by three lists of propositions: a precondition list, an add list and a delete list. If action a is executed when the propositions on the precondition list (and possibly others) are true, then the propositions on the add list are made true (regardless of their prior value), and the propositions on the delete list are made false (regardless of their prior value). The outcome of executing the action under other circumstances is undefined. A planning problem consists of a planning

domain model, a set of propositions that are assumed to hold, and a set of propositions all of which must be true after execution of an action sequence. The problem is to find such an action sequence or prove that no such action sequence exists. A sample is shown in Figure 1.

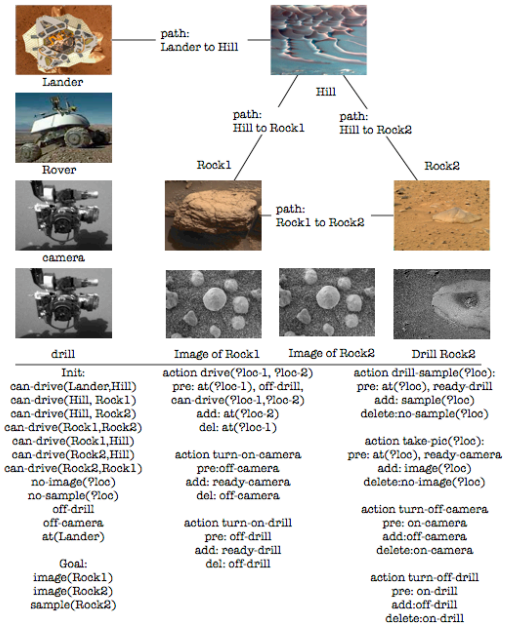


Figure 1. A simple planetary rover domain and its STRIPS model. The domain consists of static locations at which the rover can be, routes between locations that can be navigated by the rover, and goals for the rover to achieve, e.g. take images or drill samples at different locations.

STRIPS is somewhat inconvenient for formulating models in which metric time and resources are required; propositions must capture the exact time actions occur, or the exact amount of available resource. Modern plan domain description formalisms such as PDDL [Fo03] allow richer description of conditions and effects that allow concise declarations of resource availability, action concurrence, and arbitrary temporal constraints between actions (e.g. action *a* must start 5 minutes after *b* ends). Further refinements allow specification of plan quality functions such as minimization of makespan or plan steps, maximization of the value of achieved goals, and other more elaborate goals. Additional refinements permit specification of planning problems in which actions have uncertain outcomes and the initial state may not be known with certainty.

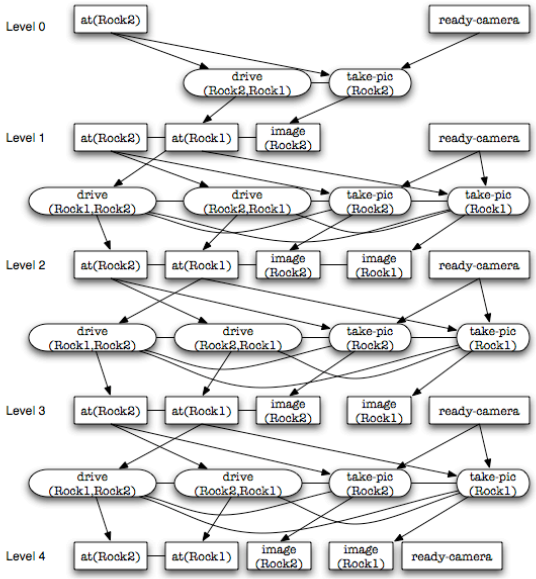


Figure 2. A partial plan-graph for the planetary rover domain. This plan-graph is limited to propositions that mention the camera, rover position and take-image goals. It is limited to 4 levels. Arcs indicate mutual exclusions between propositions or actions, and arrows indicate action preconditions and effects.

We introduce some concepts that will appear later in the paper. Many planners assume all actions are totally ordered; such planners can find feasible plans, but ignore the possibility that two actions can be concurrent. Partial order planners account for action concurrency, and are superior when the makespan of the plan (longest path of totally ordered actions) must be minimized, but have proven to be slow when searching for feasible plans. The *plan-graph* [BlFu97] is a commonly used tool in automated planning; it is a compact representation of a superset of all feasible plans, and can be used in many ways to provide heuristic guidance to planners. The plan-graph is constructed as follows. The first level consists of the propositions defining the initial state. The next level consists of all applicable actions (an action is applicable if all propositions in the precondition list appear in the previous level, and no two are marked as mutually exclusive). Actions *a* and *b* cannot be concurrent if *b* deletes *a* precondition of *a* or a proposition on the add list of *a*. Actions that cannot be concurrent are marked as being mutually exclusive. The following level consists of all the propositions in the initial level, as well as all effects from actions in the second level. Pairs of propositions that can only be achieved by mutually exclusive actions are marked as mutually exclusive. This process continues until the set

of propositions and mutual exclusion annotations does not change between successive levels. A plan-graph for the planetary rover domain is shown in Figure 2. Feasible plans must be extracted from the plan-graph, but the calculation of the plan-graph has been shown to significantly reduce the work of finding such plans. The plan-graph cannot produce minimum makespan plans directly since it assumes all actions at level *k* precede all actions at level *k+1*, but can produce plans of low makespan. A *relaxed plan-graph* [Ho01] is a plan-graph for a problem instance in which the delete lists for all actions have been omitted. While both plan-graphs and relaxed plan-graphs can be calculated in polynomial time, the relaxed plan-graph can be computed much faster, and have been used extensively in heuristics for planners. A relaxed plan-graph for the planetary rover domain of Figure 1 using the same initial propositions as those used in Figure 2 is shown in Figure 3.

3. Habeas Datum: The Operators' Perspective

Planning is employed in operational settings ranging from human decision makers deciding on courses of action to controllers generating actions for robots or automated systems. Data mining can improve planning in one of several ways:

* Data mining can be employed to increase the speed of planners.
* Data mining can be employed to improve the quality of results returned by planners.
* Data mining can be used to learn the rules for domains in which there is inherent uncertainty.

In this section we will discuss how the operational use of planners influences the sources of data used to improve planners.

The principal source of data used to improve a planner will come from running that planner. Running the planner generates plans; it may also generate large amounts of other information such as planning decisions and the output of simulations, checkers or human validation of final plans. Data generation and data mining can be done "offline" i.e. prior to the use of the planner in operations, or "online", i.e. while the planner is operating. Plan generation can be "cheap", in the sense that many plans can be created in a short period of time, or "expensive"; while cheap for academic settings, plan generation is often expensive in real applications. In some circumstances, especially when learning domain rules, plans may not be generated by automated planners, but provided from some other source to the data mining system.

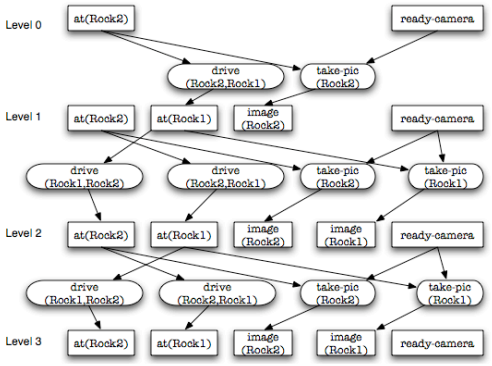


Figure 3. A partial plan-graph for the planetary rover domain. This plan-graph is limited to propositions that mention the camera, rover position and take-image goals. No new propositions are achieved after 3 levels. Arrows indicate action preconditions and effects.

Making use of data gathered in online settings is critical. By some measure, the best examples will be gathered in an online setting, as it is here that the most relevant planning problems are solved. At design time, this may require accepting some costs to ensure that relevant information is collected from applications for use by offline data mining systems as part of the revision process. This is especially important when considering the use of data mining to speedup planning; users will have little idea how to control combinatorial search in planning, although they will generally be able to suggest plans they believe to be valid or optimal, which provides clues to domain specific heuristics. The issue of labeling examples plays into this design decision. Some data collected during planning can be labeled with no action on the part of users (e.g. valid vs invalid plans, decisions leading to backtracking, plan quality measurements). However, this is not universally the case. Most planners and domain models are imperfect, requiring users to modify automatically generated plans. For example, the plan quality function may be missing a critical feature. Users of planners will often not have time to suitably label examples. Such unlabeled examples are suitable for driving clustering algorithms; data mining can, for example, be used to suggest changes to the plan quality function based on the plans chosen by the user instead of the suggested "optimal" plan.

While most of this paper will focus on planners used by people, it is worth mentioning that planners used onboard robots or as parts of control systems will be severely resource constrained; while this does not preclude data mining, it significantly limits the types of online data mining that can be used to improve planner performance. Memory and processor time are often at a premium; as a consequence, most data mining enhancements will have to be done offline.

A significant advantage of data mining applied to planning is that there are few to no challenges due to missing data: the data collection is almost completely under the control of the application developer. One exception to this is explicit user labels of plan quality, but this can often be circumvented; plans that are discarded, modified or actually used can easily be labeled with little or no burden on users. Furthermore, if time is available, classifiers can be rebuilt any time after new examples are generated, allowing for much finer control in response to new data. Finally, data mining systems can potentially invoke the planner to generate new instances, an issue we will return to at the end of the paper.

4. Speeding up Planners using Data Mining

The use of data mining to generate macros or plan libraries is a common method of speeding up planning. Macros are interpreted as real actions; that is, they must be formally derived so that they can be used in plans with the same guarantees as the actions in the original domain model. Plans from a library intended for reuse, however, are subject to fewer restrictions; they can be interpreted as heuristics in that the planner can modify the plan from the library arbitrarily. We will describe work on macros, and observe that similar issues apply to generating plans for reuse.

The MacroFF planner of Botea et al. [Bo05] employs a three step process for generating macros: 1) generation of types, 2) generation of macros using types, 3) evaluation of macros for relevance. The solutions to the simplest problems in a domain are mined to evaluating macros. For these simple problems, all macro operators are added to the domain, giving each macro a chance to participate in a solution plan and increase its weight. For ranking, each macro operator is assigned a weight that estimates its efficiency. All weights are initialized to 0. Each time a macro is present in a plan, its weight is increased by the Figure 4. number of occurrences of the macro in the plan (occurrence points), plus 10 bonus points. The occurrence points decide the relative ranking of common macros.

Local search is often confounded by plateaus, which are regions of the search space where neighboring solutions have identical quality, therefore providing no guidance to search algorithms. Another local search-based planner using data is the Marvin system of Coles and Smith [Co04], which learns macros that local search planners use to escape plateaus. When the start of a plateau is detected—that is, when no successor state with a strictly-better heuristic value can be found—best-first search commences from the current state. During best-first search, each successor state stores the actions that have been applied to reach it since the start of the plateau: when a strictly-better state is eventually found, this list of actions is the plan segment that forms the basis of the plateau-escaping macro-action. Macro actions are simplified to enhance reuse; parallel threads are separated and useless actions are eliminated. Macros are then candidates for escaping plateaus that are encountered in the future; they are tried after the original actions in the domain. The macros apply to all instances in a domain, and so they can be reused on other instances.

A number of issues have been identified when mining macros and plan libraries. Large numbers of macros or plans may be collected, especially specific ones that are not widely applicable. If added to domain models with impunity, increased memory and matching time may result in poor planner performance. Generalization can lead to elimination of redundant macros or plans, or they can be discarded due to age or likelihood to apply. However, generalization across problems in domains is sometimes difficult, and may limit their ultimate utility.

Local search-based planners employ data mining techniques to guide search. An example of such a planner is SG-Plan [ChWa03]. This system poses planning as a constrained-optimization problem. This problem is solved by transforming the problem into an unconstrained optimization problem with an extended set of variables, each of which is interpreted as a Largrange multiplier or penalty on solutions that violate a constraint. SG-Plan then uses local search to find plans that may violate one or more constraints, and adjusts the Lagrange penalties in a manner that adjusts the penalty on violated constraints. Subsequent searches are less likely to violate these constraints in the future. The central issues with this technique is the speed of convergence; memory is not a problem. While theoretically shown to converge, significant tuning is required to make such algorithms perform well in practice, and by alone the technique is not enough to solve large problems.

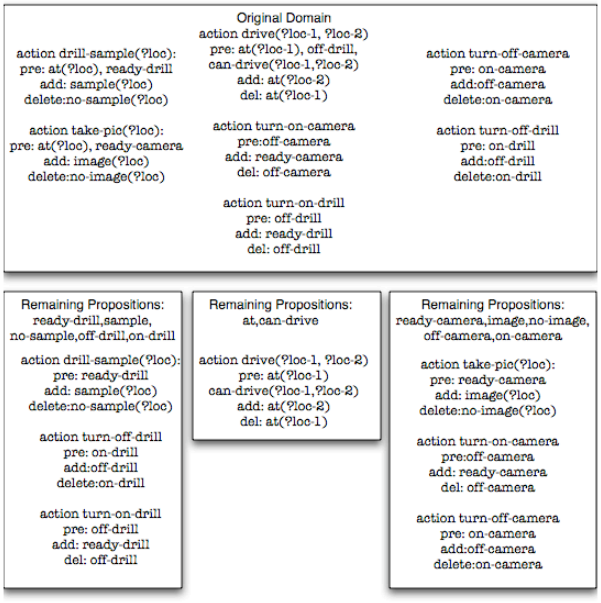


Figure 4. Creating pattern databases for the planetary rover domain. Three different simplified problems are shown: one consisting of propositions only mentioning the state of drilling goals, one consisting of propositions only mentioning locations, and one consisting of propositions only mentioning image goals.

Data mining can be used to generate variable and value ordering heuristics, which are critical to planner speed. A pattern database [Ed01] is constructed by first simplifying a planning problem; this can be done by eliminating propositions from the domain, thereby simplifying the actions. The resulting simple problems can be solved to completion on smaller problems; an example is shown in Figure 4. The pattern database is used to solve larger problems by simply determining whether the relevant part of the state resulting by application of a candidate action matches a plan in the database, and combining the quality of the resulting plans as a heuristic estimate. A useful property of pattern databases is that, under certain conditions, they can be used to create admissible heuristics. This is desirable since the first feasible plan found by the A* algorithm using admissible heuristics is also an optimal plan. Since it is difficult to know in advance how to construct good pattern databases, they are built by mining the set of all possible abstractions.

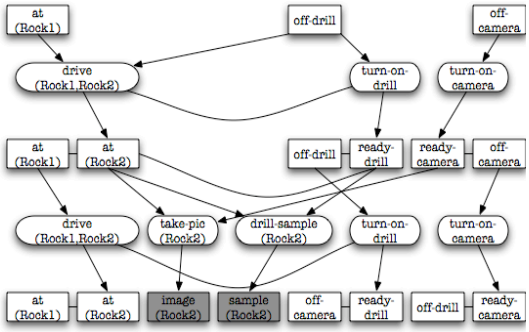
Yoon et al. [Yo06] also use data mining to learn heuristics. Their approach is to generate data from the relaxed plan-graph for a problem instance. Propositions are annotated to create new facts indicating that propositions are present in preconditions, add or delete lists of actions in the plan, and goals for the instances. Features of the database correspond to sets of propositions that have particular properties, e.g. the set of observations to schedule in a rover planning problem. The function approximation problem is to learn features that are correlated with minimum makespan plans that are generated separately. Thus, the relaxed plans are used to generate features that are correlated to minimum makepsan plans; this function is used to evaluate plans resulting from candidate actions.

Vrakas et al. [Vr03] used data mining to learn which planner solved a problem most effectively. All planners so analyzed were derived from a single A*-based planning algorithm; the parameter variations tested included features such as heuristic weight settings, whether the planner performs progression or regression search, and an agenda size setting (allowing beam-search algorithms). The resulting planner was derived from 437 configurations; the resulting planner was an improvement over each individual configuration.

A number of issues have been identified when using data mining to generate heuristics. The key problem is quality of the approximated functions, and the related problem of trying to generate data that leads to better heuristics. Admissibility can be a problem as well; while pattern databases can be used to generate admissible heuristics, and "controlling" planners using admissible heuristics leads to planners that are admissible, not all data mining approaches will guarantee admissibility. It can also be difficult to guarantee accuracy and admissibility in the same heuristic.

In addition to learning heuristics, it is also possible to learn "nogoods", i.e. constraints on the solution space that lead to early backtracking, thereby potentially speeding up planning. Khambampati [Ka00] describes nogood recording techniques from CSPs employed to enhance Graphplan-based planning. The first step in this process requires transforming the plan-graph into a constraint satisfaction problem (CSP); this is done by recognizing that each action at each level of the plan-graph can be either present or absent in the final plan, and similarly, that each proposition at each level of the plan-graph can be true or false in the final plan. Constraints are derived from the mutexes to ensure that only non-conflicting actions and propositions are present in the plan. As dead-ends are found in CSP, the planner acquires and minimizes nogoods, consisting of sets of actions that lead to constraint (mutex) violations. An example is shown in Figure 5. The search for plans normally generates nogoods for a single instances. Kambhampati et al. [Ka97] describe a similar method for learning nogoods for partial order planners, and Selman and Kautz employ transformations of planning into propositional satisfiability, making it possible to use SAT-solvers that learn nogoods [SeKa99]. Upal [Up03]

generalized this technique to acquire nogoods for reuse across problem instances by including the initial state assertions in the nogoods. Nogood learning can be used in the same manner by planners that use propositional satisfiability representations instead of CSP formulations [SeKa97].



1. Achieve image(Rock2) by take-pic(Rock2)
2. Achieve sample(Rock2) by drill-sample(Rock2)
3. Achieve at(Rock2) by drive(Rock1, Rock2)
4. Achieve ready-drill by turn-on-drill
5. Achieve ready-camera by turn-on-camera
6. Conflict between drive(Rock1,Rock2) and turn-on-drill causes backtrack
7. Nogood involves at(Rock2), ready-drill and ready-camera
8. Observe that ready-camera is not involved in nogood
9. Simplify nogood to at(Rock2) and ready-drill

Figure 5. No-good learning during planning in the planetary rovers example. This example assumes that plans are extracted from a plan-graph of level 2.

The issues with data mining for nogoods are similar to the issues with mining for macros or plan library components. Small nogoods (consisting of few actions) are more general and easier to store, but nogood minimization is computationally expensive. This effort must be balanced against simply continuing to search, especially when solving a single problem instance. Large numbers of nogoods may be collected, especially specific ones; both memory and checking time increases. Generalization can lead to elimination of redundant nogoods, or they can be discarded due to age or likelihood to apply.

Planning with local search can be an effective strategy for avoiding problems with combinatorial search spaces. Local search algorithms work by performing a limited number of modifications to a proposed plan, and choosing a modification that improves upon the current plan. This strategy works well when there are many feasible plans, and the task is to search for optimal plans. One difficulty with local search is to create the set of modification operators; it is important to ensure there are sufficient modification operators to guide search, but if there are too many such operators, search performance will be degraded. Ambite and Knoblock [Am00] describe a way to use data mining to automatically generate search control operators for a particular planning domain. They automatically generate optimal partial order plans for small problem instances for domains, then mine these plans to generate plan rewrite rules. Rules are generated by analyzing the differences between initial plan and final plan; facts only present in the initial plan form the antecedent of the rule, facts only present in the final plan for the consequent of the rule. Proposed rules are evaluated in order of their size; if a proposed operator improves at least one plan, it is retained, otherwise it is not. This process terminates when no improving rules are found.

A number of issues have been identified when mining plans for controlling local search. Generation of too many local search operators is problematic; either all plan modifications must be considered, or some means of controlling the time to generate alternatives must be considered.

5. Improving Plan Quality using Data Mining

Most of the techniques described in the previous section have an impact on plan quality as well as planner speed; since planning is often computationally expensive, the ability to find feasible plans more quickly translates to the ability to find better quality plans in the same allotted time.

Another possible problem in local search is the presence of long sequences of unproductive plan modifications. This can result from the presence of local optima from which no sequences of purely greedy plan improvements is possible. Boyan and Moore [Bo98] attempt to circumvent this problem by learning good starting points for local search. Figure 6 provides a motivation for why this is a good idea; one could simply learn to start so close to the global optimum that greedy search will solve the problem. Boyan and Moore mine sequences of local search moves, and learn a function that predicts the properties of a starting point for local search that will lead to a good solution. Search proceeds in two phases: one greedy optimization phase selects a good starting plan, and a second greedy optimization phase starts at that plan and produces a solution to the problem. They observe that if the local search is Markovian, i.e. the successor plan only depends on the current plan, then every plan visited can be used as training data. This assumption applies directly to most

local search algorithms; with some extensions it also applies to tabu search and other search techniques that employ memory. The resulting augmentation outperforms other local search algorithms that do not use adaptation on some simple scheduling problems.

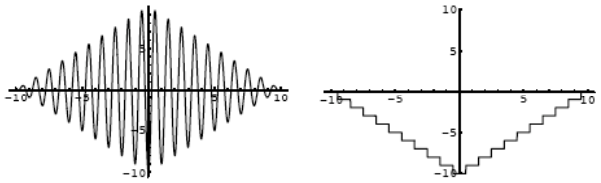


Figure 6. A simple one-dimensional optimization problem with many local minima, and the function that predicts the expected value of purely greedy optimization starting at any point. Reproduced from [Bo98].

Optimal planning problems in which infeasible plans are common offer other challenges to local search; should local search enter infeasible regions or not? If so, how to balance repairing constraints and feasible solution quality? Rogers et al. [Ro06] mine the trajectories of local search operations performed by planners for satellite observation scheduling problems and classify trajectories based on the presence of cycles, infeasible plans along trajectories, proximity of those infeasible plans to the set of feasible plans, and the presence of "shortcuts", i.e. paths from a poor quality plan to a better plan that include infeasible plans. The analysis showed that search in infeasible regions was a profitable course due to the presence of shortcuts; algorithm modifications using tabu lists to allow limited search in the infeasible region leads to improved search quality.

6. Learning Domain Rules using Data Mining

In the previous sections we have implicitly assumed that a domain model is available, and correctly represents the actual rules of the planning domain. This is often not the case: domain models may be inaccurate, may change as time passes, or may not be directly available. The field of data mining has also been applied to learning domain rules for planning. In this section we interpret the notion of "learning the domain" loosely, and describe work both in learning action rules for planning domains as well as learning plan quality functions.

Gervaiso et al. [Ge99] use machine learning to learn a user's preferences for feasible plans. The system takes as input users' feedback on plans that are generated, and learns a function that mimics those preferences. Using this feedback produces a system that, over time, generates better responses according to user studies.

Pasula et al. [Pa04] describe a data mining application designed to infer the rules of a planning domain in which action outcomes are uncertain from a set of plans. Given a data set *D*, a proper rule set *R* includes exactly one rule that is applicable to every example *d* in *D* in which some change occurs, and that does not includes any rules that are applicable to no examples. Rule sets are constructed using a scoring function that favors rules that fit the data well and penalizes overly complex rules, which in this case corresponds to many preconditions and many possible outcomes. Rule sets are constructed by generating candidate rules from old ones using a small set of operators; the new rule maximally increasing the score is added to the new set. The approach is shown to compare favourably to using Dynamic Bayes Networks to learn the domain rules.

Winnner and Veloso [Wi02] describe a data mining approach to generating domain specific planners from example plans; planners employ a very limited programming language which is nonetheless powerful enough to create concise, efficient planners for specific domains. Their DISTILL planner maintains a family of domain specific planners which are incrementally built by analyzing sample plans. These plans are transformed into programs that use the simple programming language in a manner that is biased towards efficient inclusion in an existing domain specific planner, then merged with such a planner.

7. Here Be Dragons: Challenges in Applying Data Mining to Planning

The problems of poor generalization, over-fitting function approximation, feature selection and speed of function approximation are not specific to data mining applications employed for planning. However, there are several reasons why data mining may be unsuitable or difficult for planning due to the peculiarities of planning problems.

When data mining is performed in an offline setting, the speed of function approximation and the data storage problems of data mining are mitigated to some degree. The function utilized by the planner will certainly be more compact than the data used to generate the function, and planner performance will not be penalized for the time spent to perform function approximation.

When used in online settings, efficiency and predictability of the behavior of the planner is vital. Operators are trained to expect certain behavior, and so "self-adapting"

planners that employ data mining must be built with caution. For planners employing data mining techniques online, the concerns of speed and memory are even more serious; the design of operational planners can be seriously impacted if data mining leads to significant slowdowns in the time spent planning. Time and resource constraints may preclude expensive data mining operations even if future planner behavior may be improved.

If opportunities to revise planner behavior are present, data mining can be employed to create the "next release". Even in this setting care must be exercised. If plans generated by users are destined for use by offline data mining, applications must be designed to capture these plans without impacting the user experience. This can be done even for planners used in onboard control settings.

There are more technical challenges as well. Algorithms such as A* are prominently used in automated planning; this is because, when certain representational assumptions are made and A* is used with admissible heuristics, the first plan returned is the shortest (or otherwise optimal) plan. However, to preserve this feature, data mining applications that are used to construct heuristics must return admissible heuristics or sacrifice the guarantee. As we have seen, approaches such as pattern databases can preserve these guarantees, but this limits the types of data mining techniques that can be used.

When employed for learning domain rules and heuristics, data mining must work with highly structured data (e.g. plans) and output highly structured functions (e.g. rules or mappings from rules to quality measures). Again, while the papers referenced here show this is possible, highly structured data place special constraints on data mining that may preclude the use of some data mining techniques.

Data mining can suffer from the problem of "insufficient examples". Curiously, in many of the applications of data mining to planning, this problem does not arise or has been circumvented. Nogood recording benefits from dead-ends in search, which is the common experience. Data mining applied to local search has focused on leveraging data available in average search trajectories (e.g. plateau escaping macros, detours and shortcuts to improving solutions.) When solved problems for a domain are needed, small planning problems provide such examples; there is no concrete evidence suggesting they do not generalize.

The International Planning Competition is a forum in which automated planners are tested on common problems. (Interested readers should refer to the Journal of Artificial Intelligence Volume 24, dedicated to the 2004 competition.) Awards are given based on planner speed, planner coverage (i.e. number of problems and domains that they can solve) and solution quality. Given the prevalence of data mining techniques applied to planning, it is worth pointing out that SG-Plan [ChWa03] and SAT-Plan [SeKa99] have employed some of the techniques we describe in this paper and performed well in the planning competition. However, these planners only use a limited number of data mining techniques. Other competitors that have performed well have not employed such techniques: there is clearly more scope for data mining to improve planning in the future.

8. Challenges for Planning and Opportunities for Data Mining

In closing, we describe some opportunities for data mining to improve the quality of planning.

First, we note that many of the techniques we describe above employ targeted generation of data that is mined using knowledge-intensive techniques to improve planning. These techniques may be complemented by, or improved upon, using techniques applied to unstructured data. For example, clustering techniques may improve the grouping of predicates used to generate pattern databases, and pattern recognition techniques may guide heuristics that trigger nogood learning or minimization only when it is expected to maximize expected benefit to a planner.

A related opportunity concerns the opportunity presented by using the planner to generate more data for data mining. Most of the techniques described here, even those exploiting a planner to gather data, make no explicit effort to influence the dataset. In an offline setting, the data mining system can simply run the planner again in order to generate more data if necessary to improve the planner even further. This suggests that the data mining system try to limit the effort spent improving the planner by focusing on generating data where it will do the most good. Doing so will require trading off exploration and exploitation, in a manner reminiscent of experiment generation and k-armed bandits problems.

Next, we note that many NASA and industrial planning problems require validation of plans produced by AI planners. This is usually because the declarative languages used by such planners are inadequate to capture the full domain description, or because modeling shortcuts are needed to enable automated planning. Often validation is done through the use of process simulations. Examples of such simulation methodologies include Discrete Event Simulations (DES), Testability Analysis, and Reliability models (e.g., Fault Trees.) A variety of software packages have been developed both within NASA and commercially based on these simulation methodologies. Commercial examples include: Arena, Extend, CORE, Satellite Toolkit

(StK); government lab (including NASA) developed simulators include Sapphire 7, Mission Simulation Facility [Pi04] and ROver Analysis Modeling and Simulation (ROAMS) [Ye99].

Finally, automated planning can incorporate such high fidelity simulation while generating plans, but the computational expense of such simulations often precludes doing so except in the coarsest possible way: validation of the complete plan after plan generation is complete. Such loose integration is undesirable since many plans may be infeasible, and the high expense of such plan validation leads to slow planning. Even when tight integration is possible, expensive validation steps will slow planning. Data mining can be used to improve the quality of planning in several ways. First, the validator can be viewed as a function that can be approximated in order to speed up the planner. In this case, data mining of the validator can produce an efficient, approximate function that can serve as a fast, first cut once automated planning is complete. Even more valuable would be an approach that can be used much earlier in the planning process in a manner similar to nogood learning; this approach can lead to early termination of planning. Second, data mining can be of use in building heuristics that guide the planner towards feasible solutions or higher quality solutions.

9. References

[Fi71] Fikes and Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, Artificial Intelligence, Vol. 2, 1971, pp 189-208.

[Fo03] Fox, M. and Long, D. "PDDL2.1: An extension of PDDL for expressing temporal planning domains." Journal of Artificial Intelligence Research. 20, 2003: p. 61-124.

[Sm04] Smith, A, and Coles, A. MARVIN. Proceedings of the 24th UK Special Interest Group on Planning, 2004.

[Ed01] Edelkamp, S. Planning with Pattern Databases. Proceedings of the European Conference on Planning, 2001

[Bo05] Botea A., Enzenberger M., Müller M., and Schaeffer J. Macro-FF: Improving AI Planning with Automatically Learned Macro-Operators. Journal of Artificial Intelligence Research, 24, 2005: p. 581-621.

[Ka00] Kambhampati, S. Planning Graph as (dynamic) CSP: Exploiting EBL, DDB and other CSP Techniques in Graphplan. Journal of Artificial Intelligence Research 12, 2000 p. 1-34.

[Up03] Upal, M. A. Learning Graphplan Memos through Static Domain Analysis. Proceedings of the Sixteenth Canadian Conference on Artificial Intelligence, Springer Verlag, New York, 2003.

[Ho01] Hoffmann, J. and Nebel, B. The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research, 14:253–302, 2001.

[Gh04] Ghallab, M., Nau, D. and Traverso, P. Automated Planning: Theory and Practice. Morgan Kaufmann, 2004.

[Ka97] Kambhampati, S., Katukam, S. and Qu, Y Failure Driven Dynamic Search Control for Partial Order Planners: An explanation-based approach. Artificial Intelligence. 88(1-2) p. 253-313, 1997

[Vr03] Vrakas, D., Tsoumakas, G., Bassiliades, N., and Vlahavas, I. Learning Rules for Adaptive Planning. Proceedings of the 13th International Conference on Automated Planning and Scheduling, 82-91, 2003.

[Am00] Ambite, J. L., Knobock, C. and Minton, S. Learning Plan Rewriting Rules. Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling Systems, 2000.

[Ro06] Rogers, M.F., Howe, A. and Whitley, D. L. Looking for Shortcuts: Infeasible Search Analysis for Oversubscribed Scheduling Problems. Proceedings of the 16th International Conference on Automated Planning and Scheduling, 2006

[Ge99] Gervaiso, M. and Iba, W. and Langley, P. Learning User Evaluation Functions for Adaptive Scheduling Assistance. Proceedings of the International Conference on Machine Learning, 1999.

[Pa04] Pasula, H. M. and Zettlemoyer, L. S. and Kaelbling, L. Learning Probabilistic Relational Planning Rules.. Proceedings of the 14th International Conference on Automated Planning and Scheduling, 2004

[Wi03] Winner, E. and Veloso, M. DISTILL: Towards Learning Domain-Specific Planners by Example. Proceedings of the 13th International Conference on Automated Planning and Scheduling, 2003

[Pi04] Pisanich, G., Plice, L,, Neukom, C., Fluckiger, L, and Wagner, M. Mission Simulation Facility: Simulation Support for Autonomy Development. Proceedings of the 42nd AIAA Aerospace Sciences Conference, Reno, NV, January 2004.

[Ye99] Yen, J., Jain, A., and Balaram., J. ROAMS: Rover Analysis, Modeling and Simulation. In proceedings of *Artificial Intelligence, Robotics, and Automation in Space,* 1999.

[ChWa03] Y. Chen and B. W. Wah, Automated Planning and Scheduling using Calculus of Variations in Discrete Space, Proc. International Conference on Automated Planning and Scheduling 2003.

[Bo98] Boyan, J. and Moore, A. Learning Evaluation Functions for Global Optimization and Satisfiability. Proceedings of the National Conference on Artificial Intelligence, 1998.

[KaSe99] Henry Kautz and Bart Selman. Unifying SAT-based and Graph-based Planning Proc. International Joint Conference on Artificial Intelligence, 1999.

[BlFu97] A. L. Blum and M. L. Furst. Fast planning through planning graph analysis. Artificial Intelligence Journal, 90(1-2), 1997, p. 225-279.