PLANNING AND SCHEDULING

AMEDEO CESTA

Istituto di Scienze e Tecnologie della Cognizione – Consiglio Nazionale delle Ricerche, Roma

ALFREDO MILANI

Dipartimento di Matematica e Informatica – Università degli Studi di Perugia

1 Introduction

The design of autonomous systems able to interact with the environment to accomplish complex goals, such as intelligent agents, autonomous robots and unmanned vehicles, requires to reason about how and when to act, and about the effects of system actions in the world and along the timeline. Automated Planning and Scheduling (P&S) [26] is the area of Artificial Intelligence which concerns models and techniques for synthesizing and maintaining plans of actions, strategies of execution, and resource allocation policies which allow us to realize complex tasks. More precisely, planning is the process of synthesizing sequences of actions or activities which accomplish a desired goal from an initial state, while scheduling focuses on the process of allocating given activities and resources in time while respecting various types of constraints such as capacity, precedence and duration.

Planning. Since its very first origin with the classical work of Fikes et al. [22], automated planning has been given a formal logical characterization consisting in an agent-centric point of view, where the agent executes actions which modify the world allowing it to obtain a desired situation or goal. Nevertheless, the toy problems typical of classical literature present challenges which are different from those arising in more realistic application domains. Real-sized planning domains are affected by combinatorial complexity of different nature (see [4, 21] and many others for complexity results) and for this reason a large part of early and current research efforts focus on search control techniques. Some of the approaches use representations and techniques specially designed for general planning domains (e.g., state-space search, plan space search, planning graph based search, heuristic search), while other techniques are based on the reduction of planning to a known problem, such as the Constraint Satisfaction Problem (CSP), model checking or propositional satisfiability (SAT) to take advantage of specialized solvers for those problems. Moreover, the requirement of relaxing various limits and restrictions of the classical planning approach, like atomicity of time and the closed world assumption, led to investigate several approaches, such as models which allow us to specify constraints on time and metric quantities, and models for planning under uncertainty. An influential initiative for the area has been the international planning competition (IPC) held every other year since 1998. The competition has led to a significant improvement in the performance of planners over the years, and has fostered the development of problems with increasingly complex features. Functional to the competition has been the development of PDDL (Planning Domain Definition Language), a language for modeling planning domains.

Scheduling. The scheduling problem has been studied extensively in Management Science, Operations Research and AI. Early AI research focused on the representation of ill-structured domain features which are difficult to capture in terms of mathematical modeling, and on search control through reasoning on domain constraints, e.g., [23]. The collection [47] gives a quite exhaustive perspective on scheduling research from the 80s and early 90s. Similarly to most of AI of the early days, this work shows how several scheduling systems have been successful and influential, e.g., ISIS, OPIS, etc., although the scientific principles underlying that research were mostly hidden within the complexity of the system architectures. AI research on scheduling has significantly evolved over the last fifteen years and the principles behind the early work have emerged, the reproducibility of techniques has been assessed, and a bulk of new results has allowed contributions of AI studies in the multidisciplinary scheduling arena. Very important for this advancement has been the positive interaction of early work with the studies on Constraint Satisfaction (CSP) or Constraint Programming. The distinctions between modeling (variable and constraints), deductions (propagation rules) and decisions (heuristic reasoning and choices) contributed to clarify various aspects related to the integration of techniques, thus allowing us to solve increasingly complex scheduling problems.

```
Operator: Move_to
Parameters: ?vehicle ?from ?to
Preconds: (at ?vehicle ?from) (road ?from ?to)
Effects: (at ?vehicle ?to) at ?vehicle ?from)

Operator: load
Parameters: ?vehicle ?obj ?place
Preconds: (at ?obj ?place) (at ?vehicle ?place)
Effects: (into ?vehicle ?obj) not(at ?obj ?place)

Operator: unload
Parameters: ?vehicle ?obj ?place
Preconds: (into ?vehicle ?obj) (at ?vehicle ?place)
Effects:  (at ?obj ?place) not(into ?vehicle ?obj)
```

Figure 1: A simple domain

P&S Applications. Although its origins can be attributed to distinct groups of people, research in P&S has merged into AI to shape a unique Þeld of study. Instrumental in the integration of efforts has been the emergence of a number of application areas in which there is no or very little distinction between causal reasoning, i.e., how to synthesize a plan which achieves a goal (planning), and how to allocate resources for the plan (scheduling). These two aspects of problem solving were not separated but part of a continuum, and the emergence of these domains led to a dedicated research line which is quite complementary to the integration of ÒclassicalÓP&S techniques. For example, a leading application area for this Þeld is space mission support, and one of the most notable projects in this are has been the DS-1 Experiment in 1999 [31] that showed how AI planning and scheduling technology can be integrated in a closed loop for autonomous control of a spacecraft in deep space. The experiment shows how the synthetic aspect of planning, deciding what to do, cannot be easily separated from reasoning on a continuous temporal evolution, e.g., complex temporal constraints, and on the availability of on-board resources for accomplishing the plan.

2 The Classical Planning Problem

A planning problem, according to the formulation usually referred to as *classical planning*, is the problem of Þnding a sequence of actions, or instances of operators, which transforms an initial state into the desired goal state of the world. More formally, a planning problem is given by a 4-ple (A, I, O, G), where A is a set of ßuents, or propositions, used to characterize states. A planning state $s \in S$ is an assignment of truth values to ßuents $s : A \rightarrow \{T, F\}$ which describes a state of affairs in the domain. I and G respectively describe the initial and goal state. O is a set of operators $o \in O$ which transform states into states $o : S \rightarrow S$. A prominent feature which is common to most planning models is the description of domain operators in terms of *preconditions* and *effects*. Operator preconditions are logical formulae over ßuents, which must be veriÞed before the action is executed, while effects establish which conditions hold on ßuents after action execution.

An Example. Suppose that we are modeling a transportation planning domain where three operators, namely *move_to*, *load* and *unload* are available, respectively describing movements of vehicles between cities, and loading/unloading of objects on vehicles. A possible preconditions/effects description of operator $move\_to$ is given in Fig. 1. It establishes that a vehicle can move from a starting location to a destination, if the vehicle is at the starting point and there exists a road connecting the two locations. The effect, i.e. the consequence, of executing the action is that the vehicle is no longer at the starting position, while, in the new state, it is at the destination location. Similarly, $load$ establishes that an object can be loaded into a vehicle if it is in the same location as the vehicle, thus making the property $into$ hold between the object and the vehicle. The $unload$ operator describes the opposite state transformation. It should be noted that some properties, such as $road$, are not mentioned in the effects, meaning that the corresponding ßuent values persist in the next state. A problem instance for this domain could be given by an initial state which describes a road map of connections, objects and vehicles distributed among locations. For instance, let the initial state be the following

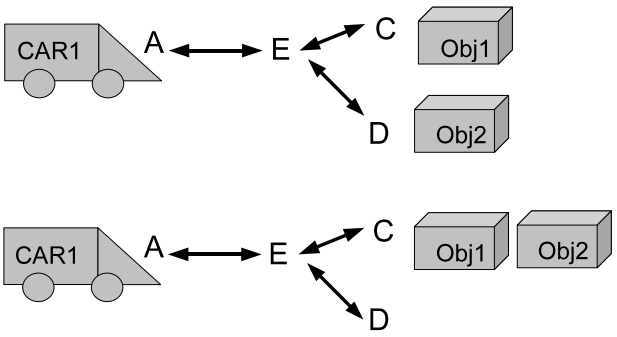

Figure 2: Initial and Þnal states

```
I={ (road_A_B) (road_B_C) (road_B_D)
    (road_B_A) (road_C_B) (road_D_B)
    (at_car1_A) (at_obj1_D) (at_obj2_C)  }
```

and the goal state

```
G={ (at_obj1_C) (at_obj2_C) (at_car1_A) }
```

which requires that in the Þnal state all objects must be in location `C` and `car1` is back in the initial location `A`. The initial state and the required Þnal goal state of the world are illustrated in Fig. 2. Despite the simplicity of the modeling formalism, the preconditions/effects approach is fairly general since quite diverse domains can be described. However, it has to be pointed out that the classical characterization of planning we have just described is based on some underlying simplifying hypotheses: the *closed world assumption*, i.e. the initial state of the world is completely known to the planner; *atomic time and sequential actions*, i.e., executing an action causes a state transition which is considered atomic and instantaneous, because the model does not describe what is happening during its execution, thus the only signiÞcant relation among actions is precedence; *deterministic effects*, i.e., there is no uncertainty about action execution Ñ once an action is executed the next state is completely determined by its effects, so actions cannot fail and external events are not possible since a state change cannot happen without an action causing it. Several research works are proposing extensions in order to overcome the limits and the applicability of the classical model to real world problems.

Extending the basic model. One important extension consists in introducing into planning models the ability to manage resources and time constraints by attaching numbers and metric quantities to action descriptions. The idea is to extend the concept of planning state by introducing numerical variables, which represent the state of resources and other costs, and extending, accordingly, the concept of numerical preconditions and numerical effects, the latter establishing how the variables are updated as a consequence of action execution. For instance, the description of operator *move_to* can be enriched with a *numerical precondition* about the metric quantities $fuel$ and $distance$ as

```
(distance ?from ?to ?km)
(> (fuel ?vehicle) (multiply ?km 1/8))
```

which requires to have enough fuel before applying the action. Moreover the *numerical effect*

```
(decrease(fuel ?vehicle)(multiply ?km 1/8))
(increase elapsed_time (divide ?km 50))
```

establishes how to decrease the fuel amount and how to compute the total elapsed execution time, assuming, for instance, a consumption of 8 litres per kilometer, and an average speed of 50 km/h. Similarly, making reference to starting/ending instants of actions allows to describe sophisticated scheduling constraints and goals in the planning problem. For instance, a possible numerical goal could be

```
G={(at_obj1_C)
(>= (minus (end_move_AB) (start_moveAB) 1/2)
(>= (fuel_car1) 40)}
```

which requires to move `obj1` at location `C` from `A` to `B` in less than half an hour, and still having at least 40 litres of fuel at the end of the plan. A quite natural further extension, derived from the introduction of numerical quantities, is to see planning as an *optimization problem*, in which one wants to obtain an admissible solution plan, i.e. it reaches the goals, and minimize/maximize some linear or nonlinear metric of costs or utilities, such as for instance

```
minimize(10*elapsed_time - 2*fuel_car1)
```

Moreover many planning models extend the expressiveness of the language with the possibility of using both quantifiers, e.g., using goal expressions like

```
(for_each ?obj (at C ?obj)),
```

conditional effects, and abstract action descriptions, where, for instance, the operator *transfer_to* is an abstraction of more detailed operators such as *drive_to* and *flight_to*. Time is devoted a special treatment in the extensions proposed to classical planning. Models exist which allow: action duration, parallelism, complex temporal constraints, external timed events and continuous time dependent quantities.

A further issue in planning has concerned the specification of explicit search control knowledge. The most used approach in this respect are Hierarchical Task Network (HTN) models [19, 20]. HTN is based on the notion of activity decomposition. The domain knowledge provided in HTN models describes how complex high level tasks can be decomposed into inter-related subtasks down to primitive tasks, e.g., the abstract task of "transferring objects from one place to another" is decomposed into the sequence of subtasks: "load objects into a vehicle", "transport to destination" and "unload them". A HTN planner is submitted some high level task as a goal, and the plan synthesis consists in a continuous activity of subtask expansions and plan refinement until a consistent network of primitive actions is obtained. HTN planners have been used successfully as the base of many practical planning systems, because they entail the "how to" procedural vision of the domain expert and the consequent limitation of search from first principles that characterizes the classical approaches. Their excellent performance is mainly due to the fact that procedural knowledge is encoded into the decomposition hierarchy. On the other hand, a drawback of HTN planners is the reduced flexibility, since they are unable to handle tasks that were not explicitly anticipated by the designer, even if the available primitive actions are sufficient for constructing a suitable solution plan.

Research progression. Research in planning evolved significantly over the last 20 years. At least two aspects are worth reminding, namely the introduction of the concept of planning graph [5], and the idea of planning as satisfiability [29, 30]. In the work of Blum and Furst [5] the classical idea of search in the state space is revived by a successful planner, Graphplan, which significantly outperformed any previous partial order planner. Graphplan certainly exploited the increased availability of memory and computing resources, but its key point was a very powerful structure, the planning graph. The planning graph can be seen as a very compact form of reachability tree [28], which represents the search space and some structural constraints of the problem until a fixed depth k. In the original work, the solution plan is extracted with a backward depth first search. Graphplan generated a huge amount of work (an example is [1]) concerning algorithms for solution extraction ranging from forward search to non-directional and heuristic search, as well as techniques for preprocessing, compacting and extending the planning graph structure. While previous research has been biased by a constructive approach to plan synthesis, the key idea in Graphplan consists in unfolding the domain theory into a specialized data structure and in the development of search techniques that reason on what is "in the domain" and not only on what is "in the current plan". The work of Kautz and Selman [29] marked the first idea of solving planning by reduction to another known problem, namely the propositional satisfiability problem (SAT). Exploiting, again, the planning graph, it was possible to devise Blackbox [30], a planner which iteratively transformed each expansion of the planning graph into a propositional formula and submits it to a SAT solver. A satisfying assignment of the formula represents a solution plan, and iterative expansion/encoding steps either yield a plan or a termination condition signifying that no plan exists is eventually reached. Using a standard representation for the formula, any available SAT solver from a standard library can be employed as a planner. Many techniques for propositional encoding of planning problems have been proposed by various authors.

Disregarding the theoretical aspects of planning as SAT, the remarkable contribution of Blackbox has been the general idea of approaching planning by problem reduction techniques. The Blackbox approach, in fact, decomposes the problem of planning into the two distinct sub-problems of finding the best encoding, and finding the best solver. Moreover, in a problem reduction scheme, the planner benefits from the improvements and research results in the targeted solvers. Successively, the problem reduction scheme has been used in several lines of planning research: CSP encoding has been extensively applied to planning [44, 17], as well as reduction to Integer Linear Programming (ILP) for planning with metric constraints and optimization goals [6, 45]. A further encoding uses model-checking techniques, e.g., [12, 18].

Addressing nondeterministic. Relaxing the closed world and deterministic assumptions of the classical approach has major consequences in the planning model, and in the same notion of solution plan. More realistic hypotheses can be assumed: actions can fail during execution; actions cannot produce the expected effects; the state of the world is not completely known, and it can change in an unpredictable way due to events which are not under the control of the planner. Moreover, we should assume the capacity of sensing the world during planning execution. Under these hypotheses a solution plan cannot be a predetermined sequence of actions, but it should reflect the possible courses of execution. This is the case of contingency planning models which extend the classical approach by building solution plans with conditional constructs for the various contingencies which can be detected during the execution. Following this same idea, a bulk of research concerned the coupling of planning and execution, e.g., [33], the synthesis of "universal plans"[40], a variant of the planning problem known as conformant planning [41, 13, 9], or planning with incomplete information and sensing [38, 14, 37]. All of them introduce into the classical framework the issues of robustness of a plan with respect to "expected"contingencies during execution. A fairly different approach to planning with uncertainty is to model the probabilistic nature of transitions between states by the widely used techniques of Markov Decision Processes (MDPs) [15, 8]. The action outcomes are labeled with their respective probabilities, and the notion of plan is represented in these models by the concept of policy, i.e., a decision structure which specifies which action should be taken in any possible situation. Models based on Partially Observable MPDs (POMDPs) allow us to manage situations where, as in the real cases, limited sensors and cost of sensing activity does not allow a complete observability of the resulting state. MDP and POMDP based planners face the main difficulty of the size of the state space, and an inadequate representation of time which is generally assumed to be atomic. However these techniques have proven to be useful and effective in some specific domains such as robot navigation tasks.

The planning competition. The first planning competition was held in 1998 and driven by Drew McDermott, who also led the effort to synthesize the first release of PDDL [35] as an interlingua to allow the comparison of planners on the same problems and domains. For the first time planners become comparable in a measurable way, albeit on an extension of the classical model. In later work PDDL has been significantly extended to deal for example with durative actions and numeric variables, and more recently to include preferences and plan constraints ñ see the site *http://zeus.ing.unibs.it/ipc-5/* for details. Since 2004, a competition on nondeterministic domains is also held. A clear positive effect of the competition has been the improvement of performance of planning systems and the fostering of innovative ideas. As an example it is worth mentioning a number of planners that have shown excellence in the various IPCs, and afterwards have been influential for the whole area, such as HSP [7], FF [27], LPG [25], CPT [46] and others. A side effect of the success of the IPC could be the focalization of research topics around issues relevant to the advancements of the competition with a decrease of intellectual efforts on a number of connected research topics. In particular the bias represented by the specific language used in the competition can limit some of the possible developments on generalized planning languages for real world applications.

3 Constraint-Based Scheduling

A complete account of scheduling techniques is outside the scope of this paper. We here present shortly the problem, a single example of state of the art AI scheduling technology, and then provide some comments with respect to the integration with planning techniques.

A scheduling problem. Scheduling is primarily concerned with figuring out *when* a set of predefined tasks should be executed so that the final solution guarantees "good"performance relatively to the optimization of an objective function. Let us focus on a family of problems known as *project scheduling*, whose main elements are the following: (a) a set $A = \{a_1, \ldots, a_n\}$ of *activities* or tasks. Every activity is characterized by a processing time p_i; (b) a set $R = \{r_1, \ldots, r_m\}$ describing the *resources* required to execute the activities. The execution of each activity a_i can require an amount req_{ik} of resource r_k during its processing time. Various kinds of resources can be taken into account: disjunctive or cumulative, renewable or consumable, among others; (c) *constraints*: the constraints are rules that limit the possible allocations of the activities. They can be divided into two types: (1) *resource constraints* limit the maximum capacity of each resource. For example, there may only be a certain number of machines or people available to work on some activities at any given time; (2) *temporal constraints* impose limitations on the times in which activities can be scheduled. A binary constraint is imposed between two activities, for instance in order to mutually bind the instant of occurrence of their start

times. Such constraints are often formulated as bounds on differences between activity start/end time points [16]. A scheduling problem consists in computing a consistent assignment of start and end times for each activity satisfying both resource and temporal constraints while optimizing a certain evaluation function, e.g., the early finish time of the whole set of activities or *makespan*. The particular constraints expressed in project scheduling problems are interesting because they allow to model a quite broad range of real domains that require modeling causal relations between activities, coordination between multiple steps, and a rich variety of time and resource constraints.

CSPs and scheduling. Scheduling problems are very hard problems: for instance, simple scheduling problems like job-shop scheduling are NP-hard [24]. Therefore, scheduling represents an important application for constraint directed search. A Constraint Satisfaction Problem (CSP) consists of a network of constraints defined over a set of variables where a solution is an assignment to the variables that satisfies all constraints. Constraint Programming is an approach to solving combinatorial optimization problems based on the CSP representation [34]. The framework involves the combination of sophisticated search technology and constraint propagation. Constraint propagation actively uses constraints to prune the search space. These propagation techniques are generally polynomial w.r.t. the size of the problem, and aim at reducing the domains of variables involved in the constraints by removing the values that cannot be part of any feasible solution. Various techniques with distinct pruning power can be defined for different types of constraints. The search for a solution to a CSP can be viewed as modifying a constraint graph by adding and removing constraints, where the constraint graph is an evolving representation of the search state, and a solution is a state with a single value remaining in the domain of each variable, and all constraints are satisfied. Several constraint programming approaches have been developed in this direction. For instance, the reader can refer to [2] for a thorough analysis of constraint-based techniques for scheduling problems. The work of constraint-directed scheduling of the 80s (see for example [42, 39]) developed into Constraint-based Scheduling approaches in the late 90s (see for example [36, 3]). These approaches are based on the representation of a scheduling problem and the search for a solution to it by focusing upon the constraints in the problem. A typical formulation of the problem is that of finding a consistent assignment of start times for each goal activity. Under this model, decision variables are time points that designate the start times of various activities and CSP search focuses on determining a consistent assignment of start time values.

Precedence constraint posting. Other approaches to scheduling operate with a problem formulation more akin to least-commitment frameworks. According to this formulation, referred to as *Precedence Constraint Posting* [43, 11], the goal is to post sufficient additional precedence constraints between pairs of activities for the purpose of pruning all inconsistent allocations of resources to activities. This short section shows a basic method to generate solutions to project scheduling proposed in [11] for problems with binary resources, and then extended to more general problems in later work, e.g., [10]. It is is based on the fact that the relevant events on a scheduling problem can be represented as a temporal CSP, usually called Simple Temporal Problem (STP) [16].

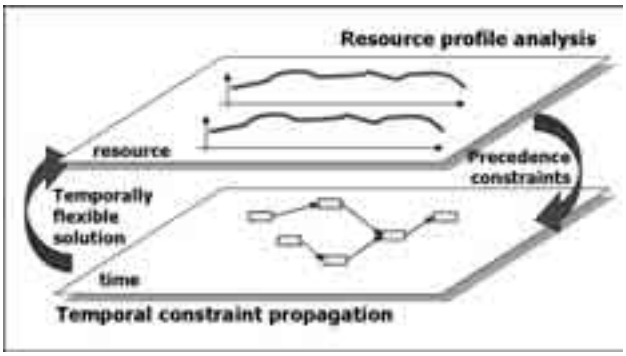

Figure 3: Precedence Constraint Posting Schema

The search schema used in this approach focuses on decision variables which represent conflicts in the use of the available resources; the solving process proceeds by ordering pairs of activities until all the current resource violations are removed. This approach is usually referred to as the Precedence Constraint Posting (PCP) because it revolves around imposing precedence constraints to solve the resource conflicts, rather than fixing rigid values to the start times. The general schema of these approaches is described in Fig. 3. It consists in representing, analyzing, and solving various aspects of the problem in two separate layers, namely the temporal layer and the resource layer. In the former, the temporal aspects of the scheduling problem like activity durations, constraints between pairs of activities, due dates, release time, etc., are considered. The latter represents and reasons upon the resource aspects of the problem.

Reasoning on time and resource constraints. The temporal aspects of scheduling problems are represented through an STP (simple temporal problem) network [16]. This is a temporal graph in which the set of nodes represents a set of temporal variables named time points, tp_i, while temporal constraints, of the form $tp_i - tp_j \leq d_{ij}$, define the distances among them. Each time point has initially a domain of possible values equal to $[0, H]$ where H is the horizon of the problem (H can be infinite). The problem is represented by associating with each activity a pair of time points which represent, respectively, the start and the end time of the activity. A temporal constraint between two time-points may define either ordering constraints between two activities (when the two time-points do not belong to the same activity) or activity durations (when the two time-points belong to the same activity). By propagating the temporal constraints, it is possible to bound the domains of each time point, $tp_i \in [lb_i, ub_i]$. In

the case of empty domains for one or more time points the temporal graph does not admit any solution. The STP propagation is polynomial [16]. The temporal layer maintains in polynomial time (using constraint propagation) a set of solutions deÞned by a temporal graph. This result is taken as input by the second layer. The resource layer takes into account the other aspect of the scheduling problem, namely resources. The problem is that there are constraints on resource utilization (i.e., capacity). Resources can be single- or multi-capacity, and reusable or consumable. The input of this layer is a temporally ßexible solution Ñ a set of temporal solutions. Like in the previous layer, it is possible to use constraint propagation (i.e., resource propagation) to reduce the search space. Even though there are several methodologies described in the literature (e.g., see [36, 32]) these are not sufÞcient in general. In fact they are not complete, that is, they are not able to prune all inconsistent temporal solutions. Therefore, it is necessary to introduce a method for deciding among the possible alternatives. For this reason a PCP procedure uses a *Resource ProÞle* to analyze resource usage over time and detect periods of resource violations and the set of activities, or *contention peaks*, which create this situation. The procedure then proceeds to post additional constraints on the time layer to level (or solve) one or more detected contention peaks. These new constraints are propagated in the time layer to check the temporal consistency. Then the time layer provides a new temporally ßexible solution that is analyzed again using the resource proÞles. The search stops when either the temporal graph becomes inconsistent or the resource proÞles are consistent with the resource capacities. The outcome of a PCP solver is a STP that not only contains the temporal constraints belonging to the initial problem, but also the additional precedences which have been added during the resolution process. If we observe Fig. 3 at a higher level of abstraction we may say that solving a scheduling problem can be seen as the problem of synthesizing temporal evolutions of the resource proÞles that are consistent with all the problem constraints. In addition, the PCP method shows quite well how the process of solving a scheduling problem consists in reasoning on time and resource constraints in order to Òcreate spaceÓ on the resource allocation of activities with the aim of ruling out contention peaks. This Òpeak-ßatteningÓprocedure represents the combinatorial core of the scheduling process. Given its high computational complexity, peak ßattening is carried out with incomplete algorithms.

Integrating planning and scheduling. Going back to how the current temporal planners represent resources, that is by means of numeric variables, it is possible to see what is missing to exactly integrate the two types of reasoning into more powerful solvers. The main activity of planners consists of deciding action selection, while Òcreating spaceÓ on resources cannot be reasoned upon explicitly and is considered somehow a side effect of the planning activity. As a consequence, for problems strongly characterized by complex resource constraints, the performance of current planner somehow relies on search tools that reason on planning knowledge and, as a consequence, are not tailored for resource reasoning. This is an area for further research to create a generation of solvers that take complete advantage from both research traditions.

4 Current Research Scenario

The current research arena has more ramiÞcations with respect to those described here. In particular, most of the current efforts focus on the problem with planning in non-deterministic domains and a signiÞcant part of the community is interested in developing techniques for either decision-theoretic, probabilistic and conformant planning with several combinations of those features. These research directions require a speciÞc survey. A scheduling counterpart of these efforts is concerned with schedule robustness, a topic that is attracting increasing research. Related to this is the problem of P&S and execution, an issue which is relevant for all applications connected to autonomy. A set of issues are connected to the problem of creating P&S technology which is deployable to solve real-world problems. In this context, a few very important topics which are somewhat under-addressed are knowledge acquisition, knowledge engineering, validation and reÞnement, as well as problems connected with user-interaction, such as mixed-initiative problem solving and automated explanation generation. In general, there is an increasing awareness of the fact that, in order to include P&S in real domains, research should not be centered only on search control but it should also focus on a more general approach to the problem of plan life-cycle management.

REFERENCES

[1] M. Baioletti, S. Marcugini, and A. Milani. DPPlan: an Algorithm for Fast Solutions Extraction from a Planning Graph, *Proc. of AIPS-00*, 2000.

[2] P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling*. Kluwer Academic Publishers, 2001.

[3] J. C. Beck, E. D. Davenport, A. J. Davis, and M. S. Fox. The ODO Project: Towards a UniÞed Basis for Constraint-Directed Scheduling. *Journal of Scheduling*, 1:89–125, 1998.

[4] T. Bylander. The Computational Complexity of Propositional STRIPS Planning. *ArtiÞcial Intelligence*, 69(1–2):165–204, 1994.

[5] A.L. Blum, and M.L. Furst. Fast Planning Through Planning Graph Analysis. *ArtiÞcial Intelligence*, 90:281–300, 1997.

[6] A. Bockmayr and Y. Dimopoulos. Mixed integer programming models for planning problems. *Proc. of CPÕ98 Workshop on Constraint Problem Reformulation*, 1998.

[7] B. Bonet, and H. Geffner. Planning as heuristic search. *ArtiÞcial Intelligence*, 129:5–33, 2001.

[8] A. Cassandra, L. Kaebling, M. Littman. Acting Optimally in Partially Observable Stochastic Domains. *Proc. of AAAI-94*, 1994.

[9] C. Castellini, E. Giunchiglia, and A. Tacchella. SAT-based planning in complex domains: Concurrency, constraints and nondeterministic. *Artificial Intelligence*, 147(1–2):85–117, 2003.

[10] A. Cesta, A. Oddi, and S. F. Smith. A Constraint-based method for Project Scheduling with Time Windows. *Journal of Heuristics*, 8(1):109–136, 2002.

[11] C. Cheng and S. F. Smith. Generating Feasible Schedules under Complex Metric Constraints. In *Proc. of AAAI-94*, 1994.

[12] A. Cimatti, E. Giunchiglia, F. Giunchiglia, and P. Traverso. Planning via Model Checking: A Decision Procedure for *AR*. In *Proceedings of ECP-97*. pp. 130–142, LNAI 1348, 1997.

[13] A. Cimatti, and M. Roveri. Conformant Planning via Symbolic Model Checking. *J. Artif. Intell. Res.*, 13:305–338, 2000.

[14] A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1—2):35–84, 2003.

[15] T. Dean, L. Kaebling, J. Kirman, and A. Nicholson. Planning with Deadlines in Stochastic Domains. *Proc. of AAAI-93*, 1993.

[16] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[17] M. Do, and S. Kambhampati. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artificial Intelligence* 132:151-182, 2001.

[18] S. Edelkamp Taming Numbers and Durations in the Model Checking Integrated Planning System. *J. Artif. Intell. Res.*, 20:195–238, 2003.

[19] K. Erol, J.A. Hendler, D.S. Nau. UMCP: A Sound and Complete Procedure for Hierarchical Task-network Planning. *Proc. of AIPS-94*, 1994.

[20] K. Erol, J.A. Hendler, D.S. Nau. HTN Planning: Complexity and Expressivity. *Proc. of AAAI-94*, 1994.

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

[22] R.E. Fikes, and P.E. Hart, and N.J. Nilsson. Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, 3:251–288, 1972.

[23] M.S. Fox, and S.F. Smith. ISIS —a knowledge-based system for factory scheduling. *Expert System*, 1, 1:25–49. 1984.

[24] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1979.

[25] A. Gerevini, A. Saetti, and I. Serina. Planning Through Stochastic Local Search and Temporal Action Graphs in LPG. *J. Artif. Intell. Res.*, 20:239–290, 2003.

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

[27] J. Hoffmann, and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res.*, 14:253–302, 2000.

[28] S. Kambhampati, E. Parker, and E. Lambrecht. Understanding and Extending Graphplan. In *Proceedings of ECP-97*. pp. 260–272, LNAI 1348, 1997.

[29] H. Kautz and B. Selman. Planning as satisfiability. *Proc. of ECAI-92*, 1992.

[30] H. Kautz and B. Selman. Blackbox: Unifying sat-based and graph-based planning. *Proc. of IJCAI-99*, 1999.

[31] A.K. Jonsson, and P.H. Morris, and N. Muscettola, and K. Rajan, and B.D. Smith. Planning in Interplanetary Space: Theory and Practice. *Proc. of AIPS-00*, 2000.

[32] P. Laborie. Algorithms for Propagating Resource Constraints in A.I. Planning and Scheduling: Existing Approaches and New Results. *Artificial Intelligence*, 143(2):151–188, 2003.

[33] S. Lemai and F. Ingrand. Interleaving Temporal Planning and Execution in Robotics Domains. *Proc. of AAAI-04*, 2004.

[34] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

[35] D. McDermott, and and the AIPS-98 Planning Competition Committee. PDDL —The Planning Domain Definition Language. *Tech. Rep. CVC TR-98-003/DCS TR-1165*, Yale University, 1998.

[36] W.P.M. Nuijten and E.H.L. Aarts. A Computational Study of Constraint Satisfaction for Multiple Capacitated Job Shop Scheduling. *European Journal of Operational Research*, 90(2):269–284, 1996.

[37] M. Oglietti. Understanding planning with incomplete information and sensing. *Artificial Intelligence*, 164(1–2):171–208, 2005.

[38] R.P.A. Petrick, and F. Bacchus. A Knowledge-Based Approach to Planning with Incomplete Information and Sensing, *Proc. of AIPS-02*, 2002.

[39] N.M. Sadeh. *Look-ahead Techniques for Micro-opportunistic Job Shop Scheduling*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh PA, March 1991.

[40] M. Shoppers. Universal plans for reactive robots in unpredictable environments. *Proc. of IJCAI-87*, 1987.

[41] D.E. Smith, and D.S. Weld. Conformant Graphplan In *Proc. of AAAI-98*, 1993.

[42] S.F. Smith. OPIS: A Methodology and Architecture for Reactive Scheduling. In [47], 1994.

[43] S.F. Smith and C. Cheng. Slack-based Heuristics for Constraint Satisfactions Scheduling. In *Proc. of AAAI-93*, 1993.

[44] P. Van Beek and X. Chen. Cplan: A constraint programming approach to planning. *Proc. of AAAI-99*, 1999.

[45] M. Van den Briel and S. Kambhampati. Optiplan: Unifying IP-based and Graph-based Planning. *J. Artif. Intell. Res.*, 24:919–931, 2005.

[46] V. Vidal, and H. Geffner, Branching and Pruning: An Optimal Temporal POCL Planner Based on Constraint Programming. *Artificial Intelligence*, 170:298–335, 2006.

[47] M. Zweben, and M.S. Fox (Eds.). *Intelligent Scheduling*, Morgan Kaufmann, 1994.